



HelixSwarm

Helix Swarm Guide

2022.2
August 2022

PERFORCE

www.perforce.com



Copyright © 2013-2022 Perforce Software, Inc..

All rights reserved.

All software and documentation of Perforce Software, Inc. is available from the [Perforce](#) website. You can download and use Perforce programs, but you can not sell or redistribute them. You can download, print, copy, edit, and redistribute the documentation, but you can not sell it, or sell any documentation derived from it. You can not modify or attempt to reverse engineer the programs.

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

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

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

Perforce and Inter-File Branching are trademarks of Perforce.

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

Any additional software included within Perforce is listed in "[License statements](#)" on page 1363.

Contents

1 About P4 Code Review	19
P4 Code Review videos	19
Related topics	19
What's new in P4 Code Review	21
2025.3	21
2 Quickstart	23
P4 Code Review quick reference	23
Menu quick reference	23
Quick URLs quick reference	26
My Dashboard page quick reference	27
Global dashboard page quick reference	30
Reviews list page quick reference	34
Review page quick reference	39
P4 Code Review user tasks	60
Start a code review	61
Edit the reviewers on a review	63
Check for code reviews you need to act on	64
List reviews you are involved with	65
Contribute comments or code changes to a code review	66
Get a local copy of the code in a review for evaluation	69
Fix errors that cannot be merged in a review	70
Change the author of a review	72
Change your user notification settings	73
Unfollow all projects and users you are following	73
P4 Code Review administration tasks	74
Change notification settings for a group	74
Manage project branches	75
Unfollow all projects and users for another user	83
Obliterate a review	84
Change the logging level	85
Check the queue workers	85
Automatically deploy code within a review	87
3 Install and upgrade P4 Code Review	89
Important restrictions	89
Review the runtime dependencies	89
P4 Code Review and P4 Server installation considerations	89

Choose the installation process	90
Upgrade P4 Code Review	91
Runtime dependencies	91
Recommended operating systems	92
Apache web server	93
PHP	94
P4 Server requirements	98
P4 Server event notification	99
Worker dependencies	101
Redis server	101
Supported P4 Visual Client (P4V)	102
Supported P4 AS (HAS)	102
Supported web browsers	102
Workflow prerequisites	102
Optional dependencies	103
Security-enhanced Linux (SELinux)	103
Choose the installation process	104
Install and configure P4 Code Review from a package (recommended)	106
Install and configure P4 Code Review on Ubuntu	106
Install and configure P4 Code Review on RHEL	118
Install and configure P4 Code Review on Amazon Linux 2	140
Run P4 Code Review using a Docker container	156
Prerequisites	157
P4 Code Review images	157
Running the Docker container	157
Persisting Docker containers for a production environment	159
Restarting the Docker container	160
Migrating P4 Code Review to a Docker container	161
Advanced configuration options	162
Example of running Swarm using docker-compose	165
Install and configure P4 Code Review manually from a Tarball	166
Redis configuration	167
Apache configuration	171
PHP configuration	174
P4 Code Review configuration	179
Configuring P4 Server event notification	182
Copy the Perl trigger script on to your P4 Server	189
Configure the API token for the P4 Code Review trigger	190
Ensure trigger script works	192

Update the P4 Server triggers table	193
Configure the P4 Server to promote all shelved changes	198
Optional installation steps	198
Copy the Perl trigger script on to your P4 Server	200
Configure the API token for the P4 Code Review trigger	200
Ensure trigger script works	202
Update the P4 Server triggers table	203
Configure the P4 Server to promote all shelved changes	207
Optional installation steps	208
Set up a recurring task to spawn workers	208
Set up a cron job to delete AI summaries	212
Common post-installation options	214
Post-install configuration options	214
Validate your P4 Code Review installation	227
Troubleshooting: Azure Application Gateway issues with P4 Code Review and P4V	228
Troubleshooting: Review not created	228
Troubleshooting: Review cannot be updated using triggers	229
Upgrading P4 Code Review	231
Upgrading a Ubuntu package installation	231
Upgrading a RHEL package installation	243
Upgrading a Docker container	258
Upgrading a tarball installation	260
Moving your P4 Code Review instance	278
Summary	278
Installing P4 Code Review on your new server	278
Copying your existing P4 Code Review configuration	279
Configuring P4 Server event notification	279
Replacing your original P4 Code Review instance with your new instance	283
Validating your new P4 Code Review instance	284
Additional security measures	285
Hide your Apache version and Linux OS	285
Disable Apache directory listings	287
Security risks of using a self-signed certificate	287
4 Basics	288
My Dashboard	288
Filtering	289
Review summaries	289
Global Dashboard	291
Toolbar	293

Sidebar	296
P4 Server dashboards	298
Navigating directly to a specific P4 Server instance in P4 Code Review	301
Activity	302
Files	303
Browsing deleted files and folders	304
File display	305
File edit	309
Download files as a ZIP archive	310
Supported syntax highlighting in the Review page	311
Commits	313
Range filter	314
File Commits	314
Remote depot commits	315
Jobs	315
Edit the Job columns	316
Job display	316
Add jobs	318
Unlinking jobs	319
Changelists	319
Changelist display	319
Diffs	322
Viewing a diff	324
Comments	328
Adding comments	328
Editing comments	331
Tasks	332
Comment features	336
Mark comments as read	341
Mark comments as unread	342
Archiving comments	343
Restoring comments	343
Users	344
Viewing your user profile	344
Viewing another user's profile	350
Viewing another user's profile when you have admin or super user privileges	351
Groups	352
Listing groups	353
Viewing a group	354

Projects	360
Listing projects	360
Viewing a project	361
Private projects	366
Workflows	367
Listing workflows	368
Viewing a workflow that you own	369
Viewing a shared workflow that you do not own	370
Viewing the global workflow	370
Work-in-progress tag	371
Summary	372
Using the work-in-progress tag	372
Tests	373
Listing tests	374
Viewing a test that you own	375
Viewing a shared test that you do not own	376
Notifications	376
@mention notifications	380
(1) Committed change notifications	381
(2) Review start notifications	382
(3) Moderator notifications	382
(4) Group member notifications	382
(5) Disable notifications	383
(6) Comment author notifications	383
(7) Tests have finished	383
Log in/Log out	383
Log in with a password	383
Log in with SSO	384
Log out of P4 Code Review	384
require_login	384
Notable minor features	385
Quick URLs	385
Links in descriptions and comments	387
Search	389
Jira integration	389
Avatars	390
Following	390
Time	390
Keyboard shortcuts	390

About P4 Code Review	392
Custom error pages	392
Short links	393
Markdown	393
Common text styles	393
Text styling example	395
Markdown in comments and review descriptions	397
Markdown in projects	398
5 Code reviews	400
Benefits	400
Facilities	400
Workflow	401
Models	401
Pre-commit model	402
Post-commit model	402
Internal representation	402
Reviews list	406
Review filters	408
Review display	411
Review summary	413
Review state	413
Vote buttons	414
Change state button	415
Review actions button	415
Review description	421
Information panel	423
Files tab	430
Comments tab	441
Activity tab	442
Activities	443
Start a review	443
Update a review	445
Fetch a review's files	446
Deleting shelves	447
Edit reviewers	448
Add a changelist to a review	450
Append a pending changelist to a review	451
Replace review with a pending changelist	452
Replace review with a committed changelist	453

Responsibility	454
Authors	454
Moderators	455
Required reviewers	457
Add yourself as a reviewer	458
Remove yourself as a reviewer	458
Review workflow	458
Basic review workflow	459
Additional workflow tasks	459
Review creation and modification outside of P4 Code Review	462
States	464
Self-approval by review authors	465
State change restrictions with moderation	465
State change restrictions without moderation	467
Required reviewers	468
Change review state	468
6 Groups	471
Add a group	471
Edit a group	476
Delete a group	476
7 Projects	477
Add a project	477
Project settings	478
Participants tab	479
Branches tab	483
Integrations tab	494
General Settings tab	495
Edit a project	497
Membership	497
Add a member	498
Remove a member	499
Owners	500
Moderators	500
Default reviewers	502
Retain default reviewers	503
Delete a project	505
When an owner is deleted outside of P4 Code Review	506
8 Workflows	507
Why should I use P4 Code Review workflow?	507

Workflow overview	509
Workflow rules	510
Workflow basics	512
Example workflows	513
Merging multiple workflows	515
Add a workflow	521
Edit a workflow	526
Delete a workflow	526
9 Tests	528
Add a test	528
Pass special arguments to your test suite	531
Edit a test	534
Delete a test	534
10 Integrations	535
Jira	535
Jira integration	536
Jira integration and Perforce job links	538
LibreOffice	543
Limitations	543
Installation	544
Zip archive	544
Installation	544
Configuration	544
Download files as a Zip archive	545
TeamCity	546
11 Administration	547
AI for code analysis	547
How to set up OpenAI integration for P4 Code Review	549
How to set up LM Studio integration for P4 Code Review	550
How to set up a generic AI model integration for P4 Code Review	551
How to write the Custom AI Adapter in P4 Code Review to support an in-house AI model	555
Automated deployment for reviews	562
Automated testing for reviews	563
Configuring Jenkins for P4 Code Review integration	566
Avatars	570
Disable avatar lookups	572
Backups	572

Changelist files limit	573
Client integration	573
Customized P4 Code Review installations	574
Comment attachments	575
Comment attachment storage	576
Maximum comment attachment size	577
Comment mentions	577
Regular expressions in exclude lists	579
Comments	580
Comment notification delay	580
Comment threading	581
Commit credit	582
Commit-edge deployment	582
P4 Code Review commit-edge configuration	583
P4V Authentication	584
Commit timeout	584
Configuration overview	585
Diff configuration	594
max_diffs	595
max_total_diff_size	595
max_size_chunk	595
Email configuration	595
Sender	597
Transport	597
Recipients	598
notify_self	598
Use BCC	598
Use Reply-To	598
Email subject prefix	599
Save all messages to disk	599
Email headers	599
Email thread indexing	600
Filter review related emails	601
Emoji	601
Environment	602
mode	603
hostname	603
external_url	603

base_url	604
logout_url	605
emoji_path	605
Excluding Users from Activity Streams	606
Extensions Management	606
List	607
Delete	608
Install	608
Config	609
Tokens	611
Disable	611
Enable	612
Save	612
Load	612
Ping	613
Files configuration	613
max_size	613
download_timeout	615
allow_edits	615
Groups configuration	616
High Availability	616
Set up active-passive in P4 Code Review	617
Pre-disaster preparations	617
Volume sharing between P4 Code Review instances	618
Failover from active node to passive node	618
Why the passive node does not require the setup script	618
Ignored users for reviews	619
Job keywords	619
Adding a new job keyword	620
License	622
Localization & UTF8 character display	622
Localization	623
Non-UTF8 character display	624
UTF8 conversion	624
Logging	625
Web server logging	625
P4 Server logs	625
P4 Code Review logs	625
Logrotate	627

Reference ID	627
Trigger Token Errors	628
Performance logging	628
Mainline branch identification	629
Regular expressions	630
Project mainline and README check	631
Markdown	631
Menu helpers	632
Add a menu item to a menu	633
Make Groups menu item visible to admin-user and above only	635
Create a menu item using only the custom module name	636
Add a menu item to the project menu of a specific project	636
Multiple P4 Server instances	638
Set up the P4 Code Review configuration file for the P4 Servers	639
Configure event notification for each P4 Server	641
Configure a cron job for each P4 Server instance	648
Further information	650
Notifications	652
Global settings	654
Obliterate Review	657
When you obliterate a review	657
Obliterate a review	658
P4TRUST	658
If a certificate changes	658
Projects	659
Restrict project name and branch definition editing to administrators	659
Limit adding projects to administrators	659
Limit adding projects to members of specific groups	660
Project readme	660
Projects tab initial fetch	661
Allow project members to view project settings	661
Prevent users from creating a branch that includes paths to which they do not have permission	662
Redis server	662
P4 Code Review connection to Redis	663
Redis server configuration file	665
Manually verify the Redis caches	665
Review cleanup	666
Review keyword	668

Review keyword configuration	668
Create a review	669
Add a changelist to a review	670
Remove mentions from review descriptions	672
Reviews filter	673
filter-max	674
result_sorting	674
date_field	674
Reviews	674
Allow author change	675
Allow author obliterate review	675
Disable approve for reviews with open tasks	676
Disable self-approval of reviews by authors	676
Disable tests on approve and commit	676
Protected end states	677
Expand all file limit	678
Expand group reviewers	678
Maximum initial reviewers and reviewer groups in secondary navigation bar	679
Moderator behavior when a review spans multiple branches	679
More context lines	680
Process shelf file delete when	680
Synchronize review description	681
Review complexity	682
Maximum files limit	683
Automatically resolve outdated files	684
Search	684
maxlocktime	685
p4_search_host	685
Comment attachment thumbnail and blur generation	685
Security	686
Require login	686
Prevent login	686
Auto-create new users	687
Sessions	687
Security headers	689
Disable commit	695
Restricted Changes	695
Limit adding projects to administrators	696
Limit adding projects to members of specific groups	696
IP address-based protections emulation	696

Disable system info	698
HTTP client options	698
Strict HTTPS	700
Apache security	700
PHP security	701
proxy_mode	701
Short links	702
Single Sign-On PHP configuration	703
P4 AS	704
Skip activity	705
Slack integration	706
Prerequisites	706
Install the Slack integration	706
Slack configuration	707
Use Slack integration through a proxy server	708
Setup project configurations	709
Add P4 Code Review application to a private channel	711
Slack notification trigger	711
swarm_root	712
System Information	712
Perforce	713
Log	714
PHP Info	714
Queue Info	715
Cache Info	719
Tag processor	720
wip tag configuration	721
Test definitions	722
project_and_branch_separator	722
Trigger options	723
Command-line options	723
Configuration items	727
Unapprove modified reviews	729
Uninstall P4 Code Review	729
Background	730
Uninstall steps	730
Upgrade index	732
status_refresh_interval	732

batch_size	732
Users	733
Dashboard refresh interval	733
Display full names	733
Review preferences	734
Time preferences	735
How P4 Code Review handles deleted users	736
Workers	740
Worker status	741
Worker configuration	741
Manually start workers	742
Manually restart workers	742
Workflow global rules	742
Workflow prerequisites	743
workflow	743
Global workflow	744
12 Extending P4 Code Review	751
Resources	751
jQuery	751
JavaScript resources	751
PHP resources	751
Laminas Framework resources	751
Development mode	752
To enable development mode:	752
To disable development mode:	752
Modules	753
Module overview	753
Task details	762
Example email module	767
CSS & JavaScript	776
Sample JavaScript extensions	777
Sample CSS customizations	778
CSS customization when P4 Code Review is connected to multiple P4 Server instances	781
P4 Code Review API	783
API versions	783
Current API versions	783
Unsupported API versions	784
Get P4 Code Review version information	785

P4 Code Review API basics	786
Authentication	786
Requests	787
Pagination	789
Responses	789
API endpoints (v9)	789
Activity : P4 Code Review Activity List	789
Cache: P4 Code Review Cache	801
Changes : API controller providing a service for changes	813
Comments : P4 Code Review Comments	817
Groups : P4 Code Review Groups	834
Login : P4 Code Review Login API	851
Projects : P4 Code Review Projects	863
Reviews : P4 Code Reviews	888
Servers : P4 Code Review Servers API	932
Users : P4 Code Review Users	933
Workflows : Controller for querying, creating and updating workflows	937
API (v9) examples	961
API endpoints (v10)	973
Activity: P4 Code Review activity	973
AiAnalysis: P4 Code Review AI code analysis	990
Attachments: P4 Code Review comment attachments	1005
Changes: P4 Code Review changes	1012
Comments: P4 Code Review comments	1026
Files: P4 Code Review files	1072
Groups: P4 Code Review groups	1096
Jobs: P4 Code Review Perforce jobs	1100
Login: P4 Code Review login	1103
Menus: P4 Code Review main menu items	1118
Projects : P4 Code Review Projects	1124
Reviews: P4 Code Reviews	1147
Search: P4 Code Review search	1257
Specs: Spec fields	1269
Testdefinitions: P4 Code Review Test definitions	1276
Testruns: P4 Code Review Test Integration	1286
Users: P4 Code Review users	1312
Workflows : P4 Code Review workflows	1321
Glossary	1347
Getting help	1362
License statements	1363

1 | About P4 Code Review

P4 Code Review enables collaboration and code review for teams using P4 Server, helping ship quality software faster. P4 Code Review is a collaboration tool for all types of intellectual property. It unites teams to foster brainstorming, formalize content reviews, and keep projects moving forward. Contributors share files, comment, suggest tasks, vote up or down, and submit work directly within its elegant, web-based interface. P4 Code Review automates the entire process via notifications and makes it easy to monitor progress.

P4 Code Review videos

Watch the following videos for an overview of P4 Code Review and to understand its basic operation:

Introduction to P4 Code Review

Create a project in P4 Code Review

Create a review in P4 Code Review

Collaborate on a review in P4 Code Review

Related topics

Getting started:

- If you are experienced with code review but unfamiliar with P4 Code Review, start with the [Quick reference](#) and [Quickstart](#) sections.
- If you are unfamiliar with code review, start with the [Basics](#) section.
- If you are unfamiliar with the P4 Code Review workflow feature, start with the [Workflow overview](#) section.
- For a step-by-step walk-through of setting up a workflow, adding it to a project, and using it to progress through a review, see the [P4 Code Review Documentation](#).

Users:

- [Code reviews](#)
- [Projects](#)

- [Groups](#)
- [Workflows](#)

Administrators:

- [P4 Code Review administration](#)
- [P4 Code Review Integrations](#)
- [Extending P4 Code Review](#)
- [Install and configure P4 Code Review](#)
- ["Upgrading P4 Code Review" on page 231](#)

What's new in P4 Code Review

2025.3

Released: September 2025

P4 Code Review includes the following key features and enhancements. For a complete list of what's new and bug fixes in this release, see the [release notes](#).

Use wildcards for simpler project branch paths

For more flexible project branch definitions, you can now use the `*` wildcard anywhere in the file path. Previously, only the `...` wildcard was allowed in a file path, and only at the end.

For example, branch paths previously looked like this:

```
//ProjectY/UserZ/VersionA/LibA/SubLib1/Data1/...  
//ProjectY/UserZ/VersionA/LibA/SubLib2/...  
//ProjectY/UserZ/VersionB/LibA/SubLib1/Data1/...  
//ProjectY/UserZ/VersionB/LibA/SubLib2/...
```

After introducing the `*` wildcard, the paths can be simplified to:

```
//ProjectY/UserZ/*/LibA/SubLib1/Data1/...  
//ProjectY/UserZ/*/LibA/SubLib2/...
```

On the Project branch page, click **Expand wildcards** to simplify the branch patch that contain wildcards.

Comment improvements

- Attach supporting files to a comment without adding text.
- To post a comment, use the new keyboard shortcut: (**Ctrl/Cmd + Enter**).

Improved performance for large file diffs

For improved performance, large diffs in code reviews are now paginated. Previously, a diff was loaded and displayed all at once. Now, the first section of data is displayed and **Show next** and **Show previous** buttons are available for navigating the entire diff. If a diff is truncated, a banner at the top of the file informs you. This functionality is also available through the API endpoints.

Review page enhancements

- Toggle syntax highlighting on the review page by using a new keyboard shortcut: (**Shift + Alt + H**).
- Expand or collapse all files on the review page by selecting **Expand all files** and **Collapse all files** in the Diff Actions menu .
- Experience better navigation through fixes to auto-scrolling issues, including jittering file lists, lost scroll position when expanding files, and inaccurate docked file headers in reviews.

2 | Quickstart

Often, the best way to learn how software works is to try it. If the interface is sufficiently discoverable, you may not need to refer to documentation much, or at all. Sometimes, you just need to see the required steps to complete a task; that's what this chapter is all about.

In this section:

P4 Code Review quick reference

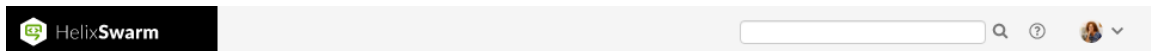
This section contains quick reference guides to help to familiarize you with P4 Code Review:



In this section:

Menu quick reference

The Header, Main, and Project menus are used to quickly navigate around P4 Code Review.

Header

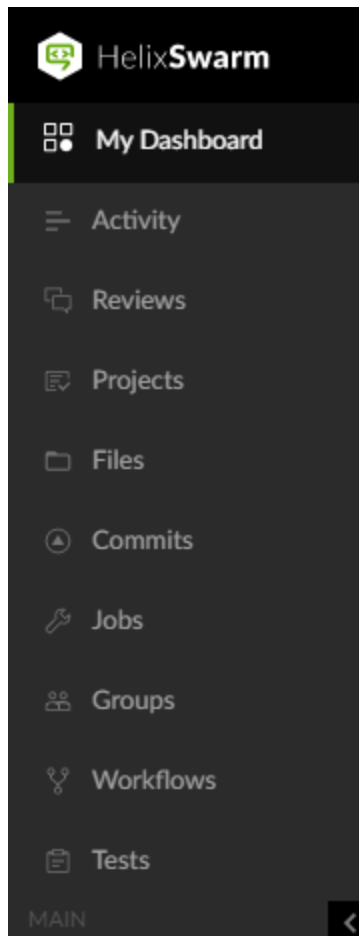


- Click on the P4 Code Review logo to see a list reviews that need your attention, see the [My Dashboard](#) page.

- **Search box:** search for projects, groups, users, and files that match your search string.
-  click to view the P4 Code Review help.
- **Log in** (only displayed if you are not logged in to P4 Code Review): click to log in to P4 Code Review, see [Log in to P4 Code Review](#).
- **User id** dropdown menu (only displayed if you are logged in to P4 Code Review):
 - **My Profile:** click for your activity, shelves, profile settings, and notification settings. For more information about your profile, see ["Viewing your user profile" on page 344](#).
 - **System Information** (*admin* and *super* users only): click to view the P4 Code Review system information page. For details about the information on this page, see ["System Information" on page 712](#).
 - **About P4 Code Review:** click to view P4 Code Review version information.

If you are a user with *super* privileges, P4 Code Review also displays the trigger token required when you configure P4 Server for P4 Code Review

 - **Log Out:** log out of P4 Code Review, see ["Log out of P4 Code Review" on page 384](#).

Main menu

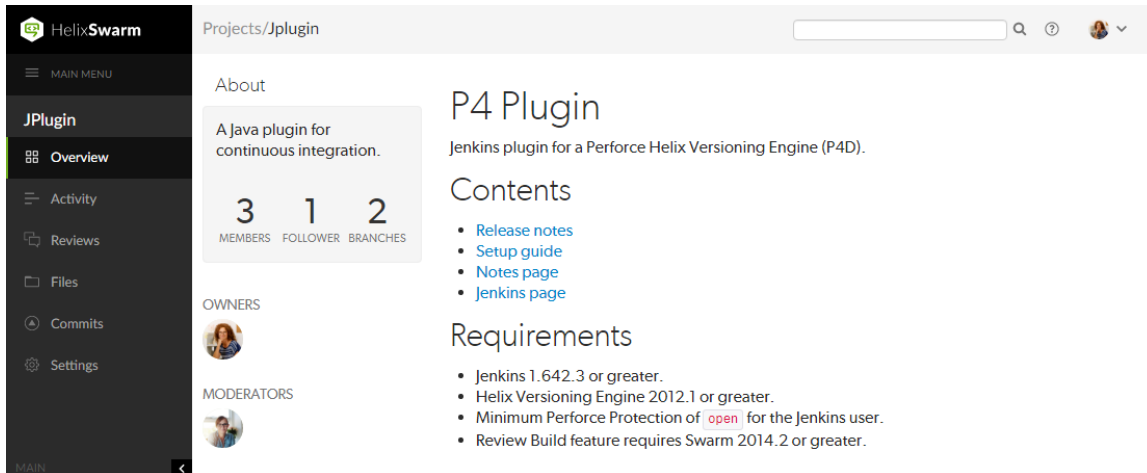


- **My Dashboard:** click to display a list reviews that need your attention, see [My Dashboard](#).
- **Activity:** click to display a list of events that have occurred recently in the associated P4 Server, see [Activity](#).
- **Reviews:** click to display a list of open and closed reviews. For more information about the reviews list page, see ["Reviews list" on page 406](#).
- **Projects:** click to display a list of P4 Code Review projects. For more information about the projects page, see ["Projects" on page 360](#).
- **Files:** click to display a list of files in the P4 Server, starting at the top level. For more information about the files page, see ["Files" on page 303](#).
- **Commits:** click to display a list of commits made in the P4 Server. For more information about the commits page, see ["Commits" on page 313](#).
- **Jobs** (only displayed if Perforce jobs are configured): click to display a list of Perforce jobs in the P4 Server. For more information about the jobs page, see ["Jobs" on page 315](#).
- **Groups** (displayed by default but can be hidden by administrator, see ["Groups configuration" on page 616](#).): click to display the P4 Server groups in P4 Code Review, see ["Groups" on page 471](#).
- **Workflows** (not displayed if the Workflow feature is disabled): click to display a list of P4 Code Review workflows. For more information about the workflows page, see ["Workflows" on page 367](#).
- **Tests** (not displayed if the Workflow feature is disabled): click to display a list of P4 Code Review tests. For more information about the tests page, see ["Tests" on page 373](#).
- **<** (only displayed if the main menu is expanded): click to collapse the menu.
- **>** (only displayed if the main menu is collapsed): click to expand the menu.
- **P4 Server instance name** (only displayed if P4 Code Review is connected to multiple P4 Servers): the instance name is displayed in the bottom left of the menu.

Project menu


1. Click the **Projects** link in the P4 Code Review main menu.
2. Select a project from the [Projects page](#).

The project overview page is displayed:



- Click on the P4 Code Review logo to see a list reviews that need your attention, see the [My Dashboard](#) page.



- **MAIN MENU** click to display the "[Main menu](#)" on the previous page.
- **Overview** (only displayed if there is a README markdown file in the project's mainline, see "[Markdown in projects](#)" on page 398): click to display an overview of the project. For more information about the project overview page, see "[Overview page](#)" on page 361.
- **Activity**: click to display the activity stream for the project. For more information about the project activity page, see "[Activity page](#)" on page 362.
- **Reviews**: click to display a list of open and closed code reviews for the project. For more information about the project reviews page, see "[Reviews page](#)" on page 363.
- **Files**: click to display a list of files for the project. The project Files page shows a list of files for the project, starting with a folder view representing each branch. Branches are designated with the *branch* icon . For more information about the files page, see "[Files page](#)" on page 364.
- **Commits**: click to display a list of changes made to the project. For more information about the commits page, see "[Commits page](#)" on page 364.
- **Jobs** (only displayed if Perforce jobs are configured): click to display a list of jobs associated with the project. For more information about the jobs page, see "[Jobs page](#)" on page 365.
- **Settings** (only displayed if you have permission to view or edit the project): click to display the project settings. For instructions on how to change the project settings, see "[Project settings](#)" on page 478.
- **<** (only displayed if the project menu is expanded): click to collapse the menu.

- > (only displayed if the project menu is collapsed): click to expand the menu.
- **P4 Server instance name** (only displayed if P4 Code Review is connected to multiple P4 Servers): the instance name is displayed in the bottom left of the menu.

Quick URLs quick reference

P4 Code Review handles URLs intelligently to reduce the amount of information you need to enter to quickly locate what you are looking for:

1. Enter a URL in the following format:
`https://myswarm.url/<identifier>`
2. P4 Code Review checks the `<identifier>` and redirects you to the first match it finds. P4 Code Review checks for the identifier in following order:
 - [Reviews](#)
 - [Changelists](#)
 - [Depot paths](#)
 - [Projects](#)
 - [Jobs](#)
 - [Users](#)
 - [Groups](#)
 - Depot names

Note:

- Changelist and review identifiers must be numeric.
- If you enter an identifier that does not exist for any type of resource, P4 Code Review displays a Page Not Found error.

Example usage

Display the latest version of a review

For example, to display review 124: enter the URL `https://myswarm.url/124`, P4 Code Review redirects to `https://myswarm.url/reviews/124`.

Tip:

- To display a specific version of a review, enter the full URL including reviews and append `/v<n>/` to the URL. For example, to display version 2 of review 124: enter `https://myswarm.url/reviews/124/v2/`
- To display a diff between two versions of a review, enter the full URL including reviews and append `/v<n,n>/` the URL. For example, to display the diff between version 2 and 4 of review 124: enter `https://myswarm.url/reviews/124/v2,4/`

- To open a review with a specific tab open, append `#<tab name>` to the end of the URL (`#<tab name>` must be the last item in the URL):
 - **Files tab:** `#files` (default if `#<tab name>` is not specified)
 - **Comments tab:** `#comments`
 - **Activity tab:** `#activity`

Display a changelist

For example, to display changelist 123: enter the URL `https://myswarm.url/123`, P4 Code Review redirects to `https://myswarm.url/changes/123`.

Tip:

To open a changelist with a specific tab open, append `#<tab name>` to the end of the URL:

- **Files tab:** `#files` (default if `#<tab name>` is not specified)
- **Comments tab:** `#comments`

Display the latest version of a file

Enter the URL `https://myswarm.url/depot/alpha/readme.txt`, P4 Code Review redirects to `https://myswarm.url/files/depot/alpha/readme.txt`.

Tip:

- To display a specific version of a file, enter the full URL including files and append `?v=<n>` to the URL. For example, to display version 2 of the `depot/alpha/readme.txt` file: enter `https://myswarm.url/files/depot/alpha/readme.txt?v=2`
- To open a file with a specific tab open, append `#<tab name>` to the end of the URL (`#<tab name>` must be the last item in the URL):
 - **View tab:** `#view` (default if `#<tab name>` is not specified)
 - **Commits tab:** `#commits`

Display the user profile page for jsmith

Enter the URL `https://myswarm.url/jsmith`, P4 Code Review redirects to `https://myswarm.url/users/jsmith`.

My Dashboard page quick reference

The purpose of the dashboard is to allow you to focus on reviews that need to be done, so that other users are not blocked. The dashboard lists the most recently modified reviews first and shows your role in the review. The dashboard displays a maximum of 25 reviews. Perform an action on the current list of reviews to see if any more reviews are available.

Note:

Since it is tied to the logged in user, **My Dashboard** is only populated if you are logged in.

1. Log in to P4 Code Review.
2. If the **My Dashboard** page is not already displayed, click **My Dashboard** in the menu or click the P4 Code Review icon above the menu.

My Dashboard

Project

▼

Role

▼

Author

▼













There are new changes to your review list


🔄

Clear all

🔍

Search for a review

Description	Your Role	Last Activity	State	Tests	Complexity	Votes
<div><div></div><div><div>Updated README to have a bit more information.</div><div>Jack Boone requested a  pre-commit review 365 for mercury:dev</div></div></div>	Reviewer	44 minutes ago	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div> 137	<div><div></div><div>0</div></div> <div><div></div><div>0</div></div>
<div><div></div><div><div>Updated notes for the new year.</div><div>Jack Boone requested a  post-commit review 422 for jplugin:main</div></div></div>	Required reviewer	2 months ago	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div> 18	<div><div></div><div>0</div></div> <div><div></div><div>0</div></div>
<div><div></div><div><div>Add comment to method.</div><div>Steve Russell requested a  post-commit review 328 for jplugin:main</div></div></div>	Required reviewer	3 months ago	<div><div></div><div></div></div>	<div><div></div><div></div></div>	<div><div></div><div></div></div> 6	<div><div></div><div>2</div></div> <div><div></div><div>0</div></div>



All available reviews are displayed

Jump directly to a specific area of the dashboard page using the following links:

- ["Filtering" below](#)
- ["Review summaries" on the facing page](#)

Filtering

Project	Role	Author	There are new changes to your review list		Clear all	Search for a review	
---------	------	--------	---	--	-----------	---------------------	--

The filters are used to filter your dashboard reviews:

- **Project:** filter by the project the review is part of, limiting results to **My Projects** (projects you are an owner of or a member of) or specific projects. To select multiple projects, click the projects you want to filter by. The **Project** filter will only show projects for which there are reviews in your dashboard.
- **Role:** filter by your specific role in a review, limiting results to reviews for which you are the author, a reviewer, a required reviewer, or a moderator. The **Role** filter will only show roles for which there are reviews in your dashboard.
- **Author:** filter the reviews to only those that have been authored by a certain user. Select the author from the drop down list of users. The **Author** filter will only show authors for which there are reviews in your dashboard.
- **Clear all:** click to reset all dashboard filters back to their defaults.
- **Search:** filter the reviews by searching the review descriptions.

Refresh button

Refresh button (only displayed when new content is available for your dashboard): click **Refresh** to update your dashboard with the new content.


Note:

By default, when your dashboard is open, P4 Code Review checks for new content every five minutes.

Review summaries




The dashboard displays a summary for each review.

Description	Your Role	Last Activity	State	Tests	Complexity	Votes
 Updated README to have a bit more information. <small>Jack Boone requested a  pre-commit review 365 for mercury:dev</small>	Reviewer	about an hour ago			 137	 2  0

- **Avatar** displays the avatar of the review author, click to view the profile of the author, see ["Viewing another user's profile" on page 350](#).
- **Description** contains:
 - first line of the review description, click to open the review. If the review description is too long it is truncated, click on the ellipsis  to expand it in the list page.
 - review author, click to view the profile of the author.
 - review type, either a [pre-commit review](#) or a [post-commit review](#).
 - review number, click to open the review, see ["Review display" on page 411](#).
 - review project branches, click to open the project.
- **Your role:** displays your role in the review and is the reason the review is in your dashboard. This can be Author, Reviewer, Required Reviewer, or Moderator.
- **Last activity:** displays the last time that any changes were made to the review, including votes, comments, commits, and file changes.
- **State:** a review can be in one of the following states:
 - **Needs review:** The review has started and the changes need to be reviewed.
 - **Needs revision:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
 - **Approved:** The review has been approved. The changes may need to be committed.

Note:

The Approved state only applies to review authors, it is only shown for a review that has been approved but has not been committed.

- **Tests:** shows the test suite state for the review, either tests in progress , tests passed , or tests failed .
- **Complexity:** a traffic light icon and number shows the relative complexity of the review and the total number of lines changed in the review. [Complexity can be configured](#) by your P4 Code Review administrator but, by default:

- **Red:** ≥ 300 changes
- **Amber:** < 300 and > 30 changes
- **Green:** ≤ 30 changes

Tip:

- Review complexity is only calculated for a review when the review is updated and the file content has changed.
- Review complexity is only stored for the current version of a review.

Hover over the complexity icon to display more detailed information:



- **Votes:** displays the number of up votes and down votes for the review.

Global dashboard page quick reference

Note:

Global dashboard is only available if your P4 Code Review administrator has configured P4 Code Review to connect to more than one P4 Server instance.

The purpose of the global dashboard is to allow you to focus on reviews that need to be done, so that other users are not blocked. The global dashboard lists reviews by P4 Server according to the most recently modified first, and shows your role in the review.

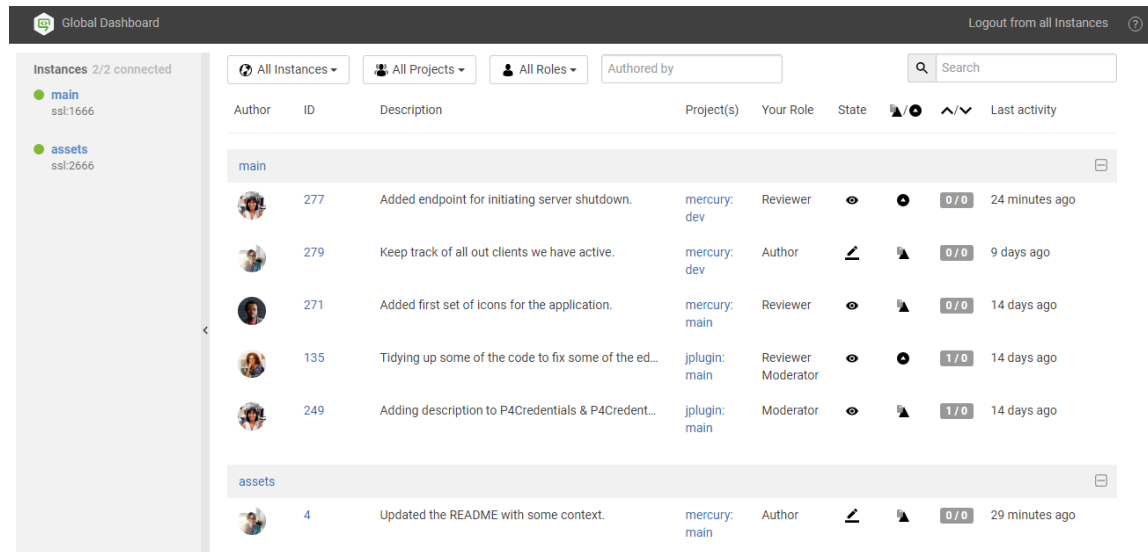
1. Enter the basic P4 Code Review URL without a P4 Server instance name, for example: `https://swarm.company.com`
2. If the log in dialog is displayed it is the same dialog that is displayed when you click **Log In** from the banner, log in to the global dashboard.

For instructions about logging in to the global dashboard, see "[Log in to one or more P4 Server instances](#)" on page 293.

Tip:

- Since it is tied to the logged in user, the global dashboard is only populated for the P4 Server instances you are logged in to in P4 Code Review.
- If you are already logged in to a P4 Server instance in P4 Code Review, the dashboard for that server will be automatically populated when you open or refresh the global dashboard.

The global dashboard is displayed:



The global dashboard is made up of three main areas:

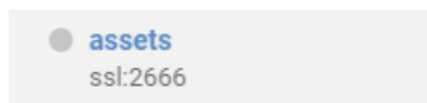
- **"Toolbar" below:** log in to P4 Server instances, log out of all P4 Server instances, and open the P4 Code Review help from the global dashboard toolbar.
- **"Sidebar" below:** Log in/logout of individual P4 Server instances and view your profile for any instance that you are logged in to.
- **"P4 Server dashboards" on the next page:** view, filter and search the dashboards of the P4 Server instances you are logged in to. Jump directly to a P4 Server, review, or project by clicking on a link.

Toolbar

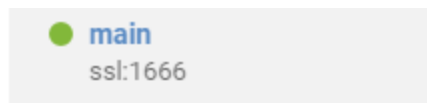
- **Log In** (only displayed in the toolbar if you are not logged in to any P4 Server instances): click to log in to one or more P4 Server instances, see ["Log in to one or more P4 Server instances" on page 293](#).
- **Logout from all instances** (only displayed in the toolbar if you are logged in to at least one P4 Server instance): click to logout from all of the P4 Server instances, see ["Logout of all P4 Server instances" on page 295](#).
- **Help?**: click to view the P4 Code Review help.

Sidebar




- **Log in status:**
 - Not logged in to the P4 Server instance:



- Logged in to the P4 Server instance:























- P4 Server instance actions:

- **Log in**  (only available if you are not logged in to the P4 Server instance): hover over the instance name and click the **Log in** button  when it is displayed, see ["Log in to a P4 Server instance" on page 296](#).
- **Logout** (only available if you are logged in to the P4 Server instance): hover over the instance name, click the **Gear** button , and select **Logout** from the dropdown menu to logout of the instance, see ["Logout of a single P4 Server instance" on page 298](#).

P4 Server dashboards

The global dashboard lists reviews by P4 Server according to the most recently modified first, and shows your role in the review.

Example global dashboard showing a number of P4 Server dashboards:

<div> All Instances ▾ All Projects ▾ All Roles ▾ Authored by Search </div>									
Author	ID	Description	Project(s)	Your Role	State				Last activity
main									
	277	Added endpoint for initiating server shutdown.	mercury: dev	Reviewer				0 / 0	24 minutes ago
	279	Keep track of all out clients we have active.	mercury: dev	Author				0 / 0	9 days ago
	271	Added first set of icons for the application.	mercury: main	Reviewer				0 / 0	14 days ago
	135	Tidying up some of the code to fix some of the ed...	jplugin: main	Reviewer Moderator				1 / 0	14 days ago
	249	Adding description to P4Credentials & P4Credent...	jplugin: main	Moderator				1 / 0	14 days ago
assets									
	4	Updated the README with some context.	mercury: main	Author				0 / 0	29 minutes ago

Jump directly to a specific area of the dashboard page using the following links:

- ["Dashboard filters" below](#)
- ["Dashboard content" on the facing page](#)

Dashboard filters

All Instances ▾	All Projects ▾	All Roles ▾	Authored by	Search
------------------------------	-----------------------------	--------------------------	--------------------------	---------------------

Used to filter your dashboard reviews, the following filters are available:





- **Instances:** filter by the P4 Server instance, limiting results to **All Instances** or to an individual instance.
- **Projects:** filter by the project the review is part of, limiting results to **All Projects** or to an individual project. The **Project** filter will only show projects for which there are reviews in your dashboard.
- **Roles:** filter by your specific role in a review, limiting results to reviews for which you are the author, a reviewer, a required reviewer, or a moderator. The **Role** filter will only show roles for which there are reviews in your dashboard.
- **Authored by:** filter the reviews to only those that have been authored by a certain user. Type in this field to get a drop down list of users to filter by.
- **Reset** (only displayed if one or more filters are set): click the **Reset** button to reset all dashboard filters back to their defaults.
- **Search:** filter the reviews by searching the description and review ID.

Dashboard content

The dashboard for each P4 Server shows a summary of the information for each review.

Tip:

- Click on the P4 Server name in the header bar to open P4 Code Review for that instance in a new tab.
- Click on the header bar to the right of the P4 Server name to collapse the dashboard for that instance. Click again to expand the dashboard for the instance .

Author	ID	Description	Project(s)	Your Role	State	 	Last activity
main 							
	277	Added endpoint for initiating server shutdown.	mercury: dev	Reviewer	 	0 / 0	1 hour ago

- **Author**avatar: the review author's avatar, hover over the avatar to see the ID and name of the review author. Click on the avatar to go to the profile of the review author, see "[Viewing another user's profile](#)" on page 350.
- **ID:** the unique number used to identify the review. Click the review ID to display the review, see "[Review display](#)" on page 411.
- **Description:** the review description may be truncated if it is too long, in which case click on the review description to expand it.
- **Project(s):** lists the project branches this review covers. A review may span multiple branches and projects. Click a branch link to navigate to the project page for that branch.
- **Your role:** displays your role in the review and is the reason the review is in your dashboard. This can be Author, Reviewer, Required Reviewer, or Moderator.
- **State:** a review can be in one of the following states:

- **Needs review:** The review has started and the changes need to be reviewed.
- **Needs revision:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
- **Approved:** The review has been approved. The changes may need to be committed.

Note:

The Approved state only applies to review authors, it is only shown for a review that has been approved but has not been committed.

- **Type:** displays the type of review, either a [pre-commit review](#) or a [post-commit review](#).
- **Votes up/down:** displays the number of up votes and down votes for the review.
- **Last activity:** displays the last time that any changes were made to the review, including votes, comments, commits, and file changes.

Reviews list page quick reference

The **Reviews** page lists code reviews that are in progress, and code reviews that are complete.

1. Log in to P4 Code Review.
2. Click **Reviews** in the main menu.

Reviews						
Opened 19 Closed		Search for a review				
Project	Role	Reviewers	States	Tests	Comments	Votes
Description	Created	State	Tests	Complexity	Comments	Votes
Added Windows to list of build platforms. Allison Clayborne requested a post-commit review 396, 31 minutes ago for jplugin:main	31 minutes ago	Open	0	137	1	0 0 0
Tidying up some of the code base. Allison Clayborne requested a pre-commit review 391, 3 days ago for jplugin:main	3 days ago	Open	0	18	1	0 0 0
Added some state exception checking. Jack Boone requested a pre-commit review 388, 4 days ago for jplugin:main	4 days ago	Open	0	6	2	0 0 2
Update to the 12th entry point, which is defined to measure the latency Jack Boone requested a pre-commit review 385, 4 days ago for maker:master	4 days ago	Open	0	1462	0	1 0 0
Update the README to use markdown rather than the old format. Jack Boone requested a pre-commit review 382, 4 days ago for blue-book:main	4 days ago	Open	0	2	0	0 0 0

Jump directly to a specific area of the **Reviews** page using the following links:

- ["Opened and Closed tabs" below](#)
- ["Review filters" on the facing page](#)
- ["Reviews list content" on page 38](#)

Opened and Closed tabs

The **Opened** and **Closed** tabs display:

- **Opened** tab: displays a list of all code reviews that have started, are being reviewed, are awaiting revisions, or need to be committed.
- **Closed** tab: displays a list of all code reviews that have been approved and committed, rejected, or archived.

Tip:

- The **Opened** and **Closed** tabs display the number of reviews in each tab. Initially this can be an over estimate which P4 Code Review dynamically corrects as more information becomes available to it.
- The **Closed** tab initially displays a - character. The - is replaced by a number when you navigate to the **Closed** tab.

Review filters

Tip:

P4 Code Review updates the URL in your browser to reflect the filter options you have selected. This makes it easy to bookmark or share your review list filter settings with the URL. If you share a URL with **Comment** or **Vote** filters selected, those filters are applied to the user running the URL. This means that their reviews list page will contain different reviews to your reviews list page.



The following filters are available (from left to right):

- **Branch** (only available on the project "[Reviews page](#)" on page 363): a dropdown menu that lets you filter which reviews to display based on the current project branches:

Branch 

- **Select branch:** all reviews for all of the selected branches are displayed. To select multiple branches, click the branches you want to filter by.
 - **Branch search:** an auto-complete search field that allows you to choose one of the branches within the current project. Once specified, only reviews for the selected project branch are displayed.
- **Project** dropdown: filter by the project the review is part of:

Project 

- **My projects:** displays all reviews for all of the projects you are participating in, as a member, owner, moderator, or follower.
 - **Select project:** displays all reviews for all of the selected projects. To select multiple projects, click the projects you want to filter by.
 - **Project search:** an auto-complete search field that allows you to choose one of the projects defined in P4 Server. Once specified, only reviews for the selected project are displayed.

Tip:

P4 Code Review creates a projectid based on the project name whenever a project is created. For example, a project called **Test Data** is given a projectid of test-data. If the project name changes, the projectid does not change.

If your search finds a match in a projectid but not in the associated project name, the project name for that projectid is returned in the search results. For example, if the project name was changed from **Test Data** to **Sample Data**, a search for *test* will return the **Sample Data** project in the search results.

- **Role** dropdown: filter reviews based on user involvement, options are:

Role 

- **All reviews:** displays all reviews.
- **Reviews I've authored:** displays reviews that you have authored.
- **Reviews I'm participating in:** displays reviews that you are a reviewer of, but not an author of.
- **Reviews I'm an individual reviewer of:** displays reviews that you are an individual reviewer of, but not a group reviewer of, or an author of.
- **Review I've authored or I'm participating In:** displays reviews that you have authored, or are a reviewer of.
- **Authored by:** an auto-complete search field that allows you to choose one of the user accounts defined in the P4 Server. Once specified, only reviews authored by the user are displayed.

When you select one of the available options, the list of options updates to match the currently selected filter, and the **Role** dropdown indicates the current filter: **All**, **Author**, **Participant**, **Author or Participant**, **Individual Reviewer**, or **user**.

- **Reviewers** buttons, (**Opened** tab only):



- **Has reviewers:** displays reviews that have one or more reviewers.
- **No reviewers:** displays reviews that have no reviewers.

- **States** buttons, (**Opened** tab):

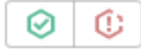


- **Needs Review:** displays reviews that need to be reviewed.
- **Needs Revision:** displays reviews that have been reviewed, but need further revisions before the review can be accepted.
- **Approved:** displays reviews that have been approved, and should be committed.

- **States** buttons, (**Closed** tab):



- **Approved:** displays reviews that have been approved and committed.
- **Rejected:** displays reviews that have been rejected.
- **Archived:** displays reviews that have been archived.
- **Tests** buttons (when automated tests are enabled for the associated project):



- **Tests passed** displays reviews that have passed tests.
 - **Tests failed:** displays reviews that have failed tests.
- **Comments** buttons:



- **I have commented on:** displays reviews that you have commented on.
 - **I have not commented on:** displays reviews that you are participating in but have not commented on

Note:

Filters for commenting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after commenting on it, then this review will not be included in the list.

- **Votes** buttons:



- **I have voted up:** displays reviews that you have voted up.
 - **I have voted down:** displays reviews that you have voted down.
 - **I have not voted on:** displays reviews that you are participating in but have not voted on.
































Note:


Filters for voting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after voting on it, then this review will not be included in the list.

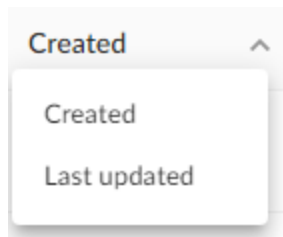
- **Clear all:** click to clear all of the filters.
- **Search for a review:** search for a partial match of review description content and an exact match for *reviewid*, *userid*, and *projectid*. The search filter boxes on the **Opened** and **Closed** tabs are independent.

Reviews list content

The review list displays a summary for each review.

Description	Created	State	Tests	Complexity	Comments	Votes
 Added Windows to list of build platforms. Allison Clayborne requested a  post-commit review 396, 31 minutes ago for jplugin:main	31 minutes ago			 137	 1	 0  0
 Tidying up some of the code base. Allison Clayborne requested a  pre-commit review 391, 3 days ago for jplugin:main	3 days ago			 18	 1	 0  0
 Added some state exception checking. Jack Boone requested a  pre-commit review 388, 4 days ago for jplugin:main	4 days ago			 6	 2	 0  2
 Update to the 12th entry point, which is defined to measure the latency Jack Boone requested a  pre-commit review 385, 4 days ago for maker:master	4 days ago			 1462	 0	 1  0

- **Avatar** displays the avatar of the review author, click to view the profile of the author
- **Description** contains:
 - first line of the review description, click to open the review. If the review description is too long it is truncated, click on the ellipsis  to expand it in the list page.
 - review author, click to view the profile of the author
 - review type, pre-commit or post-commit
 - review number and version, click to open the review
 - review creation date and time
 - review project branches, click to open the project
- **Created** or **Last activity** dropdown: click to change the order the reviews are displayed in, options are:






- **Created:** Reviews sorted by when they were created
- **Last activity:** Reviews sorted by when they were last updated

Note:

- If the **Result order** button is not displayed, reviews are sorted by either **Created** or **Activity**, as set by your P4 Code Review administrator, see ["Reviews filter" on page 673](#) for details.
- **Result order** button display is a global setting controlled by the P4 Code Review administrator. See ["Reviews filter" on page 673](#) for details.
- When review results are older than 24 hours they are displayed in numerical order within each day.

- **State:** a review can be in one of the following states:

- **Needs review:** The review has started and the changes need to be reviewed.
- **Needs revision:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
- **Approved:** The review has been approved. The changes may need to be committed.
- **Tests:** displays the test suite state for the review, either tests in progress , tests passed , or tests failed .
- **Complexity:** a traffic light icon and number shows the relative complexity of the review and the total number of lines changed in the review. By default, complexity icon displays:
 - **Red:** ≥ 300 changes
 - **Amber:** < 300 and > 30 changes
 - **Green:** ≤ 30 changes

Tip:

- Review complexity is only calculated for a review when the review is updated and the file content has changed.
- Review complexity is only stored for the current version of a review.

Hover over the complexity icon to display more detailed information:



- **Comments:** displays the number of open (non-archived) comments that are associated with the review. The icon is filled if there are comments on the review.
- **Votes:** displays the number of up votes and down votes for the review. The appropriate vote icon is filled if you have voted on the review, vote icons with only an outline show votes by others.

Review page quick reference

Tip:


The review page is described in this documentation.

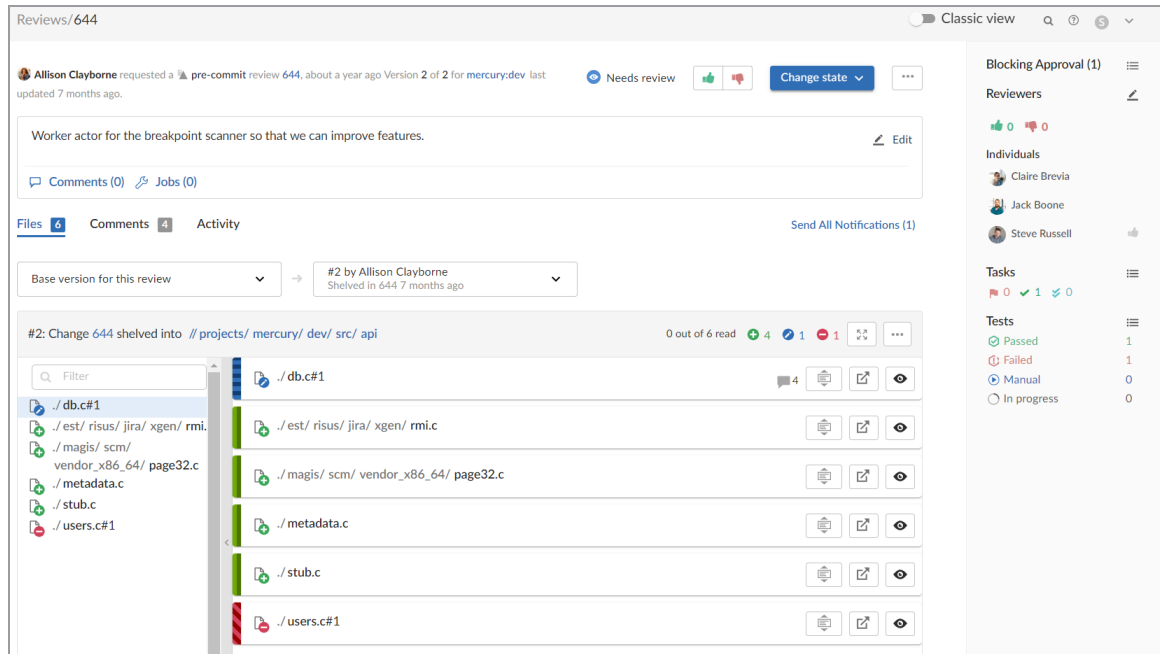
The old review page is now referred to as the classic review page. To use the classic review page, use the **Classic view** toggle switch at the top of the review page.

The P4 Code Review Classic view option will be removed in a future P4 Code Review release.

Help for the classic review page is available in the P4 Code Review 2021.2 documentation. See [Review display](#).

The review page is used when reviewing changes:

1. Log in to P4 Code Review.
2. To display the P4 Code Review home page, click the P4 Code Review icon above the menu.

3. If your dashboard is not already displayed, click the **Dashboard** tab.
4. To open a review, click the **ID** of the review.



By default, when you open the Review page for the first time, the file list panel and the information panel are hidden. Use the **Show more context** buttons to view the file list panel and the information panel or use the keyboard shortcuts, **Shift + Alt + L** to view the file list and **Shift + Alt + S** to view the Information panel.

For more information about the show more context buttons, see ["File diff panel" on page 52](#).

Jump directly to a specific area of the review page using the following links:


- ["Review description" on the facing page](#)
- ["Information panel" on page 43](#)
- ["Files tab" on page 48](#)
 - ["File diff panel" on page 52](#)
- ["Comments tab" on page 57](#)
 - ["Add a comment" on page 59](#)
- ["Activity tab" on page 60](#)

Review description


The screenshot shows a review page for review 644. At the top, it says "Reviews/644". Below that, a header bar contains the review details: "Allison Clayborne requested a pre-commit review 644, 10 months ago Version 2 of 2 for mercury:dev last updated about a minute ago." To the right of this bar are icons for "Needs review", thumbs up/down, a "Change state" dropdown, and a menu icon. The main content area has a title "Worker actor for the breakpoint scanner so that we can improve features." with an "Edit" link. Below the title are links for "Comments (1)" and "Jobs (0)". A section titled "1 archived comment" contains a comment from Jack Boone: "This description needs more detail." with a green checkmark icon, an eye icon, and a menu icon. At the bottom of the comment section is an "Add a comment" button.

The review description is made up of the following elements:

- **Review description header:**
 - **Review information:** Displays the following details:
 - **Review author** avatar and name: The avatar and name of the review author. Hover over the avatar to see the name of the review author and hover over the name to see the ID of the review author. Click on the avatar or name to go to the profile of the review author. See "[Viewing another user's profile](#)" on page 350.
 - **Review type:** Displays the review type, reviews can be [pre-commit](#) or [post-commit](#).
 - **Review ID:** The unique number used to identify the review.
 - **Review raised:** When the review was requested.
 - **Version:** The version of the review being viewed and the total number of review versions.
 - **Project branch:** The project branch the review files are in, click to go to the project page.
 - **Last updated:** When the review was last updated.
 - **Review state** icon: a review can be in one of the following states:
 - **Needs review:** The review has started and the changes need to be reviewed.
 - **Needs revision:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
 - **Approved:** The review has been approved. The changes may need to be committed.
 - **Rejected:** The review has been completed. The changes are undesirable and should not be committed.
 - **Archived:** The review has been completed for now but it is not rejected, or approved. The review has been filed away in case it is needed in the future.
 - **Voting buttons:** click a voting button to vote the review up or down.

- **Change state** button: Click to select a new state from the review dropdown menu. State change options are only displayed if you are authorized to make the state change:
 - **Needs revision:** Select to request changes to the files in the review.
 - **Needs review:** Select to request further review of the changes.
 - **Approve** (only available if the voting requirements for the review are satisfied. For information on voting requirements, see ["Required reviewers" on page 457](#)): select to approve the review.
 - **Commit** (only available for [pre-commit](#) reviews that have been approved): Select to commit the review.
 - **Approve and commit** (only available for unapproved [pre-commit](#) reviews when the voting requirements for the review are satisfied. For information on voting requirements, see ["Required reviewers" on page 457](#)): Select to approve and commit the review in a single step. See ["Approve and commit" on page 469](#).
 - **Reject:** Select to reject the review.
 - **Archive:** Select to archive the review.
- **Review actions**  button: click the **Review actions** button to:
 - **Add change:** Select to add a changelist to the review. Options available depend on whether the review is [pre-commit](#) or [post-commit](#). For information about adding a changelist to a review, see ["Add change" on page 415](#).
 - **Download zip** (if configured): Select to download a compressed archive of all of the files in the review, see ["Download files as a ZIP archive" on page 70](#).
 - **Change author** (if configured): Select to change the author of the review, see ["Change the review author" on page 418](#).
 - **Obliterate review** (by default, only available for users with *admin* or *super* user rights): See ["Obliterate Review" on page 657](#).
 - **Join review** (only if you are not a member of the review): Select to join the review. If you vote on a review, you automatically join the review and become a reviewer.
 - **Leave review** (only if you are a member of the review): Select to leave the review.
 - **Mark all comments read:** Click to mark all of the comments on this review as read.
 - **Mark all comments unread:** Click to mark all of the comments on this review as unread.
 - **Refresh projects:** Select to check if the review files are associated with any projects created or updated after the review was last updated. If any projects are found that are not already associated with the review, the review is linked to them. See ["Refresh projects" on page 420](#).
 - **Disable notifications** (only supported in the classic review page): Select to disable notifications for this review.

- **Try it out** (only supported in the classic review page and if configured): If your deployment system provides a URL to the deployment, you can select it to see the deployment results.
- **Review description:** The review description is automatically copied from the changelist that was originally used to create the review.
 When the review description size exceeds the 25% of the page, only the initial part of the review description is displayed. Select the **Expand description** button to view the full review description and select the **Collapse description** button to hide a part of the review description.

 To change the review description, click the  **Edit** button. Markdown content is displayed in the review description, but Markdown support is limited to prevent execution of raw HTML and JavaScript content. For information about Markdown, see ["Markdown in comments and review descriptions" on page 397](#)
 - **Update pending changelist** checkbox in the edit description dialog (only available if you are editing the description of a pre-commit review and you are the original author of the changelist that created the review): Select the checkbox to also apply your review description changes to the original changelist description.
- **Comments (n):** Click to add a comment to the review description, view existing description comments, or hide existing description comments. Click outside an empty comment text box or select **Esc** on your keyboard to close it. See ["Commenting on a review description" on page 329](#).
- **Jobs (n)** (if configured): Perforce jobs can be linked to the review. Click to add a Perforce job to the review, view jobs linked to the review, unlink a job from the review, or hide jobs on the review. For more information on Perforce jobs, see ["Jobs" on page 315](#).
 - **Add a job:** Click to add a job to the review. Select the job from the dialog that is displayed.
 - **jobnnnnnn** (where **nnnnnn** is the job number): Click the job number to view details of the Perforce job. For more information about Perforce jobs, see ["Job display" on page 316](#).
 - **X** button: To unlink a job from the review, click the **X** button of the job you are unlinking from the review.
- **Send All Notifications (n)** (where **n** is the number of delayed notifications): Click to manually send all of your delayed notifications for the review. For more information about comment notification delay, see ["Comment notification delay" on page 336](#).

Information panel

By default, when you open the Review page for the first time, the file list panel and the information panel are hidden. Use the **Show more context** buttons to view the file list panel and the information panel or use the keyboard shortcuts, **Shift + Alt + L** to view the file list and **Shift + Alt + S** to view the Information panel.

For more information about the show more context buttons, see ["File diff panel" on page 52](#).

The information panel on the right side of the review page contains the following elements:

- **Blocking Approval or Blocking Commit:**

Once a review has been approved, the **Blocking Approval** section is not displayed.

For pre-commit reviews, if a user adds a new requirement to an approved review, then that requirement is displayed in the **Blocking Commit** section. For example, if a project initially has the **Minimum up votes** requirement set to one and this requirement is satisfied, and the review is in approved state. Now, if the **Minimum up votes** requirement changes to two then this requirement is displayed in the **Blocking Commit** section. Only when this requirement is satisfied the user can do a commit.

- **Pending Comment Tasks:** Provides a list of pending review comments that are flagged as *tasks*. The pending comment tasks are shown only when the `disable_approve_when_tasks_open` configurable is set to true in the `SWARM_ROOT/data/config.php` file.

Note:

If P4 Code Review is configured to prevent approval of reviews with open tasks and a review has open tasks, the **Approve**, and **Approve and commit** options will not be available for the review. This option is configured by an administrator. See ["Disable approve for reviews with open tasks" on page 676](#).

To approve, or approve and commit a review with open tasks, you must address the tasks first and then set them to **Task Addressed**, or **Not a Task**. See ["Set a task to Task addressed or Not a task" on page 333](#) for details.

For more information about approving a review with pending tasks, see ["Approve a review with open tasks" on page 335](#).

- **Required Reviewers:** Provides a list of all the Individual and group reviewers that are required to vote on the review. A review can not be approved or committed until the required reviewers have voted up on the review.

A required ["Review page quick reference" on page 39](#) has a star badge over their avatar. A required ["Group reviewer" on page 425](#) either has a star badge or a star badge with a 1 over their avatar.



- **Blocking Tests:** Provides a list of workflow tests that have **Failed** or have an **In progress** status that is blocking approval for a review. A review can not be approved or committed until the review author fixes all the failed tests. For more information about a failed test, see ["Review page quick reference" on page 39](#).
- **Minimum Up votes:** Provides the number of **Minimum Up votes** required for each project that you have access to. This includes the maximum number of **Minimum Up votes** required for all branches in a project blocking approval or commit for a review. For more information about **Minimum up votes** for a project see, [General Settings tab](#). For more information about **Minimum up votes** for a project branch, see [Branches tab](#).





For [private project](#), the maximum number of **Minimum Up votes** for all private projects and their branches that is blocking approval for a review is displayed.


- **Moderator Votes:** Provides a list of branch moderators that are blocking approval or commit for a review. Depending on the configuration of `moderator_approval` configurable in the `SWARM_ROOT/data/config.php` file, the branch moderator approval is required. Ensure that the branch moderator approves the review and not just up votes it.

By default, when a review spans multiple branches that have different moderators, only one moderator from any one of the branches needs to approve the review.











P4 Code Review can be configured to require that one moderator from each branch must approve the review, this is a global setting. If a moderator belongs to more than one of the branches spanned by the review, their approval will count for each of the branches they belong to. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

- **Reviewers:**
 - **Edit**  button (if enabled): click to edit the reviewers for the review
 - **Up** vote and **Down** vote count: indicates the number of up votes and down votes the review has.
 - **Groups**: lists groups that are reviewers for the review. When at least one person in the group has voted, the avatar displays a badge indicating whether the group, as a whole, has voted up or down. Click on the group to see who has voted, and how they have voted, see ["Group reviewer" on page 425](#).
 - **Individuals**: lists individuals that are reviewers for the review. When an individual has voted on the review, their avatar displays a badge indicating whether they voted up or down. For more information about individual reviewers, see ["Individual reviewer" on page 426](#).
- **Tasks**
 - **Task Summary** icons: summary of the number and status of comments flagged as tasks for the review. For more information about tasks, see ["Tasks" on page 332](#).
 - **Red Flag** icon: indicates the numbers of open tasks on the review.
 - **Green Check Mark** icon: indicates the numbers of tasks that have been addressed on the review.
 - **Blue Double-Check Mark** icon: indicates the number of tasks that have been addressed and verified on the review.
 - **Show Task details**  button: click to display the **Tasks** dialog listing all tasks associated with the review. The tasks can be filtered by reporter and task type, and you can change the task state directly from the **Task** dialog. See ["Task details" on page 334](#).
- **Tests** (if configured): displays the test results for the current review version in the information panel

-  **Passed:** Displays the number of the tests that have passed for the review version.
-  **Failed:** Displays the number of the tests that have failed for the review version.
-  **Manual:** Displays the number of **On Demand** tests that can be run manually for the review version.
-  **In Progress:** Displays the number of tests that are running for the review version.









Show test details  button: Click to view test run information for tests associated with the review version:


Tests

Status	Title		
 Passed	main-codeSniff		Rerun test 
 Passed	main-EsLint		Rerun test 
  Failed	main-Jest		Rerun test 

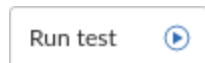
Messages

Error contacting server

 Passed	main-jsLint		Rerun test 
 Manual	main-library		Run test 
 In progress	main-modules		

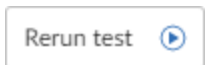
If your continuous integration system calls back to P4 Code Review with a URL, the test run is linked to the URL provided. If the test result contains any messages, click the **Show more**  button to the left of the test to view them

When the content of a review is unchanged between review versions, the tests are not rerun because the files have not changed. In this case, the earlier test results are displayed and marked with the word **(Copy)**. You can manually run these tests if required by using the



Run test button.

The following actions are available from the **Tests** dialog for tests called by a workflow:

- **Rerun test:** If a test has completed you can rerun the test, click  to rerun the test for the most recent version of the review.

- **Run test:** If the review workflow has a test set to run **On Demand**, click to manually run the test for the most recent version of the review.



Tip:

- The **Run test** and **Rerun test** buttons are only available if you are logged in to P4 Code Review, viewing the latest version of the review, and the test was called by a workflow.
- The **Rerun test** button is not available for tests that are in progress.
- When you click **Run test** or **Rerun test**, all of the buttons in the list are temporarily disabled while P4 Code Review starts the test and checks if any other tests are in progress.
- The **Rerun test** and **Run test** buttons are only available for tests that are run by a workflow. They are not available for tests that are run because they are configured on a project.

Note:

Private projects: if a test for a private project is added to your review because **Iterate tests for affected projects and branches** is selected for the test, P4 Code Review honors the private project's permissions and displays it as **Private project** in the test list to users that do not have permission to view it.

- For information about iterating tests, see "[Iterate tests for affected projects and branches checkbox](#)" on page 530.
- For information about private projects, see "[Private projects](#)" on page 366.

- < (only displayed if the information panel is expanded): click to collapse the panel.
- > (only displayed if the information panel is collapsed): click to expand the panel.

Files tab

Use the **Files** tab to view the files in the review and to see how they have changed using the P4 Code Review diff view.

Files 6 Comments 4 Activity

Send All Notifications (1)

Base version for this review

#2 by Allison Clayborne
Shelved in 644 7 months ago

#2: Change 644 shelved into // projects/ mercury/ dev/ src/ api

0 out of 6 read

+ 4 1 - 1

Filter

./db.c#1

./est/ risus/ jira/ xgen/ rmi.c

./magis/ scm/

vendor_x86_64/ page32.c

./metadata.c

./stub.c

./users.c#1

./db.c#1

42 lines hidden above

```

43  * perferendis consequuntur qui quidem. Et et aut es
44  t ex.
45  */
46  void * initVoluptatem() {
47      int loop = 1;
48
49  // Harum quaeat in distinctio vitae adipisci impe
50  dit distinctio.
51  printf("Dicta voluptas mollitia est placeat au
52  t.\n");
53  et = tempora + 1;
54  voluptas = atoi(qui)?quas:"eos";
55  runOccaecatI(ut, sed);
56  }
57  }
58  }
59  }
60  }

```

6 lines hidden

42 lines hidden above

```

43  * perferendis consequuntur qui quidem. Et et aut es
44  t ex.
45  */
46  void * initVoluptatem() {
47      int loop = 1;
48
49  // Abbatia hodie conubia litora nunc.
50  dolor = flag / 1;
51  printf("Femina sem bis non dolor.");
52  printf("Wisi, mando gladius.");
53
54  // Harum quaeat in distinctio vitae adipisci impe
55  dit distinctio.
56  printf("Dicta voluptas mollitia est placeat au
57  t.\n");
58  et = tempora + 1;
59  voluptas = atoi(qui)?quas:"eos";
60  runOccaecatI(ut, sed);
61  }
62  }
63  }
64  }
65  }

```



6 lines hidden

P4 Code Review supports stream specs in your workspace using the [Private editing of streams](#) feature. If a changelist or review contains a stream spec, it will be displayed first in **Files** with the prefix **stream: //**, for example: `stream://MyStreamDepotName/MyStreamSpecLocationName`. A changelist/review can only contain one stream spec.

The **Files** tab is made up of the following elements (from left to right):

- **Review version selectors:** Select which version of the review you want to diff. For details on using the version selectors, see ["Select review versions to view" on page 431](#).
- **File listing header:**
 - **Review version and common file path summary:**
 - **Comparing the files in the latest version of the review to Base or Head:** Displays the current revisions of the files in the review, the changelist that the review is based on, and the common path for the review files.
 - **Comparing the files in two versions of a review:** Displays which two versions of the review are being compared, which changelists the files are in, and the common path for the files in both versions of the review.
 - **Total number of files:** Indicate the number of files that are marked as read out of the total number of files in a review.
 - **File change summary icons:** Indicate the number of files that have been added, edited, and deleted in this version of the review.

+ 4 1 - 1

- **Expand the Diff view to full screen:** Select the  button to expand the diffs view to full screen. Once the diffs view is expanded to full screen, select the button a second time to exit the full screen mode or use the keyboard shortcut **Shift + Alt + F**.
- **Diff actions**  button: select the **Diff actions** button to:
 - **Expand or collapse all files** (expands or collapses all the files within a review if one or more files are in the collapsed or expanded state)
 - **Expand all files** (only displayed if all or at least one file is collapsed) Select to expand all the files within a review.
 - **Collapse all files** (only displayed if all files are expanded) Select to collapse all the files within a review.

Use keyboard shortcut **Shift + Alt + E** to expand or collapse all files in a review.

- **Show or hide all in-line comments for all files in a review** (Shows the total number of comments across all files in a review)
 - **Hide in-line comments** (only displayed if inline comments are expanded): Select to collapse all of the inline file comments.
 - **Show in-line comments** (only displayed if inline comments are collapsed): Select to expand all of the inline file comments.

Use keyboard shortcut **Alt + C** to hide or show in-line file comments.

- **Show diffs in-line or side-by-side** (only works for text files)

When a comment is added to a diff line or an unchanged line in the inline diff view, the in-line comment will span the entire file diff panel. When a comment is added to a diff line in the side-by-side view, the comment will only appear on the side where it was made. If a comment is added to an unchanged line, it will span the entire file diff panel.

 - **Show diffs in-line:** Select to display the diffs in inline format.
 - **Show diffs side-by-side:** Select to display the diffs in side-by-side format. The oldest file revision is shown in the left pane, and the newer one is shown in the right pane.

Use keyboard shortcut **Alt + L** to toggle viewing the diffs in-line or side-by-side for a text file.

- **Show or hide whitespace and tab characters for all files in a review**
 - **Show whitespace and tab characters** (display of CRLF line endings is only supported in the classic review page): Shows whitespace and tab characters for all text files. The tab characters are padded to show the correct alignment of tabs.
 - **Hide whitespace and tab characters:** Hides whitespace and tab characters for all text files.

Use keyboard shortcut **Alt + W** to show or hide whitespace and tab characters for all files in a review.

- **Show or hide whitespace diffs**

- **Show whitespace diffs:** Whitespace changes are highlighted, makes it easier to identify changes in file types where whitespace is important.
- **Hide whitespace diffs:** Whitespace changes are not highlighted, this makes it easier to see the important changes in file types where whitespace changes are not important.

Use keyboard shortcut **Alt + D** to show or hide whitespace diffs for all files in a review.

- **Disable syntax highlighter**

P4 Code Review displays the code in the default syntax colors for that coding language. However, this feature can be disabled to display the code without syntax highlighting. For more information on the supported languages, see ["Supported syntax highlighting in the Review page" on page 311](#).

- **Navigate to the next file**

Select this option to open the next file in the Diff view or use keyboard shortcut **Alt+N**.

- **Navigate to the previous file**

Select this option to open the previous file in the Diff view or use keyboard shortcut **Alt+P**.

- **Show the shortcut help dialog**

Select this option to open the shortcut help dialog to view all the file control keyboard shortcuts or use keyboard shortcut **Alt+H**.

- **File list:**

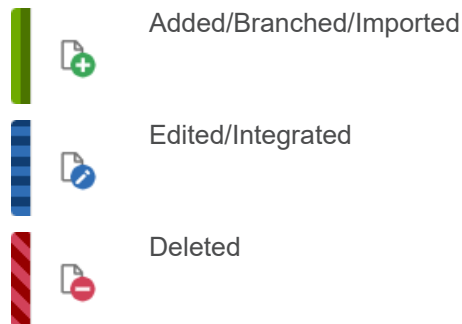
Important:

- For a review containing large number of files, using **Ctrl+F** to search for a file does not work. You can use the file filter to search for a specific file in the review. To manually select a file from the file stack, scroll through the Review page.
- By default, when you open the Review page for the first time, the file list panel and the information panel are hidden. Use the **Show more context** buttons to view the file list panel and the information panel or use the keyboard shortcuts, **Shift + Alt + L** to view the file list and **Shift + Alt + S** to view the Information panel.

For more information about the show more context buttons, see ["File diff panel" on the next page](#).

- **File filter:** Filter by file name, file extension, or folder name to find the files you are interested in. You can see the diff view for the matched search result in the File stack section.

- **File list resize:** Change the file list width by dragging the split bar left or right. Alternatively, click the split bar to collapse or expand the file list. If you change the file list width for a review, Swarm saves the width locally for that review. The saved width is used next time you view the review.
- **View complete file list:** Use the scroll bar to view the complete list of files.
- **File stack:**
 - **File change type** icons: Each file is marked with a ribbon and an icon indicating whether the file was:



- **File content panel** header: The file content panel header contains the following elements from left to right:



- **File name:** The name and revision of the file you are currently viewing.
- **Total number of file comments:** The total number of comments for a file. The comments count includes all inline comments and file-level comments. It does not include the archived comments.
- **Show Full Context** button (only displayed for text file diffs): Click to toggle between displaying only the portions of the file that have changed and the full file.
- **Open in a new tab** button: Click to open the file in a new tab or use keyboard shortcut **Alt + T**.
- **Mark file as read** button: Click to mark the file as read, click again to mark the file as unread or use keyboard shortcut **Alt + R**.

File diff panel

By default, all files larger than 1 MB in size are truncated. When a file is truncated a message is displayed. Use the ["max_size" on page 613](#) configurable in the `SWARM_ROOT/data/config.php` file to increase or decrease this limit.

If you rename a file, edit it, and request a review, only the edited content is highlighted in the diff panel.

Use AI for code analysis

Click the AI button  to get a summary explaining the diff. You can send the whole diff or just a selection for code analysis.



For more information, see ["AI for code analysis" on page 547](#).

Important: The AI code analysis feature is disabled by default. Use the `ai_review.enabled` configurable in the `SWARM_ROOT/data/config.php` file to enable AI code analysis.

For more information on how to enable AI code analysis, see ["Configuration overview" on page 585](#).

To discard an AI summary, select the **Discard summary** button.

■ Text file diff:

When you view a diff, the changes are highlighted:


- Red indicates lines that have been removed.
- Blue indicates lines that have been modified.
- Green indicates lines that have been added.

For more information on the supported extensions for syntax highlighting in the Review page, see ["Supported syntax highlighting in the Review page" on page 311](#).





When a comment is added to a diff line or an unchanged line in the inline diff view, the in-line comment will span the entire file diff panel.

When a comment is added to a diff line in the side-by-side view, the comment will only appear on the side where it was made. If a comment is added to an unchanged line, it will span the entire file diff panel.

Example inline diff view:

 ./db.c#1

4



42 lines hidden above

43

43

* perferendis consequuntur qui quidem. Et et aut est ex.

44

44

*/

45

45

void * initVoluptatem() {

46

46

int loop = 1;

47

47

48

48

// Abbatia hodie conubia litora nunc.

49

49

dolor = flag / 1;

50

50

printf("Femina sem bis non dolor.");




51

51

printf("Wisi, mando gladius.");



Steve Russell


commented on review 644 (version 1) - about a year ago



@allison.clayborne

Is this really necessary to spam the console with this?



 Add a comment

52

52

48

53

// Harum quaerat in distinctio vitae adipisci impedit distinctio.

49

54

printf("Dicta voluptas mollitia est placeat aut.\n");

50

55

et = tempora + 1;

51

56

voluptas = atoi(qui)?quas:"eos";

52

57

runOccaecati(ut, sed);

34 lines hidden

87

92

popFugiat(aperiam, quo);

88

93

addBeataeDevice(iste, ut);

89

94

}

90

95

91

96

97

97

/**

98

98

* Herbam class tortor justo, augue justo congue. Volutpat a magnis

99

99

* cursus pupula. Aliquet diam peristo ullamcorper cultura. Suspendisse

100

100

* nostra.

101

101

*/

Example side-by-side diff view:

./db.c#1

4

42 lines hidden above

43

* perferendis consequuntur qui quidem. Et et aut es

44

*/

45

void * initVoluptatem() {

46

int loop = 1;

47

48

// Harum quaerat in distinctio vitae adipisci impe

49

dit distinctio.

50

printf("Dicta voluptas mollitia est placeat au

51

t.\n");

52

et = tempora + 1;

53

voluptas = atoi(qui)?quas:"eos";

54

runOccaecati(ut, sed);

34 lines hidden

87

popFugiat(aperiam, quo);

88

addBeataeDevice(iste, ut);

89

}

90

91

42 lines hidden above

43

* perferendis consequuntur qui quidem. Et et aut es

44

*/

45

void * initVoluptatem() {

46

int loop = 1;

47

48

// Abbatia hodie conubia litora nunc.

49

dolor = flag / 1;

50

printf("Femina sem bis non dolor.");

51

printf("Wisi, mando gladius.");

52

// Harum quaerat in distinctio vitae adipisci impe

53

dit distinctio.

54

printf("Dicta voluptas mollitia est placeat au

55

t.\n");

56

et = tempora + 1;

57

voluptas = atoi(qui)?quas:"eos";

58

runOccaecati(ut, sed);

34 lines hidden

92

popFugiat(aperiam, quo);

93

addBeataeDevice(iste, ut);

94

}

95

96

Steve Russell

commented on review 644

(version 1) - about a year ago

@allison.clayborne Is this really necessary to spam the console with this?

Add a comment

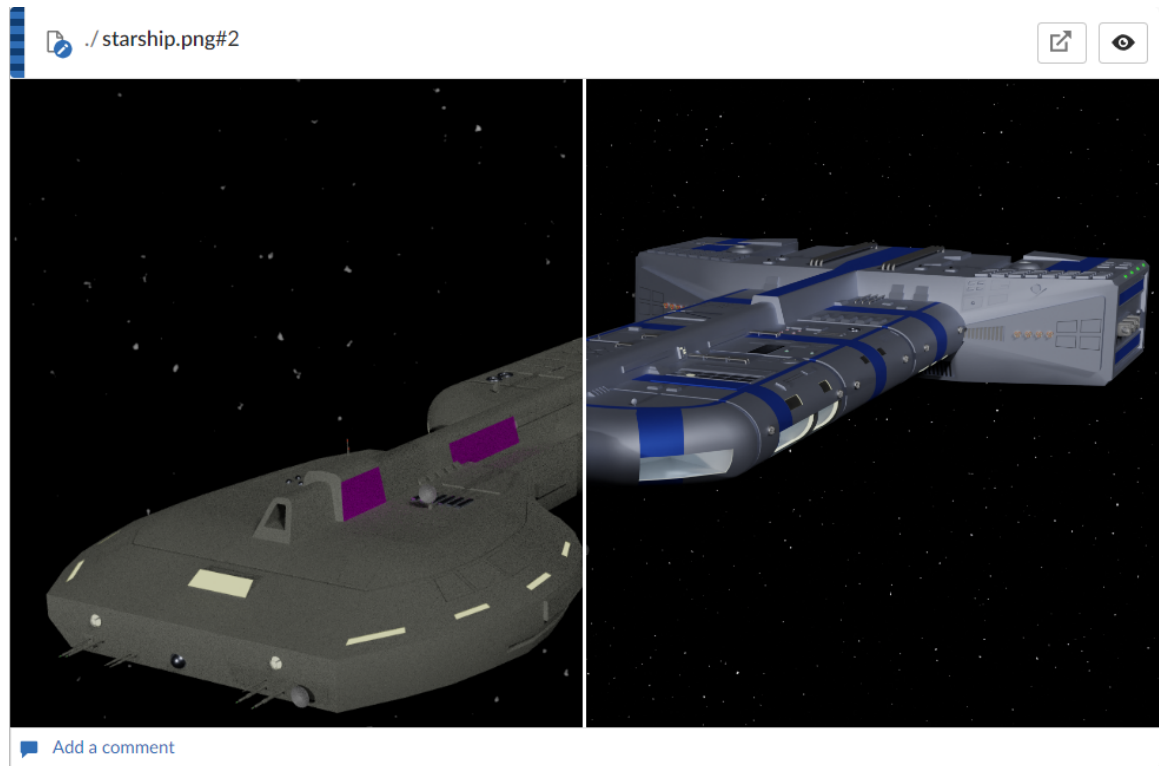
Image file diff:

Image types that are natively supported by your browser are displayed with a vertical slider on the image.

Drag the slider to the left to reveal more of the newer image and to the right to reveal more of the earlier one. Alternatively, click on the image to move the slider.

Example image diff:

55










- **Show more context buttons:**

Sometimes, the concise diff view needs to be expanded to fully understand the context of the change, use the **Show More** and **Show All** buttons to display extra lines around the change:

Tip:

The following buttons are not available for stream specs.

- **Show the file list panel**  button (only displayed on the vertical bar next to the file list panel): click to view the file list panel.
- **Show the information panel**  button (only displayed at the bottom of the information panel): click to view the detailed information panel.
- **Show All Lines to Start of File**  button (only displayed for the first change in the file): click to show all of the lines up to the start of the file.
- **Show More Lines for the Code Below**  button: click to show 10 more lines above the change, the extra lines are displayed in the pane below the button.
- **Show Entire Section**  button: click to show all of the lines between the changes that are above and below the button, the two changes and the lines between them are displayed in a single pane.
- **Show More Lines for the Code Above**  button: click to show 10 more lines below the change, the extra lines are displayed in the pane above the button.
- **Show All Lines to End of File**  button (only displayed for the last change in the file): click to show all of the lines down to the end of the file.

Comments tab

The **Comments** tab is used to view all of the comments in the review.

Files **6** Comments **4** Activity Send All Notifications (0)

5 archived comments

Steve Russell commented on review 644 (version 1) //projects/mercury/dev/src/api/db.c, line 51 - 11 months ago ✓ read ✓

Allison Clayborne commented on review 644 (version 1) //projects/mercury/dev/src/api/db.c, line 105 - 10 months ago 👁️ ...

```
+ */
+int TalisAliquam(long quandoScelerisqueSem, float count, double litora) {
+ // Peristo velit parco.
+ parco = volutpat / 1;
+ printf("Aquam arcu vehicula faucibus umbra magna.");
```

@steve.russell what do you think of this?

👍

Steve Russell commented 4 months ago 👁️ ...

This looks good to me. I am happy for it to be checked in.

👍

Allison Clayborne commented 4 months ago 👁️ ...

Thanks for the confirmation.

👍

Add a comment











Drag and drop files to attach them or [Choose your files](#)

☐ Flag as Task ☐ Post and notify (0)

Comments are made up of the following elements:

- **Send all notifications (n)** (only supported in the classic review page): Comment notifications are delayed by default, click to manually send the notification immediately for the review. For more information about comment notification delay, see ["Comment notification delay" on page 336](#).
- **Comment detail:**
 - **Commenter** avatar and name: The avatar and name of the user that made the comment. Hover over the avatar to see their name and hover over their name to see their ID. Click on the avatar or name to see their user profile. For information about user profiles, see ["Viewing another user's profile" on page 350](#).
 - **Review ID:** The unique number used to identify the review.
 - **Version** link (includes the file and line number if the comment is on a line in a file): Click to go to the review version the comment was made on. If the comment is made inline in a file the link will take you to that line in the file.
 - **Filepath** link (only if the comment is made on a file): Link to the file the comment is made on, includes line number if the comment was made on line in the file.
 - **Comment raised:** When the comment was made.
- **Tasks:** Flagging review comments as tasks is a lightweight workflow within a review that helps authors and reviewers prioritize review feedback. Any comment on a review can be flagged as a task, indicating to the review's author that the described issue needs to be addressed, and that the review is unlikely to be approved without a fix. For information about working with

tasks, see ["Tasks" on page 332](#).

- **No flag displayed:** The comment has not been flagged a task. To flag the comment as a task, click the **Comment actions**  button and select **Flag as Open task**.
- **Flagged as a Task**  button: the comment has been flagged as a task. Click to confirm that the task has been addressed, or to remove the task flag from the comment.
- **Task Addressed**  button: the comment task has been addressed. Click to verify that the task has been addressed correctly, to reopen the task if it has not been addressed correctly, or to verify and archive the task (only supported in the classic review page).
- **Task Verified**  button: the comment task has been addressed and has been verified as correct. Click to reopen the task you think it has not been addressed correctly.
- **Mark comment as read**  button (unread comments only): Click to mark a comment as read, the comment will only be marked as read for you. For more information on marking a comment as read or unread, see ["Mark comments as read" on page 341](#).
- **Restore**  button (archived root level comments only): Click to restore an archived comment. For more information about archiving and restoring comments, see ["Archiving comments" on page 343](#) and ["Restoring comments" on page 343](#).
- **Comment actions**  button:
 - **Flag as Open task:** Select flag a comment as a task. For information about working with tasks, see ["Tasks" on page 332](#).
 - **Archive:** (only available for root level comments): Select to archive a comment and any replies to that comment. For more information about archiving and restoring comments, see ["Archiving comments" on page 343](#) and ["Restoring comments" on page 343](#).
- **Comment context** (only if the comment is made inline in a file): Displays several lines of code before the line of code the comment is attached to. This helps makes sense of the comments should later changes remove those lines.
- **Comment content:** This can be text, a URL, or an attachment (only supported in the classic review page). For more information about the content of comments, see ["Comment features" on page 336](#).
- **Reply**  button: Click to reply to the comment. For information about replying to comments, see ["Reply to comments" on page 337](#).
- **Edit**  button (only available for comments that you have made): Click to edit the comment. For information about editing comments, see ["Editing comments" on page 331](#).
- **Thumbs Up**  button: Click to like a comment. For information about liking comments, see ["Reacting to comments" on page 340](#).

Add a comment

To add a comment, type your comment in the open comment text box at the bottom of the **Comments** tab page.

To reply to a comment, click the **Reply**  button and type your comment in the comment text box.

- **Comment** text box: type your comment in the text box.
- **Flag as a Task** checkbox: Select to flag the comment as a task. For more information about tasks, see ["Flag a comment as a task" on page 332](#).
- **Post** button: Click to post your comment. The comment is posted immediately but the comment notification is delayed, see ["Comment notification delay" on page 336](#).
- **Post and notify (n)** link: Click to post your comment and send the comment notification immediately.
Where **(n)** is the number of delayed comment notifications in the queue waiting to be sent, this number does not include the current comment you are working on.

Tip: Use the keyboard shortcut **Ctrl + Enter** to submit the comment or form.

To close an empty comment text box, click outside the comment text box or select **Esc** on your keyboard.

Activity tab

The **Activity** tab presents a list of the events on this review.

Files **6**
Comments **4**
Activity
Send All Notifications (1)

Allison Clayborne cleared an issue on [review 644](#) for [mercury:dev](#) 6 minutes ago

This needs another review please

Allison Clayborne archived comment on [review 644](#) for [mercury:dev](#) 8 minutes ago

This description needs more detail.

Allison Clayborne replied to a comment on [review 644](#) for [mercury:dev](#) 8 minutes ago

Thanks for the confirmation.

Steve Russell replied to a comment on [review 644](#) for [mercury:dev](#) 9 minutes ago

This looks good to me. I am happy for it to be checked in.

Events in the activity tab include:

- When the review was started
- When a new reviewer joins the review
- When the review's state changes
- When the review's files are updated
- When a reviewer votes on the review
- When someone comments on the review, or one of its files
- When tests pass or fail, provided continuous integration is configured

P4 Code Review user tasks

This section describes a number of the common P4 Code Review user tasks:

In this section:

Start a code review

Important:

If your P4 Server is configured as a commit-edge deployment, and your normal connection is to an edge server, P4 Code Review refuses to start reviews for shelved changes that have not been promoted to the commit server.

Within P4 Code Review, this means that the **Request Review** button does not appear for unpromoted shelved changes. Outside of P4 Code Review, attempts to start reviews for unpromoted shelved changelists appear to do nothing. Ask your P4 Server administrator for assistance if you cannot start a review.

An administrator of the P4 Server can automatically promote shelved changes to the commit server by setting the configurable `dm.shelve.promote` to 1.

Tip:

If your changelist only contains a stream spec and its location in the P4 Server is not associated with a P4 Code Review project, the review that you create will not have any default reviewers or workflow rules. To associate the review with a project so that it has default reviewers and obeys the project workflow rules, include a file change from the project path in the changelist when you create the review.

1. Use the P4 command-line (P4) or a client to create the shelved or committed changelist.
2. Start a code review by using one of the following approaches:

Use P4 Code Review:

1. Use P4 Code Review to view a shelved or submitted changelist.

Tip:

To view a shelved or submitted changelist, use a [Quick URL](#). For example, if your change is **54321**, visit the URL: <https://myswarm.url/54321>.

2. Click the **Request Review** button to request a review of that changelist.

Requesting a review on a shelved changelist uses the [pre-commit](#) model and requesting a review on a submitted changelist uses the [post-commit](#) model.

Important:

The **Request Review** button is disabled for changelists with a file count greater than the limit set for `max_changelist_files` in the [SWARM_ROOT/data/config.php](#) file. This is to prevent out of memory errors within P4 Code Review. For more information on changing this file count limit, see ["Changelist files limit" on page 573](#)

Use P4 command-line (P4):

When you are about to shelve or submit files:

1. Include `#review` within your changelist (separated from other text with whitespace, or on a separate line).

Once the review begins, P4 Code Review replaces `#review` with `#review-12345`, where 12345 is the review's identifier.

Note:

The `#review` keyword is customizable. For details, see ["Review keyword" on page 668](#).

2. At this time, you can add reviewers to the code review by using `@mention` for users, and `@@mention` for groups in the changelist description for each desired reviewer.

If your `@mention` or `@@mention` includes an asterisk (*) before the *userid* or *groupid*, for example `@*userid`, that user or all of the group members become required reviewers. If your `@@mention` includes an exclamation mark (!) before the *groupid*, for example `@@!groupid`, the members of that group become required reviewers but only one member of the group is required to vote. See ["Required reviewers" on page 457](#) for details.

3. Complete your shelve or submit operation.

Warning:

If you shelve a changelist and subsequently edit the description to include `#review`, a review is **not started**. You must re-shelve the files after adding `#review`.

Tip:

You can also start a review with P4V, P4 for Visual Studio, and P4 for Eclipse. See below for details:

- **P4V:** see the [P4 Code Review integration features](#) section of the [P4 Visual Client \(P4V\) Documentation](#).
- **P4VS:** see the [Managing files](#) chapter of the [P4VS User Guide](#).
- **P4Eclipse:** see the [Reviewing changes](#) chapter of the [P4Eclipse User Guide](#).


Note:

If you are using P4V and its P4 Code Review integration, and you encounter the error `Host requires authentication`, ask your P4 Server administrator for assistance. See ["P4V Authentication" on page 584](#) for details.

Once a review has started

Wait for someone else to review your code, or see ["Contribute comments or code changes to a code review" on page 66](#) More [review activities](#) are available.

Edit the reviewers on a review

A review author can edit the reviewers for a review by using the Reviewers **Edit**  button in the review page **Information panel**. Reviewers are able to join or leave reviews, or to change whether their vote is required or optional.

Note:


If the review includes content that is part of a project or branch with [Retain default reviewers](#) enabled, the following restrictions apply to the review:

- The voting option for a retained default reviewer can only be changed to a stricter level, you cannot reduce the voting level.
- Retained default reviewers cannot be removed from the review.

Additionally, the following individuals can edit reviewers:

- Users with *admin* or *super* privileges.
- If the review is [moderated](#), the moderators.
- If the review is part of a project, but not moderated, all project members.
- If the review is not part of a project, any authenticated user.

To edit reviewers on a review:

1. Go to the review.
2. From the review **Information panel**, click the **Reviewers** list **Edit**  button.
The **Manage reviewers** dialog is displayed.
3. Click the **Users** tab to edit a user or the **Groups** tab edit a group. This field auto-suggests users, and groups within P4 Server as you type (up to a combined limit of 25 entries).
4. **Change the voting requirements of a user or group:**
 - a. Click the dropdown arrow to the right of the user or group you want to change.
 - b. Select one of the following:
 - **Users:** select whether their vote is **Required** or **Optional**. A solid star means that their vote is required to approve a review, whereas the outlined star means that their vote is optional.
 - **Groups:** select whether the group vote is **Require all**, **Require one**, or **Optional**. A solid star means that all group member votes are required to approve a review, a solid star with a 1 inside means at least one group member must vote up and no group members vote down to approve a review, and the outlined star means that the group vote is optional.
5. **Add a user or group to a review:**

- a. Click the **Add user** or **Add group** dropdown icon to display a list of users or groups:
 - b. Start typing the user or group name, the list is filtered as you type.
 - c. Click on the user or group to add to the review.
 - d. Click **+** to add them as a reviewer.
 - e. Set the reviewer type from the dropdown to the right of the user or group.
6. **Remove a user or group from a review:**
 - a. Click the dropdown icon to the right of the user or group to be removed.
 - b. Select **Remove from the review** from the dropdown menu.
 7. To close the **Manage reviewers** dialog, click **X**.

Check for code reviews you need to act on

The purpose of the dashboard is to allow you to focus on reviews that need to be done, so that other users are not blocked. The dashboard lists the most recently modified reviews first and shows your role in the review. The dashboard displays a maximum of 25 reviews. Perform an action on the current list of reviews to see if any more reviews are available.

Your dashboard is available by clicking **My Dashboard** in the menu or by clicking the P4 Code Review icon above the main menu.



Tip: My Dashboard is only populated if you are logged in.

1. Log in to P4 Code Review.
2. If the **My Dashboard** page is not already displayed, click **My Dashboard** in the menu or click the P4 Code Review icon above the menu.

My Dashboard

Project

▼

Role

▼




Author

▼

There are new changes to your review list

Clear all

Search for a review

Description	Your Role	Last Activity	State	Tests	Complexity	Votes
<div><div></div><div><div>Updated README to have a bit more information.</div><div>Jack Boone requested a <div><div></div></div> pre-commit review 365 for mercury:dev</div></div></div>	Reviewer	44 minutes ago	<div><div></div></div>	<div><div></div></div>	<div><div></div>137</div>	<div><div><div></div></div>0<div><div></div></div>0<div><div></div></div>0</div>
<div><div></div><div><div>Updated notes for the new year.</div><div>Jack Boone requested a <div><div></div></div> post-commit review 422 for jplugin:main</div></div></div>	Required reviewer	2 months ago	<div><div></div></div>	<div><div></div></div>	<div><div></div>18</div>	<div><div><div></div></div>0<div><div></div></div>0<div><div></div></div>0</div>
<div><div></div><div><div>Add comment to method.</div><div>Steve Russell requested a <div><div></div></div> post-commit review 328 for jplugin:main</div></div></div>	Required reviewer	3 months ago	<div><div></div></div>	<div><div></div></div>	<div><div></div>6</div>	<div><div><div></div></div>2<div><div></div></div>0<div><div></div></div>0</div>

All available reviews are displayed

3. By default all projects with reviews you need to act on are displayed.
4. To filter the reviews displayed by project, select **My Project** or specific projects from the **Project** filter. To select multiple projects, click the projects you want to filter by. The **Project** filter will only show projects for which there are reviews in your dashboard.
5. By default all reviews you need to act on are displayed.

- To filter by your role in a review, select **Author**, **Reviewer**, **Required reviewer**, or **Moderator** from the **Role** filter. The **Role** filter will only show roles for which there are reviews in your dashboard.
- By default all authors with reviews you need to act on are displayed.
- To filter by review author, select **All Authors** or a specific author from the **Author** filter. The **Author** filter will only show authors for which there are reviews in your dashboard.
- The **Refresh** button is displayed when new content is available for your dashboard. Click **Refresh** to update your dashboard with the new content.

Note:

By default, when your dashboard is open, P4 Code Review checks for new content every five minutes.

Tip:

- For more information on the dashboard page, see ["My Dashboard" on page 288](#).
- Click on a review **ID** number to display the [code review](#).

List reviews you are involved with

- Click **Reviews** in the main menu.

The screenshot shows the HelixSwarm interface with the 'Reviews' section active. The left sidebar contains a menu with options: My Dashboard, Activity, Reviews (selected), Projects, Files, Commits, Jobs, Groups, Workflows, and Tests. The main content area is titled 'Reviews' and includes tabs for 'Opened' (19) and 'Closed'. Below the tabs are filters for Project, Role, Reviewers, States, Tests, Comments, and Votes. A search bar is also present. The table below lists several reviews with their descriptions, creation times, states, and other metrics.

Description	Created	State	Tests	Complexity	Comments	Votes
Added Windows to list of build platforms. Allison Clayborne requested a post-commit review 396, 31 minutes ago for jpluginmain	31 minutes ago	Open	0	137	1	0
Tidying up some of the code base. Allison Clayborne requested a pre-commit review 391, 3 days ago for jpluginmain	3 days ago	Open	0	18	1	0
Added some state exception checking. Jack Boone requested a pre-commit review 388, 4 days ago for jpluginmain	4 days ago	Open	0	6	2	2
Update to the 12th entry point, which is defined to measure the latency Jack Boone requested a pre-commit review 385, 4 days ago for maker:master	4 days ago	Open	0	1462	0	1
Update the README to use markdown rather than the old format. Jack Boone requested a pre-commit review 382, 4 days ago for blue-book:main	4 days ago	Open	0	2	0	0

- By default all projects are displayed.
- To filter the reviews displayed by project, select **My projects** or specific projects from the **Project** drop down. To select multiple projects, click the projects you want to filter by.
- By default all reviews are displayed.
- To filter by your role in a review, select **Reviews I've authored**, **Reviews I'm participating in**, **Reviews I'm an individual reviewer of**, **Reviews I've authored or I'm participating in**, or search for reviews authored by a specific user from the **Role** drop down.
- Click the **Opened** tab to view open reviews, click the **Closed** tab to view closed reviews. See ["Reviews list" on page 406](#) for more information.

Contribute comments or code changes to a code review

You can contribute to a code review in the following ways:

- Comment on a review
- Comment on a review description
- Comment on a specific line in a file
- Comment on a file in a review
- Reply to a comment
- Edit files in a review

Comments

Comments are the primary feedback mechanism provided by P4 Code Review. Review comments can be flagged as *tasks* for a lightweight workflow within a review that helps authors and reviewers prioritize review feedback. By default, comment notifications are delayed to allow you to add or edit comments as you progress through a review without sending a notification for each individual comment on the review. Comment notifications are rolled up into a single notification that you can either leave to be sent automatically, or you can send manually, see ["Comment notification delay" on page 336](#).

Tip:

For more information about comments and comment features, see ["Comments" on page 328](#).

Comment on a review

1. From the review page: click **Comments** to view the **Comments** tab.
2. Add your comment in the text area.
3. **Optional:** select the **Flag as Task** checkbox to mark the comment as a task that needs to be addressed.
4. Click **Post**.

Comment on a review description

1. From the review page description area: click **Comments (n)** (where **n** is the number of comments that already exist).
2. Click **Add a comment**.
3. Add your comment in the text area.
4. **Optional:** select the **Flag as Task** checkbox to mark the comment as a task that needs to be addressed.
5. Click **Post**.

Tip:

To hide the description comments, click **Comments n** (where **n** is the number of description comments that exist). To display the comments again, click **Comments n**.

Branched the main line to both a new candidate and dev branch.
[Edit](#)

Comments (0)
Jobs (0)

There are no review description comments yet.

All are failing to build on the different branches

Drag and drop files to attach them or [Choose your files](#)

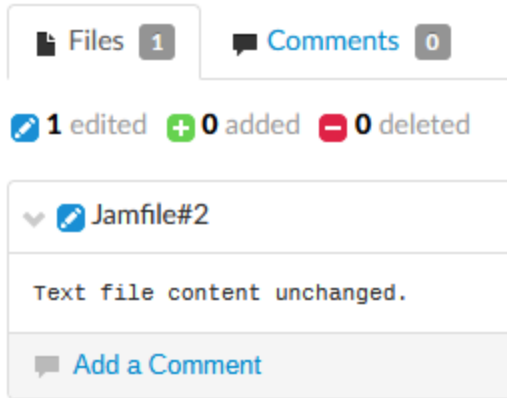
Post
Cancel
☒ Flag as Task
☐ Post and notify (0)

Comment on a specific line in a file


1. From the review page: click **Files** to view the **Files** tab.
2. Click on the line you want to comment on.
3. Add your comment in the text area.
4. **Optional:** select the **Flag as Task** checkbox to mark the comment as a task that needs to be addressed.
5. Click **Post**.

Comment on a file in a review

1. From the review page: click **Files** to view the **Files** tab.
2. If there are multiple files, click the file you want to comment on to expand its view.
3. Click the **Add a Comment** link in the footer of the file display.
4. Add your comment in the text area.
5. **Optional:** select the **Flag as Task** checkbox to mark the comment as a task that needs to be addressed.
6. Click **Post**.



Reply to a comment

1. From the review page: click **Comments** to view the **Comments** tab.
2. Navigate to the comment you are replying to.
3. Click **Reply**  below the comment you are replying to.
4. Add your comment in the text area.
5. **Optional:** select the **Flag as Task** checkbox to mark your reply as a task that needs to be addressed.
6. Click **Post**.

Editing review files

Edit files in a review

1. [Get a local copy](#) of the review's files.
2. Edit the files as required.
3. Prepare a changelist with the edited files and include `#review-1234` within the changelist's description (separated from other text with whitespace, or on a separate line), where `1234` is the review's identifier.

Warning:

If you use an invalid review identifier, it will appear that nothing happens. P4 Code Review is currently unable to notify you of this situation. If the review has not been correctly updated, use the **Add Change** button in the review heading to add the changelist to the review, see ["Add a changelist to a review" on page 450](#).

4. Depending on the [model of code review](#) you are using, you would:
 - Shelve the files (for pre-commit).
 - Submit the files (for post-commit).

Get a local copy of the code in a review for evaluation

P4 Code Review manages one or more changelists containing shelved copies of all of the files belonging to a specific review. You can unshelve the files to receive a copy of the review's code, or you can click the [Download .zip](#) button on the review to download a ZIP archive containing all of the review's files.

Determine the changelist that contains the files in the review

1. Visit the review's page.
2. The current review version's changelist appears in the file list heading.

#3: Change 697707 shelved into //depot/main/swarm

In this example, the changelist is **697707**. You use the identified changelist in place of *shelved changelist* below.

Note:

P4 Code Review can version file updates in reviews. For more information, see ["Review display" on page 411](#).

Using P4

For a shelved changelist, use a command-line shell and type:

```
$ p4 unshelve -s shelved changelist
```

For a committed changelist, use a command-line shell and type:

```
$ p4 sync @committed changelist
```

Note:

Your client's view mappings need to include the changelist's path.

Using P4V

For a shelved changelist:

1. Select **Search > Go To**.
2. Change the select box to **Pending Changelist**.
3. Type in the shelved changelist number and click **OK**.
4. Select the files in the **Shelved Files** area.
5. Right-click and select **Unshelve**.
6. Click **Unshelve**.

For a committed changelist:

1. Select **Search > Go To**.
2. Change the select box to **Submitted Changelist**.
3. Type in the submitted changelist number and click **OK**.
4. Select the files in the **Files** area.
5. Right-click and select **Get this Revision**.
6. Click **Close**.

Download files as a ZIP archive

The **Download zip** option is used to download a ZIP archive that contains all of the files in the review. The file revisions of the downloaded files are the file revisions in the most recent review version selected in the review version selector, see ["Select review versions to view" on page 431](#).

Note:

The **Download zip** option is not displayed if the zip command-line tool is not installed on the P4 Code Review server. For information about installing, and configuring the zip command-line tool, see [Zip archive](#).

When you select the **Download zip** option, P4 Code Review performs the following steps:

1. Scans the files/folders:
 - Checks that you have permission to access their contents, according to the P4 Server protections.
 - Checks that the total file size is small enough to be processed by P4 Code Review.
2. Syncs the file contents to the P4 Code Review server from the P4 Server.
3. Creates the ZIP archive by compressing the file content.
4. Starts a download of the generated ZIP archive.

Note:

- You might not see all of the above steps; P4 Code Review caches the resulting ZIP archives so that repeated requests to the same files/folders can skip the sync and compress steps whenever possible.
- If an error occurs while scanning, syncing, or compressing, P4 Code Review indicates the error.

Fix errors that cannot be merged in a review

The problem can occur when you attempt to **Commit** or **Approve and Commit** via the P4 Code Review UI and the shelved files are out of date.

P4 Code Review cannot currently help with resolving conflicts; you need to use a P4 Server client such as p4 or P4V to resolve conflicts.

Resolve via P4

1. Acquire a [local copy](#) of the files.
2. Sync the files to the head version:

```
$ p4 sync
```

3. Begin resolving files with:

```
$ p4 resolve
```

4. Choose an appropriate option to resolve each file. For example:

```
$ p4 resolve
/home/bruno/bruno_ws/dev/main/jam/command.c - merging
//depot/dev/main/jam/command.c#9
Diff chunks: 4 yours + 2 theirs + 1 both + 1 conflicting
Accept(a) Edit(e) Diff(d) Merge (m) Skip(s) Help(?) e:
```

5. Re-shelve the resolved files with:

```
$ p4 shelve
```

Note:

Ensure that the changelist description contains `#review-12345` (separated from other text by whitespace, or on a separate line), where `12345` is the identifier of the review you are updating.

Warning:

If you use an invalid review identifier, it will appear that nothing happens. P4 Code Review is currently unable to notify you of this situation. If the review has not been correctly updated, use the **Add Change** button in the review heading to add the changelist to the review, see ["Add a changelist to a review" on page 450](#).

For more information, see [p4 resolve](#) in the [P4 CLI Documentation](#).

Resolve via P4V

1. Acquire a [local copy](#) of the files.
2. In P4V, right-click your workspace folder and select **Resolve Files**.
The **Resolve** dialog appears.
3. Choose the appropriate options to resolve each file.
4. Right-click your workspace folder and select **Shelve Files**.

The **Shelve** dialog appears.

Note:

Ensure that the changelist description contains `#review-12345` (separated from other text with whitespace, or on a separate line), where `12345` is the identifier of the review you are updating.

Warning:

If you use an invalid review identifier, it will appear that nothing happens. P4 Code Review is currently unable to notify you of this situation. If the review has not been correctly updated, use the **Add Change** button in the review heading to add the changelist to the review, see ["Add a changelist to a review" on page 450](#).

5. Click **Shelve**.

For more information, see "Resolving Files" in the [P4 Visual Client \(P4V\) Documentation](#).

Change the author of a review

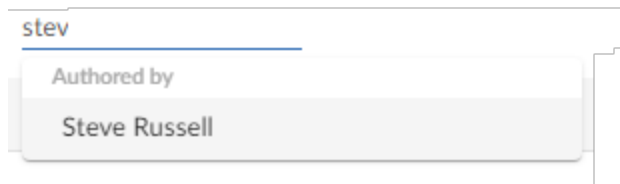
Note:


By default you cannot change the author of a review, this option must be enabled by your P4 Code Review administrator. See ["Allow author change" on page 675](#) for details.

If the author of a review is no longer available, or ownership of a review is passed to a different developer, it is useful to be able to change the author of that review.

To change the author of a review:

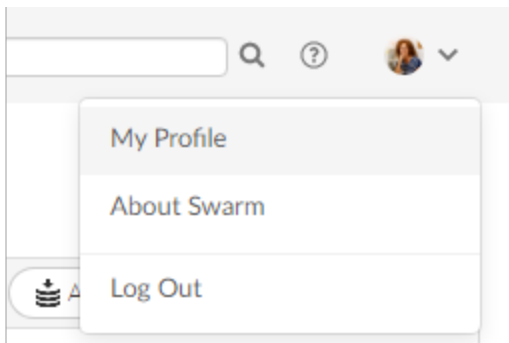
1. Click **Reviews** in the main menu.
2. Click the **Role** drop down and search for reviews by the author you want to replace. The list will auto-complete with users as you type.



3. Select the author from the list, reviews by that author are displayed.
4. Click the review you want to change the author on.
5. Click the **Review actions**  button.
6. Select **Change author**.
7. Select the new review author from the user list.
8. Click **Save** to change author of the review.

Change your user notification settings

1. Click on your *userid* in the header and select **My Profile**.



2. Click the **Notifications** tab to display your user notifications settings.
3. Configure which notifications you receive when events occur within P4 Code Review, see the ["Notifications tab" on page 349](#).

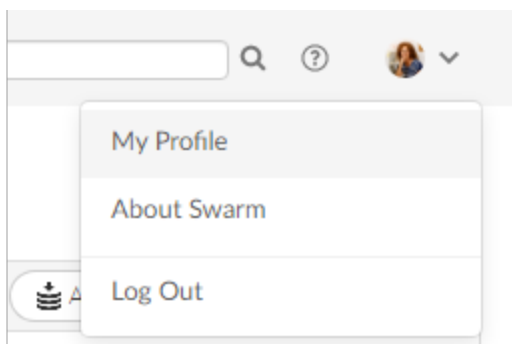
Unfollow all projects and users you are following

When you are viewing your own user profile, you can unfollow all projects and users that you are following.

Note:

This action cannot be undone.

1. Display your own user profile when you are logged into P4 Code Review by clicking on your *userid* in the header and selecting **My Profile**.



Your user page is displayed:

Users/Allison.Clayborne

allison.clayborne

Activity Shelves Settings Notifications Reviews Commits Comments Jobs

2 FOLLOWERS 1 FOLLOWING 6 PROJECTS

FULL NAME
Allison Clayborne

EMAIL ADDRESS
allison.clayborne@nowhere.xjzzj.com

Unfollow all Projects and Users for allison.clayborne

Allison Clayborne updated project (JPlugin)
A Java plugin for continuous integration. 4 days ago

Allison Clayborne updated project (JPlugin)
A Java plugin for continuous integration. 4 days ago

Allison Clayborne updated project (JPlugin)
A Java plugin for continuous integration. 4 days ago

Allison Clayborne commented on review 264 (revision 1) for mercury:dev
The strings should be declared as constants. 25 days ago
3 comments

Allison Clayborne rejected review 232 (revision 1) for test-data:data
Updating tests for test-18.txt, about a month ago
Add a comment

Allison Clayborne approved review 230 (revision 1) for test-data:data
Updating tests for test-17.txt, about a month ago
Add a comment

2. Click the **Unfollow all projects and users for *your-username*** button.
3. Click **OK** when the confirmation dialog is displayed to complete the unfollow action.
4. You will no longer be following any projects or users.

P4 Code Review administration tasks

This section describes a number of the common P4 Code Review administration tasks:

In this section:

Change notification settings for a group

Important:

You must be an owner of the group, have *super* privileges in P4 Server (**p4d**), or have *admin* privileges in **p4d** version 2012.1 or later, to edit group notifications. If you do not have sufficient permissions, P4 Code Review does not display the **Notifications** tab for the group.

1. Click **Groups** in the main menu.
2. Click the group you want to edit.
3. Click the group **Notifications** tab.
 - If **Group mailing list** is **Disabled**, emails are sent to the group members individual email addresses and only the following notification options are available:

- **Email members when a review is requested:** When any member of this group creates a review, this group will be notified.
- **Email members when a change is committed:** When a change is committed into Perforce, if the owner of the changelist is a member of this group, this group will be notified.

Tip:

When a user commits a changelist in P4 Code Review, it is committed on behalf of the changelist owner. If the changelist owner is a member of this group, this group will be notified.

- If **Group mailing list** is **Enabled** notifications are sent to the group email address and additional notifications are available for the group, see [group notifications](#) for details.

When a group mailing list is enabled it overrides the group member's preference set for the `notify_self` configurable in the `SWARM_ROOT/data/config.php` file. So even when the `notify_self` configurable is set to false the group member will receive an email notification.

There is a caveat that if a user is a group member and the same user is added individually to the review and has set the `notify_self` configurable to true, the user receives two email notifications.

For more information about `notify_self` configurable, see "[notify_self](#)" on page 598.

Manage project branches

Initial steps:

1. Visit your project page.
2. Click **Settings** in the menu.

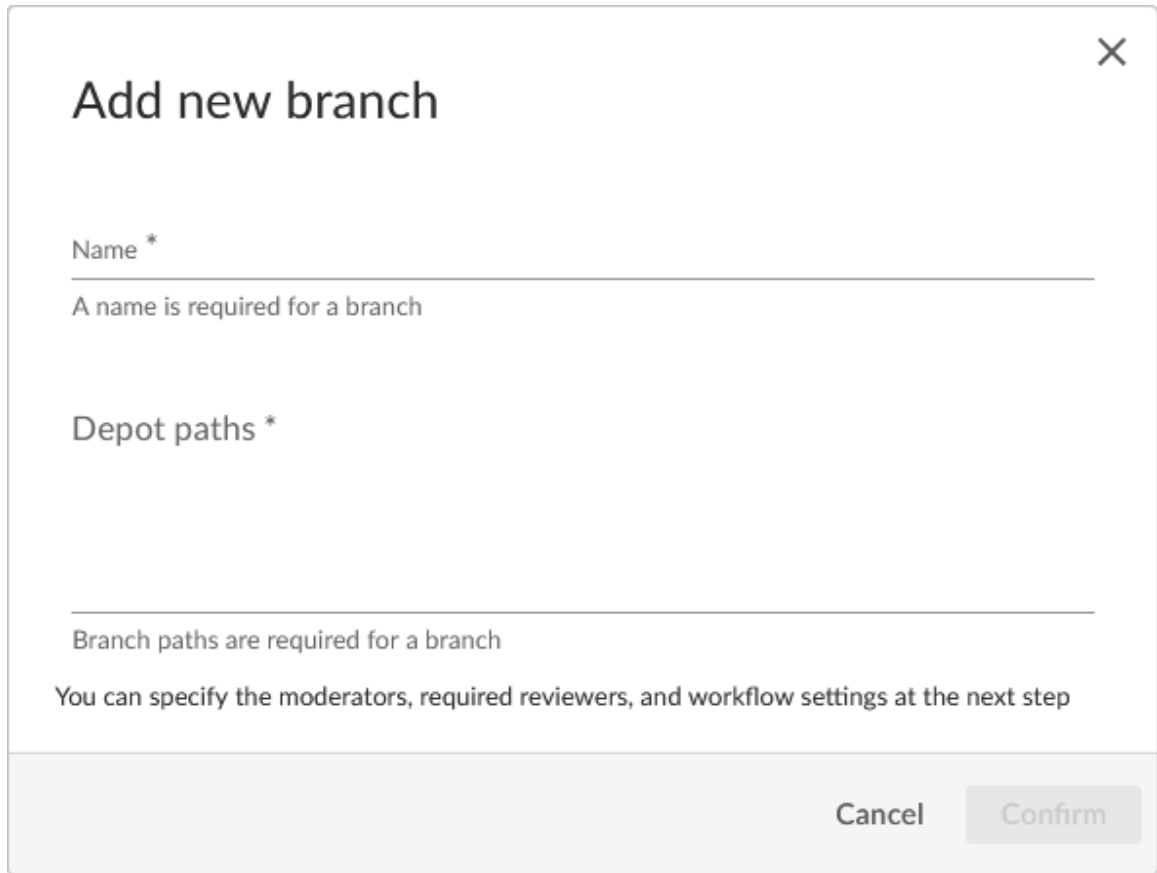
A project summary with Participants tab, Branches tab, Integrations tab, and General Settings tab is displayed.

3. Click **Branches** tab.

Here you can view and create project branches.

Adding a branch

1. Click the **Add new branch** button. This opens an **Add new branch** dialog.



Add new branch

Name *

A name is required for a branch

Depot paths *

Branch paths are required for a branch

You can specify the moderators, required reviewers, and workflow settings at the next step

Cancel Confirm

2. Enter a short **Name** for your branch.
3. Enter one or more branch paths in the **Depot paths** field using depot syntax, one path per line. For more information on branch paths, see ["Example branch paths" below](#).

Important: Branch paths are case sensitive.

4. Click the **Confirm** button.

A new branch is created.

Example branch paths

If you have entered multiple branch paths into the Depot paths field, these paths are processed in order, starting with the first path in the list.

Branch paths, and files can be excluded by putting a minus symbol - at the start of the path. In the following example:

- The first path includes all of the directories and files under `//depot/main/swarm/` in the project branch.
- The second path excludes all of the files in `-//depot/main/swarm/test/` from the project branch.

- The third path includes the ResultSummary.html file from the previously excluded `//depot/main/swarm/test/` directory.

```
//depot/main/swarm/...
-//depot/main/swarm/test/...
//depot/main/swarm/test/ResultSummary.html
```

Wildcards are supported in the project branch paths. For more information, see ["Wildcards in branch paths" on page 485](#).

Edit a branch

Once you have created a branch, additional options become available for managing the branch further. Click on the name of the branch to expand the branch pane to view the additional options. All settings described in this section are optional.

Participants
Branches
Integrations
General Settings

Associated Branches Add new branch

Main
^

Name *
Main

The branch name

Workflow
Minimum up votes

Inherit from project
2

Depot paths *

//depot/main/swarm/...
-//depot/main/swarm/test/...
//depot/main/swarm/test/ResultSummary.html

Branch moderators(1), only moderators can approve or reject reviews
Manage moderators

Default reviewers(1)
☐ Retain default reviewers
Manage default reviewers

Delete branch

Workflow

You can associate a workflow with the project branch. If you have disabled the workflow feature, this option will not be displayed.

To associate a workflow with a project branch, select one from the **Workflow** dropdown list, or enter the workflow name in the search field. You will only be able to see the workflows that you own or shared workflows. When a workflow is associated with a project, the workflow is used for all of the branches in that project.

To use the parent project workflow, select **Inherit from project** from the **Workflow** dropdown list. This option is selected by default when you create a new branch. If the parent project is set to **No workflow**, the branch will use the global workflow rules. When a project branch is associated with a workflow, the workflow of the parent project is ignored and the branch workflow is used.

To remove the existing workflow without replacing it with another workflow, select **No Workflow** from the **Workflow** dropdown list. This is the default when you create a new project.

For more information about workflows and how project workflows interact with branch workflows, see ["Workflow basics" on page 512](#).

Minimum up votes

Use **Minimum up votes** to set the minimum number of up votes required for reviews in this branch.

A review cannot be approved until all of the [Required reviewers](#) have voted up the review and the **Minimum up votes** specified has been satisfied.

When this setting is defined at the project level, minimum number of up votes is automatically applied to all branches. However, if you set **Minimum up votes** to **1** or higher on a specific branch, that branch will use its own setting and ignore the project-level setting.

To use the project-level value, set **Minimum up votes** to **Inherit from project**. This is set by default for new branches.

To override the project setting and use a custom value for a certain branch, set **Minimum up votes** to **1** or higher on that branch.

Notes on minimum up votes

- If a review includes changes across multiple projects or branches, the minimum number of up votes on each project or branch must be met before the review can be approved.
- Required reviewers are included when counting up votes.
- When the workflow rule **Count votes up from** is set to **Members** for a project or branch, only up votes from members belong to that project or branch will count toward the minimum up votes requirement. For more information on the **Count votes up from** rule, see ["Workflow rules" on page 510](#).

If workflows are disabled, all votes are counted, not just votes from project members.

Restriction: If the **Minimum up votes** requirement is higher than the total number of reviewers on a review, it will be impossible to approve it, even if every reviewer votes to approve. Take this into consideration when setting **Minimum up votes**.

Branch moderators

Only moderators can approve or reject reviews on the branch.

To add moderators to a branch:

1. Click the **Manage moderators** button.
2. Choose the **Users** or **Groupstab**.
 - a. If you selected **Users**, start typing a the name of a user. Suggested users on the P4 Server will appear automatically.
 - b. If you selected **groups**, then all of the members of that group will have the same moderator privileges for that project branch. Use this option if you wanted to add several users at once.
3. Click Add.

After you finish setting up the branch and save the project, reviews on this branch can only be approved or rejected by moderators.

Moderators can also transition a review to any other state. Below is a summary of branch moderator roles and review state rules.

Role	Allowed actions	Restrictions
Moderators	Can move a review to any state. Can approve or reject reviews. Can prevent automatic approvals.	None (unless <code>disable_self_approve</code> is enabled for self-approval). For more information, see disable_self_approve .
Authors (not moderators)	Can change state to: <ul style="list-style-type: none"> ■ Needs review ■ Needs revision ■ Archived Can attach committed changelists.	Cannot approve or reject reviews.



Role	Allowed actions	Restrictions
Authors (also moderators)	Can change state to: <ul style="list-style-type: none"> ■ Approve ■ Rejected ■ Needs review ■ Needs revision ■ Archived Can attach committed changelists.	Cannot approve their own reviews if <code>disable_self_approve</code> is enabled. For more information, see disable_self_approve .
Project members	Can change state to: <ul style="list-style-type: none"> ■ Needs review ■ Needs revision Can attach committed changelists.	Cannot approve, reject, or archive reviews.
Other users	None	Cannot change review states.
All roles	Can start a review.	Cannot change state if it's not in their allowed states (for example, can't change from Rejected).

Additional notes

- Moderators can prevent automatic approvals. For more information about automatically approving reviews using workflow rules see ["Workflow rules" on page 510](#).
- By default, if a review spans multiple branches with different moderators, only one moderator from any branch needs to approve it. You can change this (via a global setting) to require one moderator per branch. If a moderator belongs to multiple branches, one approval counts for each. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

To delete a moderator, click the delete icon against their name.

Branch moderators(1), only moderators can approve or reject reviews [Manage moderators](#) ^

Name	Type
 Allison Clayborne	User 

Rows per page: 25 ▾ 1-1 of 1 < >

Default reviewers

You can add default reviewers in the **Default reviewers** pane. To add a default reviewer:

1. Click the **Manage default reviewers** button.
2. Choose the **Users** or **Groupstab**.
 - a. If you selected **Users**, start typing a the name of a user. Suggested users will appear automatically.
 - b. If you selected **groups**, then all of the members of the selected group will be added as a default reviewer.

Restriction: You can add up to 25 users or groups combined.

3. Click Add.

From now on, each new review on this branch will automatically include these reviewers.

To keep default reviewers from being removed on reviews for this branch, turn on **Retain default reviewers**. For more information about retained default reviewers, see [Retain default reviewers](#).




Default reviewers(4)  Retain default reviewers

After you add default reviewers, you can control whether their votes are required or optional for approving a review. This is done using the **Vote Requirement** column.

Tip: Click on the Default reviewers pane to expand the section and view the table of default reviewers.






For **users**, select on of the following vote requirements.

- **Optional** - This users vote is not required to approve a review.
- **Required** - This users vote is required to approve a review

Vote Requirement		Type
Required		User
Optional	Optional 	User
Optional	Required 	User

For groups, select on of the following vote requirements.

- **Optional** - The vote from users in this group is not required to approve a review.
- **Required one** - At least one member of the group must approve the review, and no member votes down the review either.
- **Required all** - All members of the group must vote up the review to approve it.

Vote Requirement		Type ↑
Optional		Group
Required	Optional 	User
Optional	Require one 	User
Optional	Require all 	User
Optional		User

Default reviewers for reviews in multiple projects or branches:

When a review belongs to more than one project or branch:

- The default reviewers from all the related projects and branches are combined and added to the review.
- If a reviewer has different vote requirements in different places, the strictest setting is used.

For example:

- In Project-A, Reviewer X is optional.
- In Project-B, Reviewer X is optional.
- In the project branch, Branch-b, Reviewer X is required.

As a result, Reviewer X will be added as a required reviewer on the review.

Mentioning default reviewers

If users or groups are [@mentioned](#) in a new changelist description that includes [#review](#), they will be added to the review as reviewers. If any of these reviewers are already specified as default reviewers they will not be added to the review again, the reviewer's most restrictive reviewer option is used for the review.

To remove a user or group as a default reviewer, click the delete icon in their row of the table.

Important: If a default reviewer is deleted from P4 Server they will not be added to new reviews.

Removing a branch

1. Follow the [initial steps](#).
2. Expand the branch pane you want to delete.
3. Click **Delete Branch**.
A confirmation dialog appears.
4. Click **Yes** to delete the branch.

Unfollow all projects and users for another user

Note:

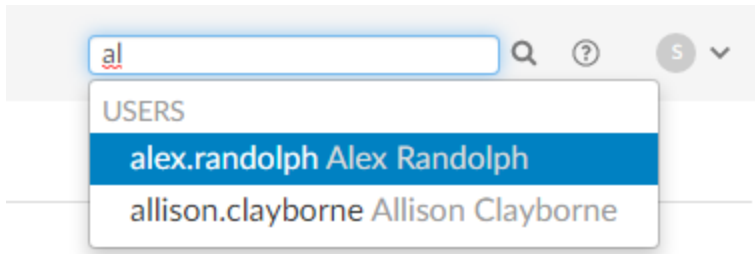
You must be logged in as a user with *admin* or *super* user privileges and viewing the user's profile page to perform this action.

When a user has been removed from the P4 Server but they are still following projects and users, it is useful to be able to remove all of their follows. This helps to keep the project and user follower lists up to date.

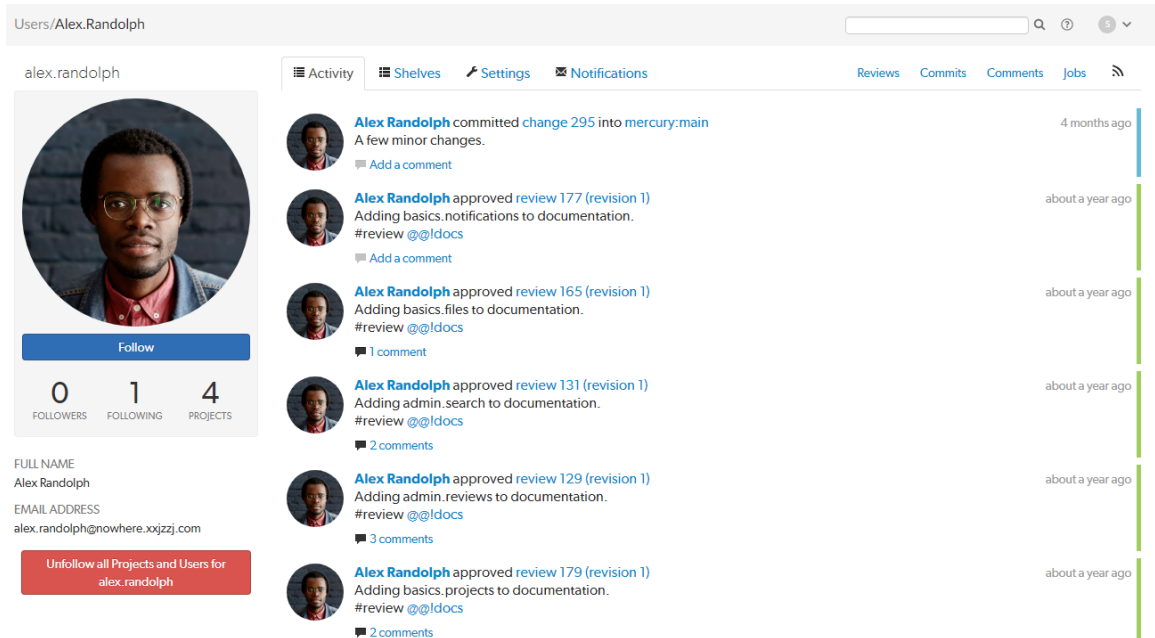
Note:

This action cannot be undone.

1. Use the **Search** box in the header to search for the user, the search field will auto-complete as you type.



2. Select the user from the search results to display the user's profile:



3. Click the **Unfollow all projects and users for *username*** button located below the user's email address.
4. Click **OK** when the confirmation dialog is displayed to complete the unfollow action.
5. The user will no longer be following any projects or users.

Obliterate a review

Note:

- By default, you must be a user with *admin* or *super* user rights to obliterate a review.
- **Optional:** P4 Code Review can be configured to allow users to obliterate reviews that they have authored. This can be configured by your P4 Code Review administrator. See "Allow author obliterate review" on page 675.


Obliterate is used to permanently delete reviews that have been created by mistake. For instance, if a review is associated with the wrong changelist, or a review contains sensitive information that should not be openly available.

For information on what happens to a review when it is obliterated, see ["When you obliterate a review" on page 657](#).

Important:

Obliterate must be used with care, the review and all of its associated metadata are permanently deleted. An obliterated review cannot be reinstated, not even by Perforce Support.

To obliterate a review:

1. Navigate to the review.
2. Click the **Review actions**  button and select **Obliterate Review**.
3. Click **Yes** on the confirmation dialog to complete the obliterate action.
4. The review is obliterated.

Change the logging level

P4 Code Review logs various activities to the data/log file. Change the logging level to increase or decrease the volume of log data by editing a configuration file.

An example configuration, in the `SWARM_ROOT/data/config.php` file:

```
<?php
// this block should be a peer of 'p4'
'log' => array(
    'priority' => 3, // 7 for max, defaults to 3
),
```

The maximum value for the log priority is 7; higher values do not result in increased logging. The minimum value is 0, which means no logging; lower values do not result in further logging reductions. For more information, see ["Logging" on page 625](#).

Check the queue workers

Note:

- P4 Code Review package installations are automatically configured with a cron job to spawn workers when you run the `configure-swarm.sh` post-installation configuration script during installation.
- P4 Code Review tarball installations should be setup with a cron job to ensure that workers are running to process events, this is part of the normal installation process. For information on setting up a cron job to spawn workers, see ["Set up a recurring task to spawn workers" on page 208](#).

P4 Code Review uses a custom queue system to process events, provide notifications, and more. The queue system is required to handle the potentially large volume of events from a busy P4 Server. A queue manager ensures that sufficient queue workers are available to process items.

You can check your queue workers in the following ways:

- If you do not have *admin* or *super* user permissions, see ["HTTP request"](#) below
- If you have *admin* or *super* user permissions, see ["System information page"](#) below

HTTP request

Check the status of the queue by making an HTTP request to `/queue/status`. The response is formatted in JSON and looks like this:

```
{"tasks":0,"futureTasks":1,"workers":3,"maxWorkers":3,"workerLifetime":"595s"}
```

This response indicates that the queue has no current tasks, there is 1 task scheduled for processing later, there are 3 queue workers available, at most 3 workers are created, and queue workers run for at most 10 minutes before self-terminating.

If the queue manager has stopped for some reason, start a new one by making an HTTP request to `/queue/worker`. No response is provided for this request.

System information page

The **Queue info** tab on the **System Information** page displays P4 Code Review basic queue worker information. As well as displaying worker information you can also display the task queue, start a worker, and start a temporary worker.

To navigate to the **Queue Info** tab:

1. Click your **User id** avatar, to the right of the P4 Code Review header.
2. Click **About Swarm**.
3. Click **System Information**.
4. Click the **Queue Info** tab.

System Information

Perforce
Log
PHP Info
Queue Info
Cache Info

Tasks	0
Future Tasks	4
Workers	1
Max Workers	3
Worker Lifetime	595s
Ping Error	false

Start a Worker
Refresh Tab
Show Task Queue
Start a Temp Worker

- For information about the **Queue Info** tab, see ["Queue Info" on page 715](#).

Automatically deploy code within a review

Important:

The project level test and deploy code features will be deprecated in a later P4 Code Review release. We recommend you use test integration to automatically deploy code within a review. For more information, see ["Add a test" on page 528](#).

Deploying code in a code review automatically involves enabling **Automated Deployment** in your project's configuration and providing a *trigger URL*. When the *trigger URL* is requested, P4 Code Review expects a deployment program to be executed.

When the deployment processing ends, P4 Code Review expects either a *success callback URL* or *failure callback URL* to be requested by your deployment program. These callback URLs should include a url parameter (either via GET or POST); when a valid-looking URL is included, clicking the deployment status indicator directs the user to the specified URL. This is intended to facilitate easy viewing of the successfully deployed review, or a report indicating why the deployment failed. The url parameter is mandatory for successful deployments, but is optional for failures.

- Navigate to the project page.
- Click the project **Settings** tab to display the **Project Settings** page.
- Automated Deployment** checkbox: select **Enable** to display the configuration field:

Automated Deployment ☒ Enable

`http://deploy-server/deploy?change={change}`

A URL that will trigger a deployment when reviews are created or updated.
Some special [arguments](#) are supported. [See help for more details.](#)

4. Provide a URL that triggers your deployment execution.

Special arguments are available to inform your deployment program of various details from P4 Code Review:

Note:

P4 for Jenkins 1.10.11 and later: P4 Code Review must send the parameters for the build to Jenkins as a POST request. To do this, enter the parameters in the **Post Body** and select **URL Encoded**.

`{change}`
The change number
`{status}`
Status of the change, *shelved* or *submitted*
`{review}`
The review's identifier
`{project}`
The project's identifier
`{projectName}`
The project's name
`{branch}`
The branch identifier(s), comma-separated
`{branchName}`
The branch name(s), comma-separated
`{success}`
Deployment successful callback URL
`{fail}`
Deployment failure callback URL

3 | Install and upgrade P4 Code Review

This chapter covers the initial installation and configuration of P4 Code Review as well as upgrading an existing P4 Code Review installation.

Important restrictions

Do not prefix group names, project names, user names, or client-names with "*swarm-*", this is a reserved term used by P4 Code Review. Prefixing a name with "*swarm-*" will result in unexpected and unwanted behavior in P4 Code Review.

For example:

Prefixing a group name with "*swarm-project-*" will result in, but is not limited to, the following issues:

- P4 Code Review notifications will not be processed correctly for the group.
- The group will not be visible in P4 Code Review.

Review the runtime dependencies

First, review the runtime dependencies before you install P4 Code Review, see "[Runtime dependencies](#)" on page 91.

P4 Code Review and P4 Server installation considerations

Before installing P4 Code Review and P4 Server you should consider the following:

- To ensure that all characters, including Unicode characters, are displayed and handled correctly by P4 Code Review, configure your P4 Server in Unicode mode. For information on configuring your P4 Server in Unicode mode, see [Set up and manage Unicode installations](#) in the [P4 Server Administration Documentation](#).
- For a small system, you can run P4 Code Review and P4 Server on the same machine.
- For larger systems, we recommend that P4 Code Review and P4 Server are run on separate machines. The machines should be close to each other to maximize network performance.

P4 Code Review needs to know about a number of P4 Server events to operate correctly. To enable this to happen, P4 Server Extensions (recommended) or P4 Server Triggers are installed on the P4 Server machine and they need to talk to P4 Code Review. In either case, performance can be negatively affected if network lag between P4 Code Review and the P4 Server is high.

- The P4 Code Review and P4 Server machines do not need to have the same operating system. For example, P4 Server could be on a Windows server and P4 Code Review could be on a RHEL server.

Note:

You cannot install P4 Code Review on a Windows machine.

Choose the installation process

Once you have reviewed the ["Runtime dependencies" on the facing page](#) and know that you can satisfy them, there are a number of ways to install P4 Code Review.

Note:

We recommend the package installation method to install P4 Code Review whenever possible, see ["Install and configure P4 Code Review from a package \(recommended\)" on page 106](#). Package installs ensure that all of the P4 Code Review dependencies are installed and this is the easiest way to install P4 Code Review. For a list of recommended operating systems for P4 Code Review, see ["Recommended operating systems" on page 92](#).

Choose one of the following installation methods (we recommend the package installation method whenever possible):

■ **P4 Code Review RPM or Debian packages (recommended):**

1. Follow the steps provided in ["Install and configure P4 Code Review from a package \(recommended\)" on page 106](#).
2. If you installed P4 Server Extensions in the previous step, skip this step and go to step 3:
Configure P4 Server for P4 Code Review, see ["Installing triggers" on page 187](#).
3. Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).
4. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

You are all set to start using P4 Code Review. Enjoy!

Tip:

To get started with P4 Code Review, see the ["Quickstart" on page 23](#) chapter.

■ **P4 Code Review Docker container:**

1. Follow the steps provided in [Run Swarm using a Docker container](#).
2. If you installed P4 Server Extensions in the previous step, skip this step and go to step 3:
Configure P4 Server for P4 Code Review, see ["Installing triggers" on page 187](#).
3. Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).
4. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

You are all set to start using P4 Code Review. Enjoy!

Tip:

To get started with P4 Code Review, see the ["Quickstart" on page 23](#) chapter.

- **P4 Code Review.tgz (Tarball):**

1. Follow steps provided in the ["Install and configure P4 Code Review manually from a Tarball" on page 166](#).
2. Configure Redis, see ["Redis configuration" on page 167](#).
3. Configure Apache, see ["Apache configuration" on page 171](#).
4. Configure PHP, see ["PHP configuration" on page 174](#).
5. Configure P4 Code Review, see ["P4 Code Review configuration" on page 179](#).
6. Configure the P4 Server to notify P4 Code Review about P4 Server events, see ["Configuring P4 Server event notification" on page 182](#).
7. Set up a recurring task to spawn workers, see ["Set up a recurring task to spawn workers" on page 208](#).
8. Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).
9. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

You are all set to start using P4 Code Review. Enjoy!

Tip:

To get started with P4 Code Review, see the ["Quickstart" on page 23](#) chapter.

Upgrade P4 Code Review

If you already have a working P4 Code Review installation and you want to upgrade P4 Code Review to a newer release, see ["Upgrading P4 Code Review" on page 231](#).

Runtime dependencies

In order to successfully install, configure, and deploy P4 Code Review, the following dependencies are required:

- A recommended operating system, see ["Recommended operating systems" on the next page](#)
- An Apache server with the modules required by P4 Code Review installed, see ["Apache web server" on page 93](#)
- A supported version of PHP with the modules required by P4 Code Review installed, see ["PHP" on page 94](#).
- A supported P4 Server deployment, and the ability to connect to it from the system hosting P4 Code Review, see ["P4 Server requirements" on page 98](#)

Note:

"P4 Server deployment" can refer to a running p4d or a proxy, replica, edge server, or commit server.

- P4 Server Extensions or Triggers to notify P4 Code Review about P4 Server events, see ["P4 Server event notification" on page 99](#).
- curl or wget for P4 Code Review worker operation, see ["Worker dependencies" on page 101](#)
- If Security-enhanced Linux (SELinux) is installed it must be configured correctly, see ["Security-enhanced Linux \(SELinux\)" on page 103](#)

Warning:

There is a risk of P4 Code Review missing data updates if P4 Code Review has been shut down while P4 Server is running and users are executing P4 Server commands.

When P4 Code Review is offline, workflow triggers will fail which could cause P4 Serveto stop working.

Optional dependencies:

- **LibreOffice:** required to view office-type documents. For more information, see ["LibreOffice" on page 103](#).
- **zip:** command-line archiving tool: Required to download zip archives of files and folders. For more information, see ["Zip" on page 103](#).
- **P4 AS SSO:** enables P4 Code Review to authenticate with SSO when the P4 Server is configured for P4 AS, see ["P4 AS SSO" on page 103](#).
- **Sendmail or equivalent:** required to use P4 Code Review email notifications. For more information, see ["P4 Code Review email notifications" on page 103](#).

Recommended operating systems

Perforce recommend that P4 Code Review is installed on one of the following operating systems:

- **Ubuntu 22.04 LTS, and 24.04 LTS** latest stable release should be used.
- **RHEL 8 and 9** latest stable release with PHP libraries installed from the Remi repository. See ["PHP" on page 94](#).
- **Rocky Linux 8 and 9** latest stable release with PHP libraries installed from the Remi repository. See ["PHP" on page 94](#).
- **Amazon Linux 2:** latest stable release with PHP libraries installed from Amazon Linux Extras. See ["PHP" on page 94](#).

Note:

You cannot install P4 Code Review on a Windows machine.

Other Linux distributions

Important:

P4 Code Review has not been tested on Linux distributions that are not on our recommended list. Because of this, our Support team may not be able to help you to the same extent as they would for the recommended operating systems.

P4 Code Review will probably run on any Linux distribution based on RHEL 8 or 9. P4 Code Review includes binary versions of P4 API for PHP, the Perforce extension for PHP. This P4 API for PHP dependency typically sets the limit to what Linux distributions you can install P4 Code Review on.

You might be able to get P4 Code Review running on another platform if you build P4 API for PHP yourself and satisfy the other runtime dependencies. Instructions on how to obtain and build P4 API for PHP from source can be found in the [P4 API for PHP Release notes](#).

Important:

P4 API for PHP does not support threaded operation. If you compile P4 API for PHP from source, ensure that the version of PHP you compile for is non-threaded.

Apache web server

P4 Code Review requires Apache HTTP Server 2.4 or newer:

- [Apache HTTP server project](#)

Important:

The following notes are applicable for RHEL 8 and RHEL 9:

- **RHEL 8:** Use the Remi repository configuration package (remi-release-8.rpm) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the epel-release-latest-8.noarch.rpm repository configuration package to give P4 Code Review access to EPEL packages.
- **RHEL 9:** Use the Remi repository configuration packages (remi-release-8.rpm and remi-release-9.rpm) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the epel-release-latest-8.noarch.rpm and epel-release-latest-9.noarch.rpm repository configuration package to give P4 Code Review access to EPEL packages.
- **P4 Code Review 2020.2 and later:** these versions of P4 Code Review use the Remi repository for RHEL 8 and RHEL 9. This provides PHP 8.x installed in the standard file system structure. This means that the old httpd24-httpd version of Apache is no longer needed, and the standard system version of Apache is being used again.

The SCL Apache site configuration file was installed at this location for P4 Code Review 2019.1 to 2020.1:

/opt/rh/httpd24/root/etc/httpd/conf.d/perforce-swarm-site.conf

If this exists when P4 Code Review is upgraded to 2020.2 and later, this file is copied to /etc/httpd/conf.d/perforce-swarm-site.conf if there is no file at the destination. It is also re-written to change references from /var/log/httpd24 to /var/log/httpd

If a site configuration file for P4 Code Review already exists in /etc/httpd, the copy and re-write is not performed.

After upgrade, httpd24-httpd is disabled.

- To avoid seeing the Apache HTTP server Linux test page when you start the Apache server, comment out the content of the welcome.conf file located in the /etc/httpd/conf.d/ directory.
- To avoid loading the Apache HTTP server example configuration instead of the P4 Code Review configuration when the Apache server starts, rename the autoindex.conf file located in the /etc/httpd/conf.d/ directory to z-autoindex.conf or similar. This is required because Apache runs the first conf file it finds in the /etc/httpd/conf.d/ directory (alphabetical order) and that must be the perforce-swarm-site.conf file.

P4 Code Review also requires the following Apache modules:

- **Ubuntu:** `mod_php` for interacting with PHP (usually installed with PHP)

Where `x` is the version of PHP you are running, for example 8.1. For a list of supported PHP versions, see ["PHP" below](#).

- **RHEL:** `php-fpm` for interacting with PHP (usually installed with PHP)
- `mod_rewrite` URL rewriting engine

For more information, see [Apache Module mod_rewrite](#).

Important:

Only the prefork MPM is supported. Use of the worker or event MPMs is not supported and is likely to cause problems because P4 API for PHP does not support threaded operation.

For more information on the prefork MPM, see [Apache MPM prefork](#).

PHP

P4 Code Review 2022.2 supports PHP 8.1, 8.2, 8.3, and 8.4. For more information about PHP, see the [PHP website](#).

Important:

- From versions 2025.1 and above, P4 Code Review no longer supports PHP 7 or older versions.
- P4 Code Review no longer supports PHP 8.0 version.

- PHP must be non-threaded because P4 API for PHP does not support threaded operation.
- **RHEL 8:** Use the Remi repository configuration package (remi-release-8.rpm) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the epel-release-latest-8.noarch.rpm repository configuration package to give P4 Code Review access to EPEL packages.
- **RHEL 9:** Use the Remi repository configuration packages (remi-release-8.rpm and remi-release-9.rpm) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the epel-release-latest-8.noarch.rpm and epel-release-latest-9.noarch.rpm repository configuration package to give P4 Code Review access to EPEL packages.

Required PHP extensions

P4 Code Review requires the following PHP extensions:

Tip:

If you install P4 Code Review from a package, all of these PHP extensions are automatically pulled in as dependencies on all platforms.

- **iconv** (character encoding converter)
For more information, see [iconv](#).

This is typically enabled by default with most PHP distributions
- **JSON** (JavaScript Object Notation)
For more information, see [JavaScript Object Notation](#).

This is typically enabled by default with most PHP distributions, although recent distributions are making this optional.
- **Session** (session handling)

This is typically enabled by default with most PHP distributions
- **P4 API for PHP version 2019.1 or later** (the Perforce PHP Extension)

This release of P4 Code Review contains the latest version of P4 API for PHP with the P4 Code Review package and tarball installations.

Note:

P4 Code Review package and tarball installations: two versions of P4 API for PHP are supplied for PHP 8 version supported by P4 Code Review. They are located in the p4-bin/bin.linux26x86_64 directory.

- perforce-php8x.so compatible with systems using SSL 1.0.2
- perforce-php8x-ssl1.1.1.so compatible with systems using SSL 1.1.1

- `perforce-php8x-ssl3.so` compatible with systems using SSL 3.0.0 (by default, Ubuntu 22.04 and 24.04 uses SSL 3.0.0)

Where x is the version of PHP 8.

P4 Code Review no longer supports PHP 8.0 version.

If the `perforce.ini` file is not pointing at the correct version of P4 API for PHP and you connect to an SSL enabled P4 Server:

- The P4 Code Review web-page will not load and you might see a Connection Reset error.
- There might be an undefined symbol: `SSLLeay` message in the Apache error log

Upgrading P4 Code Review:

- **P4 Code Review package:** the latest P4 API for PHP version is installed automatically.
- **P4 Code Review tarball installation:** you must configure P4 Code Review to use the version of P4 API for PHP in the new P4 Code Review tarball. For swarm tarball upgrade instructions, see ["Upgrading a tarball installation" on page 260](#).
- `php-xml` (DOM API for XML manipulation, the P4 Code Review RSS feed will not work if it is not installed)
- `php-mbstring` (multi-byte character strings, the P4 Code Review RSS feed will not work if it is not installed)
- `php-redis` (PHP extension for Redis, the P4 Code Review cache will not work if it is not installed)
- `php-gd` (PHP extension for creating the blur images used for image icons on comments. If `php-gd` is not installed, images attached to comments display the file extension name instead of a blur or thumbnail image.)

Troubleshooting out of memory (OOM) issues with PHP 8.x on RHEL platforms

On P4 Code Review servers running PHP 8.x on RHEL platforms, `php-fpm` threads could become idle or “zombie,” failing to release memory and causing memory bloat. Over time, this led to OOM kills and frequent `php-fpm` restarts.

To resolve, switch from `php-fpm` to `mod_php` with **Apache prefork MPM**. More steps on how to do this, see ["Troubleshooting out of memory \(OOM\) issues" on page 178](#).

Recommended PHP extension

P4 Code Review greatly benefits from the following PHP extension:

- `Imagick` (integrates ImageMagick into PHP to improve P4 Code Review's ability to preview graphics formats that web browsers typically cannot display and create to comment attachment thumbnails)

For more information about `Imagick`, see [Image Processing \(ImageMagick\)](#).

Installation instructions for [Imagick](#).

Note:

If you see an error similar to the one below for PS, PS2, EPS, PDF, or XPS files, the imagick policy.xml file needs to be edited. For example, the error for a PDF file will be similar to:

```
convert-im6.q16: attempt to perform an operation not allowed by the security policy
`PDF' @ error/constitute.c/IsCoderAuthorized/408
```

If you see the error above:

1. Open the ImageMagick policy.xml file for editing:
`/etc/ImageMagick-6/policy.xml`
2. Disable lines with the pattern below for the following `<filetype>s`: PS, PS2, EPS, PDF, or XPS:
`<policy domain="coder" rights="none" pattern="<filetype>" />`
3. Save the policy.xml file.
4. Restart Apache for the changes to become active.

Building PHP from source (not recommended)

If you build PHP from source, the following dependencies are required:

- openssl
- mcrypt
- zlib
- gettext
- curl
- apxs (Apache extension tool)

See the source PHP documentation for details on how to include these modules in your PHP build. There is normally a `--with-xxxx` option that defines where the dependency is loaded from.

For example for apxs, zlib, and openssl:

```
./configure --with-apsx2=<path_to_apsx> --with-zlib=<path_to_zlib_library> --with-openssl
```

P4 Server requirements

Note:

P4 Server 2020.2 and later: any new file being shelved that has the same content as an existing shelved file refers to the existing archive file instead of creating a duplicate archive file. No P4 Server or P4 Code Review configuration is required for this feature.

This P4 Server feature automatically reduces the space required for the P4 Code Review-managed shelved review changelists. P4 Code Review creates these changelists for its own internal use. P4 Server only updates new shelves, it does not retrospectively update your existing shelves.

For more information on the P4 Code Review-managed changelists, see ["Internal representation" on page 402](#).

P4 Code Review works with any [supported versions](#) of the P4 Server (Standard Maintenance). This release of P4 Code Review is tested against and supports the following P4 Server versions:

- 2023.2
- 2024.1
- 2024.2
- 2025.1

[Download P4 Server](#) from the Perforce website.

Important:

- P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).
- P4 Code Review does not support the P4 MFA Authenticator.

P4 Server automated user requirements for P4 Code Review

P4 Code Review requires an automated user with at least admin privileges in the P4 Server to enable P4 Code Review to run against the P4 Server. The P4 Code Review admin user also needs to be a [standard user type](#). This can be an existing user, or a new user created specifically to support P4 Code Review.

P4 Code Review does take up a license but if you are paying for the licenses you can request a background user for P4 Code Review. This is a normal user that is provided so you can use P4 Code Review without consuming one of your paid licenses. To access the request form for the background user, see [Helix Core Request for Background User](#).

Important:

If P4 AS is configured for your P4 Server, the user account running P4 Code Review must not use the P4 AS.

For more information about setting up P4 Server, see [Install the server](#) and [Upgrade the server](#) in [P4 Server Administration Documentation](#).

P4 Code Review and P4 Server installation considerations

Before installing P4 Code Review and P4 Server you should consider the following:

- To ensure that all characters, including Unicode characters, are displayed and handled correctly by P4 Code Review, configure your P4 Server in Unicode mode. For information on configuring your P4 Server in Unicode mode, see [Set up and manage Unicode installations](#) in the [P4 Server Administration Documentation](#).
- For a small system, you can run P4 Code Review and P4 Server on the same machine.
- For larger systems, we recommend that P4 Code Review and P4 Server are run on separate machines. The machines should be close to each other to maximize network performance.

P4 Code Review needs to know about a number of P4 Server events to operate correctly. To enable this to happen, P4 Server Extensions (recommended) or P4 Server Triggers are installed on the P4 Server machine and they need to talk to P4 Code Review. In either case, performance can be negatively affected if network lag between P4 Code Review and the P4 Server is high.

- The P4 Code Review and P4 Server machines do not need to have the same operating system. For example, P4 Server could be on a Windows server and P4 Code Review could be on a RHEL server.

Note:

You cannot install P4 Code Review on a Windows machine.

P4 Server event notification

P4 Code Review needs to know about a number of P4 Server events to operate correctly, this can be done by using P4 Server Extensions (recommended) or P4 Server Triggers. P4 Code Review installs include the P4 Server extension file and trigger scripts required for P4 Code Review to get the events it needs from your P4 Server.

Triggers are still supported, but we recommend you use P4 Server Extensions. P4 Server Extensions are easier to install and maintain than Triggers.

Use one of the following:

- **Recommended:** P4 Server Extensions, see ["P4 Server Extensions dependencies" below](#)
- P4 Server Triggers, see ["Trigger dependencies" on the next page](#)

P4 Server Extensions dependencies

Warning:

If you are using the P4 Server extension, do not install P4 Code Review triggers.

P4 Server Extensions must be installed on your P4 Server to complete your P4 Code Review installation. The P4 Code Review Extensions script is included with the P4 Code Review product download and is copied to your P4 Server during configuration.

To install the P4 Server extension you need:

A compatible version of P4 Server for the extension:

- **Linux:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.
- **Windows:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.

Trigger dependencies

Warning:

Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.

Tip:

We recommend you use P4 Server Extensions, P4 Server Extensions are easier to install and maintain than triggers. See "[P4 Server Extensions dependencies](#)" on the previous page.

P4 Code Review triggers must be installed on the P4 Server to complete the P4 Code Review installation.

The P4 Code Review triggers require perl 5.08+. To download Perl, see the [Perl Download page](#).

On the Windows platform, we have tested P4 Code Review against Strawberry Perl. There are two Perl modules that are also required which may not be part of a minimal Perl installation.

- `HTTP::Tiny` is required to make calls to the P4 Code Review server. If this is not present, then the trigger will attempt to use the command line curl program. This module is standard on Strawberry Perl on Windows, and available as a package with the version of Perl provided on Ubuntu 22.04.

`IO::Socket::SSL` is required if the P4 Code Review server is configured to use SSL and `HTTP::Tiny` is present. This is provided as standard by Strawberry Perl, and available on Linux.

Warning:

If the `HTTP::Tiny` module is not available the triggers require the use of curl. This must be installed for the triggers to function. On RHEL, for example, this can be done using the yum package installer using `yum install curl`.

P4 Code Review triggers also require the following perl modules to be installed:

- **Windows:**
 - `JSON` is required to exchange data between the browser and the server and is included by default with Strawberry Perl.
- **Ubuntu:**
 - `JSON` is required to exchange data between the browser and the server and must be installed.
 - `libjson-perl` is required to manipulate JSON formatted data and must be installed.

- **RHEL:**
 - JSON is required to exchange data between the browser and the server and must be installed.
 - `perl-JSON` is required to manipulate JSON formatted data and must be installed.

P4 Code Review package installation on the same machine as P4D

If P4 Code Review is installed from a package on the same machine as P4D, the triggers require the following perl modules to be installed:

- **Ubuntu:**
 - Perl 5.08+
 - `libio-socket-ssl-perl` is required if the P4 Code Review server is configured to use SSL.
 - `libjson-perl` is required to manipulate JSON formatted data and must be installed.
- **RHEL:**
 - Perl 5.08+
 - `perl-IO-Socket-SSL` is required if the P4 Code Review server is configured to use SSL.
 - `perl-JSON` is required to manipulate JSON formatted data and must be installed.

Worker dependencies

P4 Code Review uses short-lived workers to process the P4 Code Review queue, new workers are regularly spawned by a recurring task.

One of the following must be installed to ensure new workers are regularly spawned:

- **curl**, download from the [curl download page](#)
- **wget**, download from the [wget download page](#)

For more information about P4 Code Review workers, see "[Set up a recurring task to spawn workers](#)" on page 208.

Redis server

P4 Code Review requires Redis to manage its caches. P4 Code Review caches data from the P4 Server to improve the performance of common searches in P4 Code Review and to reduce the load on the P4 Server.

Redis is included with the P4 Code Review package and Tarball installations:

- **P4 Code Review package:** Redis is automatically installed on the P4 Code Review machine and P4 Code Review is automatically configured to use Redis.
- **P4 Code Review Tarball installation:** see the [Redis installation and configuration](#) section for instructions about installing Redis.

Tip:

When P4 Code Review starts it verifies the Redis cache, during this time you cannot log in to P4 Code Review. The time taken to verify the Redis cache depends on the number of users, groups, and projects P4 Code Review has. Start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves in the redis-server.conf file, see ["Redis server configuration file" on page 665](#).

Optional:

If you prefer to use your own Redis server, you must edit the Redis server connection configurable in P4 Code Review. For information on setting your Redis server connection, see ["Use your own Redis server" on page 224](#).

Supported P4 Visual Client (P4V)

P4 Code Review 2022.3 or later only works with P4 Visual Client (P4V) 2021.3 or later.

Supported P4 AS (HAS)

P4 Code Review works with any [supported versions](#) of P4 AS (Standard Maintenance).

Supported web browsers

The following browsers are supported for use with P4 Code Review:

- Apple Safari, latest stable version
- Google Chrome, latest stable version
- Mozilla Firefox, latest stable version
- Microsoft Edge, latest stable version

Other web browsers might also work, including prior, development or beta builds of the above web browsers, but are not officially supported.

Note:

- JavaScript and cookies must be enabled in the web browser for P4 Code Review to operate.
- Mobile web browsers are not supported by P4 Code Review.

Workflow prerequisites

For P4 Code Review 2019.2 and later, the workflow feature is enabled by default.

Tip:

If you are upgrading from an earlier version you will need to update your triggers, see ["Upgrading P4 Code Review" on page 231](#).

The workflow feature requires:

- Workflow must be enabled, see the ["workflow" on page 743](#) system configurable.
- Your `swarm-trigger.conf` file must be configured correctly, see ["Setup up P4 Code Review triggers" on page 188](#).
- Your P4 Server trigger table must contain the workflow trigger lines, see ["Setup up P4 Code Review triggers" on page 188](#).

Optional dependencies

LibreOffice

P4 Code Review can display previews of office-type documents when LibreOffice is installed on the P4 Code Review server. Installation is not required, but when LibreOffice is installed P4 Code Review automatically detects its presence.

For more information about LibreOffice, see ["LibreOffice" on page 543](#).

Zip

You can download a ZIP archive of files/folders when the zip command-line tool is installed on the P4 Code Review server.

For more information about installing and configuring Zip, see ["Zip archive" on page 544](#).

P4 AS SSO

P4 Code Review support for SSO using P4 AS with P4 Server requires P4 Server 2019.1 or later

For more information about configuring P4 Code Review to authenticate with P4 AS, see ["Single Sign-On PHP configuration" on page 703](#).

P4 Code Review email notifications

Sendmail or an equivalent is required to use P4 Code Review email notifications. By default, the configuration in `php.ini` relies on SendMail being installed. For more information about configuring email for P4 Code Review, see ["Email configuration" on page 595](#).

Security-enhanced Linux (SELinux)

P4 Code Review supports SELinux on RHEL and Amazon Linux 2. SELinux is an advanced access control mechanism that improves security for Linux distributions.

SELinux operates in one of three modes:

- enforcing: this mode blocks and logs any actions that do not match the defined security policy.
- permissive: this mode logs actions that do not match the defined security policy but these actions are not blocked.
- disabled: in this mode SELinux is off, actions are not blocked and are not logged.

Tip:

To check the mode SELinux is operating in, view the `/etc/selinux/config` file with `vi` or a similar editor:

```
root $ vi /etc/selinux/config
```

For instructions on configuring SELinux:

- **RHEL:** See "[SELinux configuration](#)" on page 137.
- **Amazon Linux 2:** See "[SELinux configuration](#)" on page 153.

If you see a P4 Code Review configuration error similar to the error shown below, SELinux has not been correctly configured for P4 Code Review. Check you have configured SELinux on [REHL](#) or [Amazon Linux 2](#) correctly.

Swarm has detected a configuration error

Problem detected:

- The data directory (`/opt/perforce/swarm/data`) is not writeable.

Please investigate the below PHP error below:

```
Cannot write to cache directory ('/opt/perforce/swarm/data/cache'). Check permissions.
```

```
/opt/perforce/swarm/module/Application/Module.php on line 329
```

For more information, please see the [Install and upgrade Swarm](#) documentation; in particular:

- [Runtime dependencies](#)
- [Installation](#)
- [PHP configuration](#)
- [Swarm configuration](#)

If you are using SELinux: check it is configured correctly, see [Security-enhanced Linux \(SELinux\)](#).

Please restart your web server after making any PHP changes.

If you change your configuration you must delete your Swarm config cache, see [Swarm config cache file delete](#).

Choose the installation process

Once you have reviewed the "[Runtime dependencies](#)" on page 91 and know that you can satisfy them, there are a number of ways to install P4 Code Review.

Note:

We recommend the package installation method to install P4 Code Review whenever possible, see ["Install and configure P4 Code Review from a package \(recommended\)" on the next page](#). Package installs ensure that all of the P4 Code Review dependencies are installed and this is the easiest way to install P4 Code Review. For a list of recommended operating systems for P4 Code Review, see ["Recommended operating systems" on page 92](#).

Choose one of the following installation methods (we recommend the package installation method whenever possible):

- **P4 Code Review RPM or Debian packages (recommended):**

1. Follow the steps provided in ["Install and configure P4 Code Review from a package \(recommended\)" on the next page](#).
2. If you installed P4 Server Extensions in the previous step, skip this step and go to step 3:
Configure P4 Server for P4 Code Review, see ["Installing triggers" on page 187](#).
3. Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).
4. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

You are all set to start using P4 Code Review. Enjoy!

Tip:

To get started with P4 Code Review, see the ["Quickstart" on page 23](#) chapter.

- **P4 Code Review Docker container:**

1. Follow the steps provided in [Run Swarm using a Docker container](#).
2. If you installed P4 Server Extensions in the previous step, skip this step and go to step 3:
Configure P4 Server for P4 Code Review, see ["Installing triggers" on page 187](#).
3. Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).
4. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

You are all set to start using P4 Code Review. Enjoy!

Tip:

To get started with P4 Code Review, see the ["Quickstart" on page 23](#) chapter.

- **P4 Code Review.tgz (Tarball):**

1. Follow steps provided in the ["Install and configure P4 Code Review manually from a Tarball" on page 166](#).

2. Configure Redis, see ["Redis configuration" on page 167](#).
3. Configure Apache, see ["Apache configuration" on page 171](#).
4. Configure PHP, see ["PHP configuration" on page 174](#).
5. Configure P4 Code Review, see ["P4 Code Review configuration" on page 179](#).
6. Configure the P4 Server to notify P4 Code Review about P4 Server events, see ["Configuring P4 Server event notification" on page 182](#).
7. Set up a recurring task to spawn workers, see ["Set up a recurring task to spawn workers" on page 208](#).
8. Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).
9. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

You are all set to start using P4 Code Review. Enjoy!

Tip:

To get started with P4 Code Review, see the ["Quickstart" on page 23](#) chapter.

Install and configure P4 Code Review from a package (recommended)

This chapter covers the initial installation and configuration of P4 Code Review from a package. P4 Code Review is available in two distribution package formats: Debian ([.deb](#)) for Ubuntu systems and RPM ([.rpm](#)) for RedHat Enterprise Linux (RHEL), and Amazon Linux 2.

Using distribution packages greatly simplifies the installation, updating, and removal of software, as the tools that manage these packages are aware of the dependencies for each package.

P4 Code Review packages are available for:

- **Ubuntu 22.04 LTS, and 24.04 LTS** latest stable release should be used.
- **RHEL 8 and 9** latest stable release with PHP libraries installed from the Remi repository. See ["PHP" on page 94](#).
- **Rocky Linux 8 and 9** latest stable release with PHP libraries installed from the Remi repository. See ["PHP" on page 94](#).
- **Amazon Linux 2:** latest stable release with PHP libraries installed from Amazon Linux Extras. See ["PHP" on page 94](#).

For instructions on installing P4 Code Review from a package:

Install and configure P4 Code Review on Ubuntu

P4 Code Review is available as a package for Ubuntu 22.04 LTS and 24.04 LTS. Using distribution packages greatly simplifies the installation, updating, and removal of software, as the tools that manage these packages are aware of the dependencies for each package.

Prerequisites

- Review the runtime dependencies before you install P4 Code Review, see ["Runtime dependencies" on page 91](#).
- Review the PHP requirements before you upgrade P4 Code Review, see ["PHP" on page 94](#).
- Review the P4 Server requirements before you install P4 Code Review, see ["P4 Server requirements" on page 98](#).
- Ensure the automated user has **admin** or **super** privileges in the P4 Server to enable P4 Code Review to run against the P4 Server. For more information, see ["P4 Server automated user requirements for P4 Code Review" on page 98](#).

P4 Code Review can be connected to P4 Servers (P4D) and commit servers.

- To configure P4 Code Review to connect to more than one P4 Server (P4D), see ["Multiple P4 Server instances" on page 638](#).
- To configure P4 Code Review to connect to a P4 Server configured to use *commit-edge architecture*, see ["Commit-edge deployment" on page 582](#).

Important: Do not connect P4 Code Review to P4 Broker, P4 Proxy, an edge server, forwarding replica, or read-only replica servers.

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

Installation

1. Configure the Perforce package repository, on the server to host P4 Code Review and on the server hosting your P4 Server.

Important: If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

Follow the instructions for your OS distribution:

- **Ubuntu 22.04:** As root, create the file `/etc/apt/sources.list.d/perforce.list` with the following content:

```
deb [signed-by=/usr/share/keyrings/perforce.gpg]
https://package.perforce.com/apt/ubuntu jammy release
```
 - **Ubuntu 24.04:** As root, create the file `/etc/apt/sources.list.d/perforce.list` with the following content:

```
deb [signed-by=/usr/share/keyrings/perforce.gpg]
https://package.perforce.com/apt/ubuntu noble release
```
2. Import the Perforce package signing key, on the server to host P4 Code Review and the server hosting your P4 Server.

Important: If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

If you are using an older version of OpenSSL that does not support TLSv1.2 or higher, replace 'https' with 'http' in the wget command.

Run the following commands as root for your OS distribution. The same command works on either **Ubuntu 22.04 or 24.04**:

```
wget -qO - https://package.perforce.com/perforce.pubkey | gpg --dearmor | sudo tee /usr/share/keyrings/perforce.gpg
```

```
sudo apt-get update
```

For information about how to verify the authenticity of the signing key, see [Perforce Packages](#).

3. Install the main P4 Code Review package on the server to host P4 Code Review.

```
sudo apt-get install helix-swarm
```

Important: When the P4 Code Review installation has completed, you are prompted to run the configure-swarm.sh post-installation script.

Do not run this script until you have completed the rest of these **Installation** instructions. Instructions for running the configure-swarm.sh post-installation script are in the **Post-Installation configuration** section referenced in the final step of the **Installation** instructions.

4. P4 Code Review needs to know about some P4 Server events to operate correctly. Use P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about these events. The P4 Server extension can be installed automatically by the P4 Code Review configure-swarm.sh post-installation script, but Triggers must be manually installed. To use P4 Server Extensions, ignore this step and skip to the next step.

Trigger installation only (not recommended): Install the P4 Code Review triggers package on the server hosting your P4 Server. This might be the server hosting P4 Code Review or elsewhere on your network.

- **If the server hosting your P4 Server cannot use packages**, for example when it is running Windows, you need to copy the appropriate P4 Code Review trigger script from `/opt/perforce/swarm/p4-bin/scripts` to the server hosting your P4 Server. The `swarm-trigger.pl` is for both Linux and Windows systems. Once copied, the trigger script needs to be configured. See ["Installing triggers" on page 187](#) for details.
- The package installs a config file at `/opt/perforce/etc/swarm-trigger.conf` that you will need to modify. See ["Installing triggers" on page 187](#) for more details on configuring that file.

Install the P4 Code Review triggers package on the server hosting your P4 Server

```
sudo apt-get install helix-swarm-triggers
```

5. **Optional:** Install the P4 Code Review optional package, on the server hosting P4 Code Review.

While not required, installing this package installs the dependencies required to use the ImageMagick and LibreOffice P4 Code Review modules. These modules provide previews of a variety of image and office documents.

```
sudo apt-get install helix-swarm-optional
```

6. Make your P4 Code Review installation more secure by applying recommendations related to HTTP and P4 Code Review implementation through security groups. See ["Secure your P4 Code Review installation"](#) below.
7. Complete the ["Post-installation configuration"](#) on [page 111](#) steps.

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers"](#) on [page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the "Reload Configuration button" on page 720.

4. Modify the P4 Server's `P4.Swarm.URL` property. For more information about P4 Server integration, see "Client integration" on page 573.

If your Apache server is listening on both HTTPS and HTTP in `perforce-swarm-site.conf` file, you must set the `auto_register_urlconfigurable` in the `p4` block to `false` and correctly configure the `P4.Swarm.URL` property .

If your Apache server is listening only on HTTPS and if the `auto_register_urlconfigurable` in the `p4` block is set to `true` (default value), an Apache restart will correct the property.

To get all your current values for `P4.Swarm.URL` property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the `P4.Swarm.URL` property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.

- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.
- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

Post-installation configuration

Once the helix-swarm package has been installed, additional configuration is required. Perform the following steps:

1. Use the P4 Code Review post-installation script to configure P4 Code Review on the server hosting P4 Code Review.

The P4 Code Review post-installation configuration script can be used in a few different ways. The steps below outline the most straightforward configuration using an interactive install, but you can review the options by running:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh -h
```

- If your P4 Server is configured for P4 AS, the P4 Server must be temporarily configured to allow fall-back to passwords while you establish a connection to the P4 Server. Run the following command on the P4 Server to enable fall-back to passwords:

```
p4 configure set auth.sso.allow.passwd=1
```
- To use P4 ServerExtensions, make sure that Extensions are installed and configured on your P4 Server before running the configuration script. For information about P4 ServerExtensions, see [p4 extension](#) in [P4 CLI Reference](#) and [Extensions overview](#) in [P4 Server Administration Documentation](#).

Run the interactive post-installation configuration script:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh
```

The configuration script displays the following summary:

```
-----
configure-swarm.sh: Thu Aug 26 11:29:49 PDT 2021: commencing configuration of
Swarm
Summary of arguments passed:
Interactive?  [yes]
Force?       [no]
P4PORT       [(not specified, will prompt)]
Swarm user   [(not specified, will prompt, will suggest swarm)]
Swarm password [(not specified, will prompt)]
Email host   [(not specified, will prompt)]
Create depot [(not specified, will prompt)]
Bypass Lock  [(not specified)]
Use Extensions? [(not specified, will prompt)]
Swarm host   [(not specified, will prompt, will suggest myhost)]
Swarm port   [(default (80))]
```

```
Swarm base URL  [(default (empty))]  
Create Swarm user? [yes]  
Super user      [(not specified, will prompt)]  
Super password  [(not specified, will prompt)]
```

2. Provide information to the configuration script.

After the summary, the configuration script prompts for the following information:

- a. Specify a value for `P4PORT` in the form: *my-helix-core-server:1666*

No P4PORT specified

P4 Code Review requires a connection to a Helix Core Server.
Please supply the P4PORT to connect to.

Helix Core Server address (P4PORT):

Specify the hostname and port for your P4 Server. If defined, the value for P4PORT is used as the default. The configuration script verifies that it can connect:

```
-response: [myp4host:1666]
```

```
Checking P4PORT [myp4host:1666]...
```

```
-P4 command line to use: [/opt/perforce/bin/p4 -p myp4host:1666]
```

```
Attempting connection to [myp4host:1666]...
```

```
-connection successful:
```

```
Server address: myp4host:1666
```

```
Server version: P4D/LINUX26X86_64/2021.1/2179737 (2021/04/24)
```

```
Server license: 10000 users (support ends 2022/05/16)
```

```
Server license-ip: 192.168.0.1
```

Important: If your P4 Server is deployed using the commit-edge architecture, ensure that the P4 Code Review port value points to the commit server.

For more information, see the [Commit-edge](#) chapter in the [P4 Server Administration Documentation](#).

- b. Specify the userid and password of a normal user with admin-level privileges in the P4 Server.

```
Checking Swarm user credentials...
```

```
No Swarm user specified
```

Swarm requires a Helix user account with 'admin' rights.
Please provide a username and password for this account.
If this account does not have 'admin' rights, it will
be set for this user.

Helix username for the Swarm user [*swarm*]:

Enter the userid. The default is *swarm*.

-response: [*swarm*]

Helix password or login ticket for the P4 Code Review user (typing hidden):

Restriction: To allow users to clean up reviews created by other users when the review is committed, the P4 Server user account must have *super* permissions.
See ["Review cleanup" on page 666](#).

Enter the login ticket, or password, for the userid.

Important: You must use a long-lived login ticket for the P4 Code Review user.

You can obtain a login ticket by running (in another shell):

```
p4 -p myp4host:1666 -u userid login -p
```

If the login ticket you provide expires in less than a year, you will receive a warning. If you receive this warning, ask your P4 Server administrator to make sure the *swarm* user is in a P4 Server group that has the Timeout field set to a year or more. See [p4 group](#) in the [P4 CLI Reference](#).

Checking Swarm user credentials...

-checking if user [*swarm*] exists in [*myp4host:1666*]...

-user exists

Obtaining Helix login ticket for [*swarm*] in [*myp4host:1666*]...

-login ticket obtained

Checking user [*swarm*]'s ticket against [*myp4host:1666*]...

-login ticket is good

Checking user [*swarm*] has at least access level [*admin*]...

-user has maximum access level [*admin*]

-user meets minimum access level [*admin*]

- c. Specify the hostname for the P4 Code Review UI.

swarm needs a distinct hostname that users can enter into their browsers to access Swarm. Ideally, this is a fully-qualified domain name, for example 'swarm.company.com', but it can be just a hostname, for example 'swarm'.

Whatever hostname you provide should be Swarm-specific and not shared with any other web service on this host.

Note that the hostname you specify typically requires configuration in your network's DNS service. If you are merely testing Swarm, you can add a hostname->IP mapping entry to your computer's hosts configuration.

Hostname for this Swarm server [myhost]:

d.

Tip: The default is the current hostname

The configuration script does not verify that the hostname actually works (DNS configuration may not exist yet).

e. Specify a mail relay host or leave empty to use your local mail handler.

P4 Code Review can use a mail relay host to send email notifications. Leave empty if

you want to use the local mail handler (for example Sendmail, Postfix etc), or enter a hostname (for example mx.yourdomain.com) to use a relay host.

Mail relay host:

Important: The configuration script does not verify that the mail relay host you provide actually accepts SMTP connections.

f. To enable file attachments on comments, P4 Code Review must have a depot location to save the attachment files in. P4 Code Review can create this depot and set the depot protections for you automatically or you can manually set it up later.

Important: You must be a user with *super* user permissions to create the .swarm depot and set protections.

- checking depot

P4 Code Review has the ability to store attachments against review comments.

To

do this, it needs to have a depot where they are stored. By default, this is //.swarm. We can create the depot for you automatically, and set the protections on it to the following:

```
list user * * -//.swarm/...
admin user swarm * //swarm/...
super user super * //swarm/...
```

If you want to enable attachments, and for the depot and protections to be set up for you, then say yes here. If you say no, then you can

still do this manually later.

Do you want to create a .swarm depot and set protections? (y/n) [n]

When prompted to automatically create the depot, select one of the following:

- Type **y** to automatically create the `.swarm` depot and set protections.
- Type **n** skip depot creation. You can create the depot and set the protections manually later if you want to, see ["Comment attachments" on page 575](#).

If you choose to automatically create the depot, the script will create the depot, set the protections, and report when completed.

Depot `//swarm` successfully created. Protections have been set so that only the 'swarm' and 'super' have permissions to access it directly.

- g. Configure the following to enable communication between P4 Code Review and P4 Server.
 - Configure the P4 Server to promote all shelved changes. For more information, see ["Configure the P4 Server to promote all shelved changes" on page 198](#).
 - Configure the `p4 key` access on P4 Server. For more information on `p4 key` and their access levels, see ["Hiding P4 Code Review storage from regular users" on page 226](#).
 - Allow P4 Code Review to work with 'exclusive open' files. For more information about exclusive locks, see ["Handling Exclusive Locks" on page 227](#).

Run the following to set these three configurations:

```
p4 configure dm.shelve.promote=1
p4 configure set dm.keys.hide=2
p4 configure set filetype.bypasslock=1
```

Tip: If this setting is not enabled in P4 Server, P4 Code Review will report exceptions when working with exclusively opened files similar to Cannot unshelve review (x). One or more files are exclusively open, and that you must have the `filetype.bypasslock` configurable enabled.

- h. P4 Code Review needs to know about some P4 Server events to operate correctly. Use P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about these events. P4 Server Extensions are easier to install and maintain than triggers.

Important: You must be a user with *super* user permissions to install and configure P4 Server Extensions.

Do you want to use Swarm's Helix Core server extension?

Configuring Server extensions requires super user access to the Helix Server.

If you install the Swarm server extension, do not install the Swarm triggers.

Server extensions are supported for:

* Linux: Helix server 19.2 and later.

* Windows: Helix server 21.2 and later.

Use server extensions?

When prompted to Use server Extensions, choose one of the following:

- **Recommended:** Type **y** to use P4 Server Extensions. The configuration script will try and install and configure the P4 Server extension.
- Type **n** to use P4 Server Triggers. Triggers must be installed manually, P4 Code Review will prompt you to do this when the P4 Code Review configuration script completes.

If you choose to install the P4 Server Extensions, the script will:

- check your P4 Server supports P4 Server Extensions
- check if P4 Server Extensions are installed and configured on your P4 Server. If they are, P4 Code Review does not need to do anything
- install and configure the P4 Server extension on your P4 Server

If any of these checks fail, P4 Code Review will not install the P4 Server extension and will report the issues on the configuration summary screen.

Once all of the information has been provided, the configuration script configures P4 Code Review. When it has completed the configuration, the configuration summary screen is displayed, for example:

```

.....
-restarting Apache...
-Apache restarted
configure-swarm.sh: Thu Aug 26 11:31:36 PDT 2021: completed configuration of
Helix Swarm

.....
::
:: Swarm is now configured and available at:
::
::   http://myhost/
::
:: You may login as the Swarm user [swarm] using the password
:: you specified.
::
:: Ensure that you have configured the Swarm hostname in your
:: network's DNS, or have added an IP address-to-hostname
:: mapping to your computer's hosts configuration so that you
:: can access Swarm.
::
:: Server side extensions are installed and configured
:: on your P4D server.
::
:: Documentation for optional post-install configuration, such as
:: configuring Swarm to use HTTPS, operate in a sub-folder, or on a
:: custom port, is available:
::
::
:: https://www.perforce.com/perforce/doc.current/manuals/swarm/setup.post.html
::
.....

```

If you have installed P4 Code Review on a host that does not provide other web services, you may wish to disable Apache's default site configuration. Doing so means that regardless of the hostname a user might use to reach the web server hosting P4 Code Review, P4 Code Review would be presented.

Be aware that disabling Apache's default site configuration could disable existing web services or content.

To disabling Apache's default site configuration on Ubuntu, run:

```
$ sudo a2dissite 000-default
```

To add Swarm as an HTML tab in the P4 Visual Client (P4V), see [HTML Tabs](#) section in the *P4VJS Developer Guide*.

3. The basic P4 Code Review configuration is now complete.

Tip: If your P4 Server is configured for P4 AS, you can force all of your users to authenticate via your Identity Provider (IdP) by disabling fall-back to passwords.

To disable fall-back to passwords on the P4 Server, run the following command:
`p4 configure set auth.sso.allow.passwd=0`

4. Do one of the following:
 - **If the P4 Server extension was installed by the configuration script:** Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).
 - **If the P4 Server extension was not installed, you must install triggers:** Configure P4 Server for P4 Code Review, see ["Installing triggers" on page 187](#).

Upgrading

See ["Upgrading a Ubuntu package installation" on page 231](#) for details.

Uninstall

1. Depending on how you have configured your Helix Core Server events, do one of the following:
 - Uninstall the P4 Server extension. To do this, run the following command on your P4 Server:
`p4 extension --delete Perforce::helix-swarm --yes`
 - Uninstall the P4 Code Review triggers. As a *super* user, run the `p4 triggers` command from your P4 Server and manually remove all the P4 Code Review trigger code lines.
2. Uninstall the P4 Code Review main package, the P4 Code Review triggers package, and the P4 Code Review optional package. The following example assumes that the packages are all installed on the same server:
`sudo apt-get remove helix-swarm helix-swarm-triggers helix-swarm-optional`
3. Remove the P4 Code Review trigger scripts from the server hosting your P4 Server.

Restriction: If you manually installed the trigger script, you must manually remove the script.

Install and configure P4 Code Review on RHEL

P4 Code Review packages can be used to install P4 Code Review on RHEL 8 and RHEL 9. Using packages greatly simplifies the installation, updating, and removal of software, because the tools that manage packages are aware of the dependencies for each package.

Do one of the following depending on your operating system:

- ["Install P4 Code Review on RHEL 8" below](#)
- ["Install P4 Code Review on RHEL 9" on page 125](#)

Related topics

- ["Post-installation configuration " on page 130](#)
- ["SELinux configuration" on page 137](#)
- ["Upgrading" on page 140](#)
- ["Uninstall" on page 140](#)

Install P4 Code Review on RHEL 8

Important:

- We recommend that the latest stable release is used. At the time of this P4 Code Review release, the latest stable RHEL 8 release is 8.8.
- P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

P4 Code Review can be connected to P4 Servers (P4D) and commit servers.

- To configure P4 Code Review to connect to more than one P4 Server (P4D), see ["Multiple P4 Server instances" on page 638](#).
- To configure P4 Code Review to connect to a P4 Server configured to use *commit-edge architecture*, see ["Commit-edge deployment" on page 582](#).

Important: Do not connect P4 Code Review to P4 Broker, P4 Proxy, an edge server, forwarding replica, or read-only replica servers.

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

Installing PHP on RHEL for P4 Code Review 2024.5 and later

P4 Code Review 2024.5 and later supports PHP 8.1, 8.2, or 8.3 . Depending on the PHP version available in your RHEL repositories, some adjustments may be required.

By default, RHEL has an older version of PHP (PHP 8.0, for example), unless you're using a Remi repository. If you're not using the Remi repository, you must manually install a supported version of PHP to meet the requirements for P4 Code Review 2024.5.

For example, install PHP 8.3 on RHEL:

1. Enable epel repo:

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

2. Enable remi repo:

```
dnf install -y https://rpms.remirepo.net/enterprise/remi-release-8.rpm
```

3. Enable PHP 8.3 stream:

```
dnf module enable php:remi-8.3 -y
```

4. Install PHP 8.3:

```
dnf module install -y php:remi-8.3
```

Installation

Important:

- Review the runtime dependencies before you install P4 Code Review, see ["Runtime dependencies" on page 91](#).
- Review the PHP requirements before you upgrade P4 Code Review, see ["PHP" on page 94](#).
- Review the P4 Server requirements before you install P4 Code Review, see ["P4 Server requirements" on page 98](#).
- Before continuing with the P4 Code Review install process, you must register the RHEL system to Red Hat using the Subscription Manager client.

1. Configure the Perforce package repository, on the server to host P4 Code Review and on the server hosting your P4 Server.

Important:

If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

As root, create the file `/etc/yum.repos.d/perforce.repo` with the following content:

```
[Perforce]
name=Perforce
baseurl=http://package.perforce.com/yum/rhel/8/x86_64/
enabled=1
gpgcheck=1
```

2. Import the Perforce package signing key, on the server to host P4 Code Review and the server hosting your P4 Server.

Important:

If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

Run the following command as root:

```
rpm --import https://package.perforce.com/perforce.pubkey
```

For information about how to verify the authenticity of the signing key, see [Perforce Packages](#).

3. Install the main P4 Code Review package on the server to host P4 Code Review (run these commands as root):

- a. Deploy the `epel-release-latest-8.noarch.rpm` repository configuration package to give P4 Code Review access to EPEL packages:

```
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

- b. Deploy the Remi repository configuration package to give P4 Code Review access to PHP 8.x (only required the first time you upgrade to PHP 8.x):

```
dnf install https://rpms.remirepo.net/enterprise/remi-release-8.rpm
```

Tip:

If you don't deploy the Remi repository, you will see dependency errors when you do the next steps.

- c. Install the `yum-utils` package to give access to the `yum-config-manager` command:

```
dnf install yum-utils
```

- d. Install the **Default/Single version** of PHP:

- i. Enable the module stream for PHP 8.2:

```
dnf module reset php
```

- ii. Install PHP 8.2:

```
dnf module install php:remi-8.2
```

- iii. Run an upgrade for PHP:

```
dnf update
```

- e. Install P4 Code Review and accept the prompts to import the GPG keys for Remi and EPEL when requested:

```
yum install helix-swarm
```

Important: When the P4 Code Review installation has completed, you are prompted to run the `configure-swarm.sh` post-installation script.

Do not run this script until you have completed the rest of these **Installation** instructions. Instructions for running the `configure-swarm.sh` post-installation script are in the **Post-Installation configuration** section referenced in the final step of the **Installation** instructions.

Note:

The firewall configuration may need to be adjusted to allow access to the web server.

```
sudo firewall-cmd --permanent --add-service=http
sudo systemctl reload firewalld
```

If you subsequently wish to enable HTTPS, run (as root):

```
sudo firewall-cmd --permanent --add-service=https
sudo systemctl reload firewalld
```

4. P4 Code Review needs to know about some P4 Server events to operate correctly. Use P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about these events. The P4 Server extension can be installed automatically by the P4 Code Review `configure-swarm.sh` post-installation script, but Triggers must be manually installed. To use P4 Server Extensions, ignore this step and skip to the next step.

Trigger installation only (not recommended): Install the P4 Code Review triggers package on the server hosting your P4 Server. This might be the server hosting P4 Code Review or elsewhere on your network.

Important:

- If the server hosting your P4 Server cannot use packages, for example when it is running Windows, you need to copy the appropriate P4 Code Review trigger script from `/opt/perforce/swarm/p4-bin/scripts` to the server hosting your P4 Server. The `swarm-trigger.pl` is for both Linux and Windows systems. Once copied, the trigger script needs to be configured. See ["Installing triggers" on page 187](#) for details.
- The package installs a config file at `/opt/perforce/etc/swarm-trigger.conf` that you will need to modify. See ["Installing triggers" on page 187](#) for more details on configuring that file.

Install the P4 Code Review triggers package on the server hosting your P4 Server (run this command as root):

```
yum install helix-swarm-triggers
```

5. **Optional:** Install the P4 Code Review optional package, on the server hosting P4 Code Review.

While not required, installing this package installs the dependencies required to use the ImageMagick and LibreOffice P4 Code Review modules. These modules provide previews of a variety of image and office documents. Run this command as root:

```
yum install helix-swarm-optional
```

6. Make your P4 Code Review installation more secure by applying recommendations related to HTTP and P4 Code Review implementation through security groups. See ["Secure your P4 Code Review installation" on the facing page](#).

7. Complete the ["Post-installation configuration "](#) on page 130 steps.

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers" on page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the "Reload Configuration button" on page 720.

4. Modify the P4 Server's `P4.Swarm.URL` property. For more information about P4 Server integration, see "Client integration" on page 573.

If your Apache server is listening on both HTTPS and HTTP in `performce-swarm-site.conf` file, you must set the `auto_register_urlconfigurable` in the `p4` block to false and correctly configure the `P4.Swarm.URL` property .

If your Apache server is listening only on HTTPS and if the `auto_register_urlconfigurable` in the `p4` block is set to true (default value), an Apache restart will correct the property.

To get all your current values for `P4.Swarm.URL` property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the `P4.Swarm.URL` property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.
- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.
- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

Install P4 Code Review on RHEL 9

Important:

- We recommend that the latest stable release is used. At the time of this P4 Code Review release, the latest stable RHEL 9 release is 9.5.
- P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

P4 Code Review can be connected to P4 Servers (P4D) and commit servers.

- To configure P4 Code Review to connect to more than one P4 Server (P4D), see ["Multiple P4 Server instances" on page 638](#).
- To configure P4 Code Review to connect to a P4 Server configured to use *commit-edge architecture*, see ["Commit-edge deployment" on page 582](#).

Important: Do not connect P4 Code Review to P4 Broker, P4 Proxy, an edge server, forwarding replica, or read-only replica servers.

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

Installing PHP on RHEL for P4 Code Review 2024.5 and later

P4 Code Review 2024.5 and later supports PHP 8.1, 8.2, or 8.3. Depending on the PHP version available in your RHEL repositories, some adjustments may be required.

By default, RHEL has an older version of PHP (PHP 8.0, for example), unless you're using a Remi repository. If you're not using the Remi repository, you must manually install a supported version of PHP to meet the requirements for P4 Code Review 2024.5.

For example, install PHP 8.3 on RHEL:

1. Enable epel repo:

```
dnf install -y https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm
```

2. Enable remi repo:

```
dnf install -y https://rpms.remirepo.net/enterprise/remi-release-9.rpm
```

3. Enable PHP 8.3 stream:

```
dnf module enable php:remi-8.3 -y
```

4. Install PHP 8.3:

```
dnf module install -y php:remi-8.3
```

Installation

Important:

- Review the runtime dependencies before you install P4 Code Review, see ["Runtime dependencies" on page 91](#).
- Review the PHP requirements before you upgrade P4 Code Review, see ["PHP" on page 94](#).
- Review the P4 Server requirements before you install P4 Code Review, see ["P4 Server requirements" on page 98](#).
- Before continuing with the P4 Code Review install process, you must register the RHEL system to Red Hat using the Subscription Manager client.

1. Configure the Perforce package repository, on the server to host P4 Code Review and on the server hosting your P4 Server.

Important:

If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

As root, create the file `/etc/yum.repos.d/perforce.repo` with the following content:

```
[Perforce]
name=Perforce
baseurl=http://package.perforce.com/yum/rhel/9/x86_64/
enabled=1
gpgcheck=1
```

2. Import the Perforce package signing key, on the server to host P4 Code Review and the server hosting your P4 Server.

Important:

If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

Run the following command as root:

```
rpm --import https://package.perforce.com/perforce.pubkey
```

For information about how to verify the authenticity of the signing key, see [Perforce Packages](#).

3. Install the main P4 Code Review package on the server to host P4 Code Review (run these commands as root):
 - a. Run an upgrade for PHP, this will also upgrade the P4 Code Review packages:

```
dnf update
```
 - b. Install P4 Code Review.

```
yum install helix-swarm
```

Important: When the P4 Code Review installation has completed, you are prompted to run the `configure-swarm.sh` post-installation script. Do not run this script until you have completed the rest of these **Installation** instructions. Instructions for running the `configure-swarm.sh` post-installation script are in the **Post-Installation configuration** section referenced in the final step of the **Installation** instructions.

Note:

The firewall configuration may need to be adjusted to allow access to the web server.

```
sudo firewall-cmd --permanent --add-service=http
sudo systemctl reload firewalld
```

If you subsequently wish to enable HTTPS, run (as root):

```
sudo firewall-cmd --permanent --add-service=https
sudo systemctl reload firewalld
```

4. P4 Code Review needs to know about some P4 Server events to operate correctly. Use P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about these events. The P4 Server extension can be installed automatically by the P4 Code Review `configure-swarm.sh` post-installation script, but Triggers must be manually installed. To use P4 Server Extensions, ignore this step and skip to the next step.

Trigger installation only (not recommended): Install the P4 Code Review triggers package on the server hosting your P4 Server. This might be the server hosting P4 Code Review or elsewhere on your network.

Important:

- If the server hosting your P4 Server cannot use packages, for example when it is running Windows, you need to copy the appropriate P4 Code Review trigger script from `/opt/perforce/swarm/p4-bin/scripts` to the server hosting your P4 Server. The `swarm-trigger.pl` is for both Linux and Windows systems. Once copied, the trigger script needs to be configured. See ["Installing triggers" on page 187](#) for details.
- The package installs a config file at `/opt/perforce/etc/swarm-trigger.conf` that you will need to modify. See ["Installing triggers" on page 187](#) for more details on configuring that file.

Install the P4 Code Review triggers package on the server hosting your P4 Server (run this command as root):

```
yum install helix-swarm-triggers
```

5. **Optional:** Install the P4 Code Review optional package, on the server hosting P4 Code Review.

While not required, installing this package installs the dependencies required to use the ImageMagick and LibreOffice P4 Code Review modules. These modules provide previews of a variety of image and office documents. Run this command as root:

```
yum install helix-swarm-optional
```

6. Make your P4 Code Review installation more secure by applying recommendations related to HTTP and P4 Code Review implementation through security groups. See ["Secure your P4 Code Review installation "](#) below.
7. Complete the ["Post-installation configuration "](#) on page 130 steps.

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers" on page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the "Reload Configuration button" on page 720.

4. Modify the P4 Server's `P4.Swarm.URL` property. For more information about P4 Server integration, see "Client integration" on page 573.

If your Apache server is listening on both HTTPS and HTTP in `perforce-swarm-site.conf` file, you must set the `auto_register_url` configurable in the `p4` block to false and correctly configure the `P4.Swarm.URL` property.

If your Apache server is listening only on HTTPS and if the `auto_register_url` configurable in the `p4` block is set to true (default value), an Apache restart will correct the property.

To get all your current values for `P4.Swarm.URL` property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the `P4.Swarm.URL` property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.

- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.
- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

Post-installation configuration

Important:

The following notes are applicable for RHEL 8 and RHEL 9:

- **P4 Code Review 2020.2 and later:** these versions of P4 Code Review uses the Remi repository for RHEL 8 and RHEL 9. This provides PHP 8.x installed in the standard file system structure. This means that the old httpd24-httpd version of Apache is no longer needed, and the standard system version of Apache is being used again.
The SCL Apache site configuration file was installed at this location for P4 Code Review 2019.1 to 2020.1:
`/opt/rh/httpd24/root/etc/httpd/conf.d/perforce-swarm-site.conf`
If this exists when P4 Code Review is upgraded to 2020.2 and later, this file is copied to `/etc/httpd/conf.d/perforce-swarm-site.conf` if there is no file at the destination. It is also re-written to change references from `/var/log/httpd24` to `/var/log/httpd`
If a site configuration file for P4 Code Review already exists in `/etc/httpd`, the copy and re-write is not performed.
After upgrade, httpd24-httpd is disabled.
- To avoid seeing the Apache HTTP server Linux test page when you start the Apache server, comment out the content of the `welcome.conf` file located in the `/etc/httpd/conf.d/` directory.
- To avoid loading the Apache HTTP server example configuration instead of the P4 Code Review configuration when the Apache server starts, rename the `autoindex.conf` file located in the `/etc/httpd/conf.d/` directory to `z-autoindex.conf` or similar. This is required because Apache runs the first conf file it finds in the `/etc/httpd/conf.d/` directory (alphabetical order) and that must be the `perforce-swarm-site.conf` file.

Once the `helix-swarm` package has been installed, additional configuration is required. Perform the following steps:

1. Use the P4 Code Review post-installation script to configure P4 Code Review on the server hosting P4 Code Review.

The P4 Code Review post-installation configuration script can be used in a few different ways. The steps below outline the most straightforward configuration using an interactive install, but you can review the options by running:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh -h
```

- If your P4 Server is configured for P4 AS, the P4 Server must be temporarily configured to allow fall-back to passwords while you establish a connection to the P4 Server. Run the following command on the P4 Server to enable fall-back to passwords:
`p4 configure set auth.sso.allow.passwd=1`
- To use P4 ServerExtensions, make sure that Extensions are installed and configured on your P4 Server before running the configuration script. For information about P4 ServerExtensions, see [p4 extension](#) in [P4 CLI Reference](#) and [Extensions overview](#) in [P4 Server Administration Documentation](#).

Run the interactive post-installation configuration script:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh
```

The configuration script displays the following summary:

```
-----
configure-swarm.sh: Thu Aug 26 11:29:49 PDT 2021: commencing configuration of
Swarm
Summary of arguments passed:
Interactive?    [yes]
Force?         [no]
P4PORT         [(not specified, will prompt)]
Swarm user     [(not specified, will prompt, will suggest swarm)]
Swarm password [(not specified, will prompt)]
Email host     [(not specified, will prompt)]
Create depot   [(not specified, will prompt)]
Bypass Lock    [(not specified)]
Use Extensions? [(not specified, will prompt)]
Swarm host     [(not specified, will prompt, will suggest myhost)]
Swarm port     [(default (80))]
Swarm base URL [(default (empty))]
Create Swarm user? [yes]
Super user     [(not specified, will prompt)]
Super password [(not specified, will prompt)]
```

2. Provide information to the configuration script.

After the summary, the configuration script prompts for the following information:

- a. Specify a value for `P4PORT` in the form: *my-helix-core-server:1666*

No P4PORT specified

P4 Code Review requires a connection to a Helix Core Server.
Please supply the P4PORT to connect to.

Helix Core Server address (P4PORT):

Specify the hostname and port for your P4 Server. If defined, the value for P4PORT is used as the default. The configuration script verifies that it can connect:

```
-response: [myp4host:1666]
```

```
Checking P4PORT [myp4host:1666]...
```

```
-P4 command line to use: [/opt/perforce/bin/p4 -p myp4host:1666]
```

```
Attempting connection to [myp4host:1666]...
```

```
-connection successful:
```

```
Server address: myp4host:1666
```

```
Server version: P4D/LINUX26X86_64/2021.1/2179737 (2021/04/24)
```

```
Server license: 10000 users (support ends 2022/05/16)
```

```
Server license-ip: 192.168.0.1
```

Important: If your P4 Server is deployed using the commit-edge architecture, ensure that the P4 Code Review port value points to the commit server.

For more information, see the [Commit-edge](#) chapter in the [P4 Server Administration Documentation](#).

- b. Specify the userid and password of a normal user with admin-level privileges in the P4 Server.

```
Checking Swarm user credentials...
```

```
No Swarm user specified
```

```
Swarm requires a Helix user account with 'admin' rights.
```

```
Please provide a username and password for this account.
```

```
If this account does not have 'admin' rights, it will  
be set for this user.
```

```
Helix username for the Swarm user [swarm]:
```

Enter the userid. The default is *swarm*.

```
-response: [swarm]
```

```
Helix password or login ticket for the P4 Code Review user (typing hidden):
```

Restriction: To allow users to clean up reviews created by other users when the review is committed, the P4 Server user account must have *super* permissions.

See ["Review cleanup" on page 666](#).

Enter the login ticket, or password, for the userid.

Important: You must use a long-lived login ticket for the P4 Code Review user.

You can obtain a login ticket by running (in another shell):

```
p4 -p myp4host:1666 -u userid login -p
```

If the login ticket you provide expires in less than a year, you will receive a warning. If you receive this warning, ask your P4 Server administrator to make sure the swarm user is in a P4 Server group that has the `Timeout` field set to a year or more. See [p4 group](#) in the [P4 CLI Reference](#).

Checking Swarm user credentials...

-checking if user [swarm] exists in [myp4host:1666]...

-user exists

Obtaining Helix login ticket for [swarm] in [myp4host:1666]...

-login ticket obtained

Checking user [swarm]'s ticket against [myp4host:1666]...

-login ticket is good

Checking user [swarm] has at least access level [admin]...

-user has maximum access level [admin]

-user meets minimum access level [admin]

- c. Specify the hostname for the P4 Code Review UI.

swarm needs a distinct hostname that users can enter into their browsers to access Swarm. Ideally, this is a fully-qualified domain name, for example 'swarm.company.com', but it can be just a hostname, for example 'swarm'.

Whatever hostname you provide should be Swarm-specific and not shared with any other web service on this host.

Note that the hostname you specify typically requires configuration in your network's DNS service. If you are merely testing Swarm, you can add a hostname->IP mapping entry to your computer's hosts configuration.

Hostname for this Swarm server [myhost]:

- d.

Tip: The default is the current hostname

The configuration script does not verify that the hostname actually works (DNS configuration may not exist yet).

- e. Specify a mail relay host or leave empty to use your local mail handler.

P4 Code Review can use a mail relay host to send email notifications. Leave empty if you want to use the local mail handler (for example Sendmail, Postfix etc),

or enter a hostname (for example mx.yourdomain.com) to use a relay host.

Mail relay host:

Important: The configuration script does not verify that the mail relay host you provide actually accepts SMTP connections.

- f. To enable file attachments on comments, P4 Code Review must have a depot location to save the attachment files in. P4 Code Review can create this depot and set the depot protections for you automatically or you can manually set it up later.

Important: You must be a user with *super* user permissions to create the `.swarm` depot and set protections.

- checking depot

P4 Code Review has the ability to store attachments against review comments. To do this, it needs to have a depot where they are stored. By default, this is `//.swarm`. We can create the depot for you automatically, and set the protections on it to the following:

```
list user * * -//.swarm/...
admin user swarm * //swarm/...
super user super * //swarm/...
```

If you want to enable attachments, and for the depot and protections to be set up for you, then say yes here. If you say no, then you can still do this manually later.

Do you want to create a .swarm depot and set protections? (y/n) [n]

When prompted to automatically create the depot, select one of the following:

- Type `y` to automatically create the `.swarm` depot and set protections.
- Type `n` skip depot creation. You can create the depot and set the protections manually later if you want to, see ["Comment attachments" on page 575](#).

If you choose to automatically create the depot, the script will create the depot, set the protections, and report when completed.

Depot `//.swarm` successfully created. Protections have been set so that only the 'swarm' and 'super' have permissions to access it directly.

- g. Configure the following to enable communication between P4 Code Review and P4 Server.

- Configure the P4 Server to promote all shelved changes. For more information, see ["Configure the P4 Server to promote all shelved changes" on page 198](#).
- Configure the p4 key access on P4 Server. For more information on p4 key and their access levels, see ["Hiding P4 Code Review storage from regular users" on page 226](#).
- Allow P4 Code Review to work with 'exclusive open' files. For more information about exclusive locks, see ["Handling Exclusive Locks" on page 227](#).

Run the following to set these three configurations:

```
p4 configure dm.shelve.promote=1
p4 configure set dm.keys.hide=2
p4 configure set filetype.bypasslock=1
```

Tip: If this setting is not enabled in P4 Server, P4 Code Review will report exceptions when working with exclusively opened files similar to Cannot unshelve review (x). One or more files are exclusively open, and that you must have the filetype.bypasslock configurable enabled.

- h. P4 Code Review needs to know about some P4 Server events to operate correctly. Use P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about these events. P4 Server Extensions are easier to install and maintain than triggers.

Important: You must be a user with *super* user permissions to install and configure P4 Server Extensions.

Do you want to use Swarm's Helix Core server extension?

Configuring Server extensions requires super user access to the Helix Server.

If you install the Swarm server extension, do not install the Swarm triggers.

Server extensions are supported for:

* Linux: Helix server 19.2 and later.

* Windows: Helix server 21.2 and later.

Use server extensions?

When prompted to Use server Extensions, choose one of the following:

- **Recommended:** Type y to use P4 Server Extensions. The configuration script will try and install and configure the P4 Server extension.
- Type n to use P4 Server Triggers. Triggers must be installed manually, P4 Code Review will prompt you to do this when the P4 Code Review configuration script completes.

If you choose to install the P4 Server Extensions, the script will:

- check your P4 Server supports P4 Server Extensions
- check if P4 Server Extensions are installed and configured on your P4 Server. If they are, P4 Code Review does not need to do anything
- install and configure the P4 Server extension on your P4 Server

If any of these checks fail, P4 Code Review will not install the P4 Server extension and will report the issues on the configuration summary screen.

Once all of the information has been provided, the configuration script configures P4 Code Review. When it has completed the configuration, the configuration summary screen is displayed, for example:

```
.....
-restarting Apache...
-Apache restarted
configure-swarm.sh: Thu Aug 26 11:31:36 PDT 2021: completed configuration of
Helix Swarm

.....
::
:: Swarm is now configured and available at:
::
::   http://myhost/
::
:: You may login as the Swarm user [swarm] using the password
:: you specified.
::
:: Ensure that you have configured the Swarm hostname in your
:: network's DNS, or have added an IP address-to-hostname
:: mapping to your computer's hosts configuration so that you
:: can access Swarm.
::
:: Server side extensions are installed and configured
:: on your P4D server.
::
:: Documentation for optional post-install configuration, such as
:: configuring Swarm to use HTTPS, operate in a sub-folder, or on a
:: custom port, is available:
::
::
:: https://www.perforce.com/perforce/doc.current/manuals/swarm/setup.post.html
::
::
.....
```

Note:

If you have installed P4 Code Review on a host that does not provide other web services, you may wish to disable Apache's default site configuration. Doing so means that regardless of the hostname a user might use to reach the web server hosting P4 Code Review, P4 Code Review would be presented.

Be aware that disabling Apache's default site configuration could disable existing web services or content.

Disabling Apache's default site configuration on Ubuntu hosts is easy. Run:

```
$ sudo a2dissite 000-default
```

For non-standard Apache installations, you would need to manually adjust the Apache configuration. Such changes require familiarity with Apache configuration; for more details, see [Apache HTTP server project](#).

To add Swarm as an HTML tab in the P4 Visual Client (P4V), see [HTML Tabs](#) section in the *P4VJS Developer Guide*.

3. Configure SELinux for P4 Code Review, see "[SELinux configuration](#)" below.
4. The basic P4 Code Review configuration is now complete.

Important:

If your P4 Server is configured for P4 AS, you can force all of your users to authenticate via your Identity Provider (IdP) by disabling fall-back to passwords. To disable fall-back to passwords on the P4 Server, run the following command:

```
p4 configure set auth.sso.allow.passwd=0
```

5. Do one of the following:
 - **If the P4 Server extension was installed by the configuration script:** Review the post-install configuration options to customize your P4 Code Review installation, see "[Post-install configuration options](#)" on page 214.
 - **If the P4 Server extension was not installed, you must install triggers:** Configure P4 Server for P4 Code Review, see "[Installing triggers](#)" on page 187.

SELinux configuration

P4 Code Review supports SELinux which is an advanced access control mechanism that improves security for Linux distributions.

SELinux operates in one of three modes:

- **enforcing:** this mode blocks and logs any actions that do not match the defined security policy. This is the default mode for SELinux.
- **permissive:** this mode logs actions that do not match the defined security policy but these actions are not blocked.
- **disabled:** in this mode SELinux is off, actions are not blocked and are not logged.

Tip:

To check the mode SELinux is operating in, view the `/etc/selinux/config` file with `vi` or a similar editor:

```
root $ vi /etc/selinux/config
```

SELinux must be configured to enable it to work correctly with P4 Code Review, these configuration steps are shown below.

Note:

You must complete the P4 Code Review package [Installation](#) steps, and the ["Post-installation configuration"](#) on [page 130](#) steps before configuring SELinux.

Configure SELinux to enforcing mode

Run the following commands as root:

1. Install the package that contains the **semanage** configuration tool, this is used to configure SELinux:
Follow the instructions for your OS distribution:
 - **RHEL 8:** Install the `policycoreutils-python-utils` package:

```
root $ yum install policycoreutils-python-utils
```
 - **RHEL 9:** Install the `policycoreutils-python-utils` package:

```
root $ yum install policycoreutils-python-utils
```
2. Check the current SELinux mode:

```
root $ getenforce
```
3. SELinux will report its mode as; `enforcing`, `permissive`, or `disabled`.
 - a. If the mode is not set correctly edit the `/etc/selinux/config` file with `vi` or a similar editor.

```
root $ vi /etc/selinux/config
```
 - b. Edit the config file so that `SELinux=` is set to `enforcing`.
 - c. Save the config file.
 - d. Reboot the server to complete the SELinux mode change.
4. Allow content in `/opt/perforce/swarm` to be read and written by the `httpd` process:

```
root $ semanage fcontext -a -t httpd_sys_rw_content_t "/opt/perforce/swarm(/.*)?"
```

```
root $ restorecon -R /opt/perforce/swarm
```
5. Allow the `httpd` process to connect to other networked services, for example P4D and Redis:

```
root $ setsebool -P httpd_can_network_connect 1
```
6. Allow comment attachment thumbnails to be created:

```
root $ setsebool -P httpd_tmp_exec 1
```
7. Allow the files in `p4-bin` to be executed by the `httpd` process:

```
root $ semanage fcontext -a -t httpd_sys_script_exec_t '/opt/perforce/swarm/p4-bin
(/.*)?'
root $ restorecon -R -v /opt/perforce/swarm/p4-bin
```

8. Remove the executable constraints on Redis, allowing it to be started by systemd at boot time:

```
root $ semanage fcontext -a -t bin_t /opt/perforce/swarm/sbin/redis-server-swarm
root $ restorecon -v /opt/perforce/swarm/sbin/redis-server-swarm
```

9. Restart the system:

```
root $ systemctl restart httpd
```

10. Check that you can log in to P4 Code Review.

11. **Only if required:** Relabel your filesystem, see note before relabeling:

Important:

Relabeling your file system can be a time consuming process, it is recommended that you only do this if you need to. This depends entirely on your SELinux setup, Perforce cannot give you advice on this.

```
root $ touch /.autorelabel
```

12. Reboot the server.
13. Check that you can log in to P4 Code Review.
14. SELinux is now configured for P4 Code Review.

Note:

If you can not log in to P4 Code Review it is possible that SELinux is blocking P4 Code Review because its configuration is incorrect. You will need to troubleshoot the SELinux configuration to find any issues.

Install the `setroubleshoot` package, this contains `sealert` which is used when troubleshooting SELinux:

```
root $ yum install setroubleshoot
```

`sealert` helps you to interpret the contents of the `audit.log`. Run the following command:

```
root $ sealert -a /var/log/audit/audit.log
```

Error message: If you see an error message with a title similar to the message below, it may be because you are running RHEL on a Virtual Machine (VM).

```
root $ SELinux is preventing /usr/sbin/ldconfig from write access on the directory etc.
```

Install `open-vm-tools` on the VM and reboot the VM.

```
root $ yum install open-vm-tools
```

Configure SELinux permissive or disabled mode

Run the following as root:

1. Check the current SELinux mode:

```
root $ getenforce
```
2. SELinux will report its mode as; `enforcing`, `permissive`, or `disabled`.
 - a. If the mode is not set correctly edit the `/etc/selinux/config` file with `vi` or a similar editor.

```
root $ vi /etc/selinux/config
```
 - b. Edit the config file so that `SELinux=` is set to `permissive` or `disabled` as required.
 - c. Save the config file.
 - d. Reboot the server to complete the SELinux mode change.
3. Check that you can log in to P4 Code Review.
4. SELinux is now configured for P4 Code Review.

Upgrading

See ["Upgrading a RHEL package installation" on page 243](#) for details.

Uninstall

1. Depending on how you have configured your Helix Core Server events, do one of the following:
 - Uninstall the P4 Server extension. To do this, run the following command on your P4 Server:

```
p4 extension --delete Perforce::helix-swarm --yes
```
 - Uninstall the P4 Code Review triggers. As a *super* user, run the `p4 triggers` command from your P4 Server and manually remove all the P4 Code Review trigger code lines.
2. Uninstall the P4 Code Review main package, the P4 Code Review triggers package, and the P4 Code Review optional package. The following example assumes that the packages are all installed on the same server (run the command as root):

```
yum remove helix-swarm helix-swarm-triggers helix-swarm-optional
```
3. Remove the P4 Code Review trigger scripts from the server hosting your P4 Server.

Important:

If you manually installed the trigger script, perhaps because the server hosting your P4 Server cannot use packages (e.g. Windows), manually remove the script.

Install and configure P4 Code Review on Amazon Linux 2

P4 Code Review packages can be used to install P4 Code Review on Amazon Linux 2. Using packages greatly simplifies the installation, updating, and removal of software, because the tools that manage packages are aware of the dependencies for each package.

Important:

We recommend that the latest stable release of Amazon Linux 2 is used.

Important:

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

P4 Code Review can be connected to P4 Servers (P4D) and commit servers.

- To configure P4 Code Review to connect to more than one P4 Server (P4D), see ["Multiple P4 Server instances" on page 638](#).
- To configure P4 Code Review to connect to a P4 Server configured to use *commit-edge architecture*, see ["Commit-edge deployment" on page 582](#).

Important: Do not connect P4 Code Review to P4 Broker, P4 Proxy, an edge server, forwarding replica, or read-only replica servers.

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

Installation

Important:

- Review the runtime dependencies before you install P4 Code Review, see ["Runtime dependencies" on page 91](#).
- Review the PHP requirements before you upgrade P4 Code Review, see ["PHP" on page 94](#).
P4 Code Review no longer supports PHP 8.0 version.
- Review the P4 Server requirements before you install P4 Code Review, see ["P4 Server requirements" on page 98](#).

1. Configure the Perforce package repository, on the server to host P4 Code Review and on the server hosting your P4 Server.

Important:

If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

As root, create the file `/etc/yum.repos.d/perforce.repo` with the following content:

```
[Perforce]
name=Perforce
baseurl=http://package.perforce.com/yum/rhel/8/x86_64/
enabled=1
gpgcheck=1
```

2. Import the Perforce package signing key, on the server to host P4 Code Review and the server hosting your P4 Server.

Important:

If the server hosting your P4 Server cannot use packages, for example when it is running Windows, skip this step on that server.

Run the following command as root:

```
rpm --import https://package.perforce.com/perforce.pubkey
```

For information about how to verify the authenticity of the signing key, see [Perforce Packages](#).

3. Install the main P4 Code Review package on the server to host P4 Code Review.

Run the following commands as root:

- a. Deploy the EPEL repository configuration package to give P4 Code Review access to the EPEL packages.

```
yum install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

- b. Install the yum-utils package to give access to the yum-config-manager command:

```
yum install yum-utils
```

- c. Install the Amazon Extras repository if it is not already installed:

```
yum install -y amazon-linux-extras
```

- d. Enable PHP 8.3 from Amazon Linux Extras:

```
amazon-linux-extras enable php8.3
```

- e. Install P4 Code Review and accept the prompts to import the GPG keys for EPEL when requested:

```
yum install helix-swarm
```

Important: When the P4 Code Review installation has completed, you are prompted to run the configure-swarm.sh post-installation script.

Do not run this script until you have completed the rest of these **Installation** instructions. Instructions for running the configure-swarm.sh post-installation script are in the **Post-Installation configuration** section referenced in the final step of the **Installation** instructions.

Note:

The firewall configuration may need to be adjusted to allow access to the web server.

```
sudo firewall-cmd --permanent --add-service=http
sudo systemctl reload firewalld
```

If you subsequently wish to enable HTTPS, run (as root):

```
sudo firewall-cmd --permanent --add-service=https
sudo systemctl reload firewalld
```

4. P4 Code Review needs to know about some P4 Server events to operate correctly. Use P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about these events. The P4 Server extension can be installed automatically by the P4 Code Review `configure-swarm.sh` post-installation script, but Triggers must be manually installed. To use P4 Server Extensions, ignore this step and skip to the next step.

Trigger installation only (not recommended): Install the P4 Code Review triggers package on the server hosting your P4 Server. This might be the server hosting P4 Code Review or elsewhere on your network.

Important:

If the server hosting your P4 Server cannot use packages, for example when it is running Windows, you need to copy the appropriate P4 Code Review trigger script from `/opt/perforce/swarm/p4-bin/scripts` to the server hosting your P4 Server. The `swarm-trigger.pl` is for both Linux and Windows systems. Once copied, the trigger script needs to be configured. See ["Installing triggers" on page 187](#) for details.

Run this command as root:

```
yum install helix-swarm-triggers
```

Important:

The package installs a config file at `/opt/perforce/etc/swarm-trigger.conf` that you will need to modify. See ["Installing triggers" on page 187](#) for more details on configuring that file.

5. **Optional:** While not required, installing the optional P4 Code Review package installs the dependencies required to use the LibreOffice P4 Code Review module. This enables P4 Code Review to preview office documents.
 - a. Enable LibreOffice in Amazon Linux Extras with:


```
amazon-linux-extras enable libreoffice
```
 - b. Install the P4 Code Review optional package, on the server hosting P4 Code Review (run the command as root):


```
yum install helix-swarm-optional
```

6. Make your P4 Code Review installation more secure by applying recommendations related to HTTP and P4 Code Review implementation through security groups. See ["Secure your P4 Code Review installation" below](#).
7. Complete the ["Post-installation configuration" on page 146](#) steps.

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers" on page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the "Reload Configuration button" on page 720.

4. Modify the P4 Server's `P4.Swarm.URL` property. For more information about P4 Server integration, see "Client integration" on page 573.

If your Apache server is listening on both HTTPS and HTTP in `performce-swarm-site.conf` file, you must set the `auto_register_urlconfigurable` in the `p4` block to false and correctly configure the `P4.Swarm.URL` property .

If your Apache server is listening only on HTTPS and if the `auto_register_urlconfigurable` in the `p4` block is set to true (default value), an Apache restart will correct the property.

To get all your current values for `P4.Swarm.URL` property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the `P4.Swarm.URL` property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.
- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.
- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

Post-installation configuration

Important:

- To avoid seeing the Apache HTTP server Linux test page when you start the Apache server, comment out the content of the welcome.conf file located in the /etc/httpd/conf.d/ directory.
- To avoid loading the Apache HTTP server example configuration instead of the P4 Code Review configuration when the Apache server starts, rename the autoindex.conf file located in the /etc/httpd/conf.d/ directory to z-autoindex.conf or similar. This is required because Apache runs the first conf file it finds in the /etc/httpd/conf.d/ directory (alphabetical order) and that must be the perforce-swarm-site.conf file.

Once the helix-swarm package has been installed, additional configuration is required. Perform the following steps:

1. Use the P4 Code Review post-installation script to configure P4 Code Review on the server hosting P4 Code Review.

The P4 Code Review post-installation configuration script can be used in a few different ways. The steps below outline the most straightforward configuration using an interactive install, but you can review the options by running:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh -h
```

- If your P4 Server is configured for P4 AS, the P4 Server must be temporarily configured to allow fall-back to passwords while you establish a connection to the P4 Server. Run the following command on the P4 Server to enable fall-back to passwords:

```
p4 configure set auth.sso.allow.passwd=1
```
- To use P4 ServerExtensions, make sure that Extensions are installed and configured on your P4 Server before running the configuration script. For information about P4 ServerExtensions, see [p4 extension](#) in [P4 CLI Reference](#) and [Extensions overview](#) in [P4 Server Administration Documentation](#).

Run the interactive post-installation configuration script:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh
```

The configuration script displays the following summary:

```
-----
configure-swarm.sh: Thu Aug 26 11:29:49 PDT 2021: commencing configuration of
Swarm
Summary of arguments passed:
Interactive?  [yes]
Force?       [no]
P4PORT       [(not specified, will prompt)]
Swarm user   [(not specified, will prompt, will suggest swarm)]
Swarm password [(not specified, will prompt)]
Email host   [(not specified, will prompt)]
```

```

Create depot    [(not specified, will prompt)]
Bypass Lock     [(not specified)]
Use Extensions? [(not specified, will prompt)]
Swarm host      [(not specified, will prompt, will suggest myhost)]
Swarm port      [(default (80))]
Swarm base URL  [(default (empty))]
Create Swarm user? [yes]
Super user      [(not specified, will prompt)]
Super password  [(not specified, will prompt)]

```

2. Provide information to the configuration script.

After the summary, the configuration script prompts for the following information:

- a. Specify a value for **P4PORT** in the form: *my-helix-core-server:1666*

No P4PORT specified

P4 Code Review requires a connection to a Helix Core Server.
Please supply the P4PORT to connect to.

Helix Core Server address (P4PORT):

Specify the hostname and port for your P4 Server. If defined, the value for P4PORT is used as the default. The configuration script verifies that it can connect:

-response: [myp4host:1666]

Checking P4PORT [myp4host:1666]...

-P4 command line to use: [/opt/perforce/bin/p4 -p myp4host:1666]

Attempting connection to [myp4host:1666]...

-connection successful:

Server address: myp4host:1666

Server version: P4D/LINUX26X86_64/2021.1/2179737 (2021/04/24)

Server license: 10000 users (support ends 2022/05/16)

Server license-ip: 192.168.0.1

Important: If your P4 Server is deployed using the commit-edge architecture, ensure that the P4 Code Review port value points to the commit server.

For more information, see the [Commit-edge](#) chapter in the [P4 Server Administration Documentation](#).

- b. Specify the userid and password of a normal user with admin-level privileges in the P4 Server.

Checking Swarm user credentials...

No Swarm user specified

Swarm requires a Helix user account with 'admin' rights.
Please provide a username and password for this account.
If this account does not have 'admin' rights, it will
be set for this user.

Helix username for the Swarm user [*swarm*]:

Enter the userid. The default is *swarm*.

-response: [*swarm*]

Helix password or login ticket for the P4 Code Review user (typing hidden):

Restriction: To allow users to clean up reviews created by other users when the review is committed, the P4 Server user account must have *super* permissions.
See ["Review cleanup" on page 666](#).

Enter the login ticket, or password, for the userid.

Important: You must use a long-lived login ticket for the P4 Code Review user.

You can obtain a login ticket by running (in another shell):

```
p4 -p myp4host:1666 -u userid login -p
```

If the login ticket you provide expires in less than a year, you will receive a warning. If you receive this warning, ask your P4 Server administrator to make sure the *swarm* user is in a P4 Server group that has the `Timeout` field set to a year or more. See [p4 group](#) in the [P4 CLI Reference](#).

Checking Swarm user credentials...

-checking if user [*swarm*] exists in [*myp4host:1666*]...

-user exists

Obtaining Helix login ticket for [*swarm*] in [*myp4host:1666*]...

-login ticket obtained

Checking user [*swarm*]'s ticket against [*myp4host:1666*]...

-login ticket is good

Checking user [*swarm*] has at least access level [*admin*]...

-user has maximum access level [*admin*]

-user meets minimum access level [*admin*]

- c. Specify the hostname for the P4 Code Review UI.

swarm needs a distinct hostname that users can enter into their browsers to access Swarm. Ideally, this is a fully-qualified domain name, for example 'swarm.company.com', but it can be just a hostname, for example 'swarm'.

Whatever hostname you provide should be Swarm-specific and not shared with any other web service on this host.

Note that the hostname you specify typically requires configuration in your network's DNS service. If you are merely testing Swarm, you can add a hostname->IP mapping entry to your computer's hosts configuration.

Hostname for this Swarm server [myhost]:

d.

Tip: The default is the current hostname

The configuration script does not verify that the hostname actually works (DNS configuration may not exist yet).

e. Specify a mail relay host or leave empty to use your local mail handler.

P4 Code Review can use a mail relay host to send email notifications. Leave empty if you want to use the local mail handler (for example Sendmail, Postfix etc), or enter a hostname (for example mx.yourdomain.com) to use a relay host.

Mail relay host:

Important: The configuration script does not verify that the mail relay host you provide actually accepts SMTP connections.

f. To enable file attachments on comments, P4 Code Review must have a depot location to save the attachment files in. P4 Code Review can create this depot and set the depot protections for you automatically or you can manually set it up later.

Important: You must be a user with *super* user permissions to create the .swarm depot and set protections.

- checking depot

P4 Code Review has the ability to store attachments against review comments. To do this, it needs to have a depot where they are stored. By default, this is `//swarm`. We can create the depot for you automatically, and set the protections on it to the following:

```
list user * * -//.swarm/...
admin user swarm * //.swarm/...
super user super * //.swarm/...
```

If you want to enable attachments, and for the depot and protections to be set up for you, then say yes here. If you say no, then you can still do this manually later.

Do you want to create a .swarm depot and set protections? (y/n) [n]

When prompted to automatically create the depot, select one of the following:

- Type **y** to automatically create the **.swarm** depot and set protections.
- Type **n** skip depot creation. You can create the depot and set the protections manually later if you want to, see ["Comment attachments" on page 575](#).

If you choose to automatically create the depot, the script will create the depot, set the protections, and report when completed.

Depot **//.swarm** successfully created. Protections have been set so that only the 'swarm' and 'super' have permissions to access it directly.

- g. Configure the following to enable communication between P4 Code Review and P4 Server.
 - Configure the P4 Server to promote all shelved changes. For more information, see ["Configure the P4 Server to promote all shelved changes" on page 198](#).
 - Configure the **p4 key** access on P4 Server. For more information on **p4 key** and their access levels, see ["Hiding P4 Code Review storage from regular users" on page 226](#).
 - Allow P4 Code Review to work with 'exclusive open' files. For more information about exclusive locks, see ["Handling Exclusive Locks" on page 227](#).

Run the following to set these three configurations:

```
p4 configure dm.shelve.promote=1
p4 configure set dm.keys.hide=2
p4 configure set filetype.bypasslock=1
```

Tip: If this setting is not enabled in P4 Server, P4 Code Review will report exceptions when working with exclusively opened files similar to Cannot unshelve review (x). One or more files are exclusively open, and that you must have the filetype.bypasslock configurable enabled.

- h. P4 Code Review needs to know about some P4 Server events to operate correctly. Use P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about these events. P4 Server Extensions are easier to install and maintain than triggers.

Important: You must be a user with *super* user permissions to install and configure P4 Server Extensions.

Do you want to use Swarm's Helix Core server extension?

Configuring Server extensions requires super user access to the Helix Server.

If you install the Swarm server extension, do not install the Swarm triggers.

Server extensions are supported for:

* Linux: Helix server 19.2 and later.

* Windows: Helix server 21.2 and later.

Use server extensions?

When prompted to Use server Extensions, choose one of the following:

- **Recommended:** Type **y** to use P4 Server Extensions. The configuration script will try and install and configure the P4 Server extension.
- Type **n** to use P4 Server Triggers. Triggers must be installed manually, P4 Code Review will prompt you to do this when the P4 Code Review configuration script completes.

If you choose to install the P4 Server Extensions, the script will:

- check your P4 Server supports P4 Server Extensions
- check if P4 Server Extensions are installed and configured on your P4 Server. If they are, P4 Code Review does not need to do anything
- install and configure the P4 Server extension on your P4 Server

If any of these checks fail, P4 Code Review will not install the P4 Server extension and will report the issues on the configuration summary screen.

Once all of the information has been provided, the configuration script configures P4 Code Review. When it has completed the configuration, the configuration summary screen is displayed, for example:

```

.....
-restarting Apache...
-Apache restarted
configure-swarm.sh: Thu Aug 26 11:31:36 PDT 2021: completed configuration of
Helix Swarm

.....
::
:: Swarm is now configured and available at:
::
::   http://myhost/
::
:: You may login as the Swarm user [swarm] using the password
:: you specified.
::
:: Ensure that you have configured the Swarm hostname in your
:: network's DNS, or have added an IP address-to-hostname
:: mapping to your computer's hosts configuration so that you
:: can access Swarm.
::
:: Server side extensions are installed and configured
:: on your P4D server.
::
:: Documentation for optional post-install configuration, such as
:: configuring Swarm to use HTTPS, operate in a sub-folder, or on a
:: custom port, is available:
::
::
https://www.perforce.com/perforce/doc.current/manuals/swarm/setup.post.html
::
.....

```

Note:

If you have installed P4 Code Review on a host that does not provide other web services, you may wish to disable Apache's default site configuration. Doing so means that regardless of the hostname a user might use to reach the web server hosting P4 Code Review, P4 Code Review would be presented.

Be aware that disabling Apache's default site configuration could disable existing web services or content.

To disable Apache's default configuration, manually adjust the Apache configuration. Such changes require familiarity with Apache configuration; for more details, see [Apache HTTP server project](#).

To add Swarm as an HTML tab in the P4 Visual Client (P4V), see [HTML Tabs](#) section in the *P4VJS Developer Guide*.

3. Configure SELinux for P4 Code Review, see "[SELinux configuration](#)" below.
4. The basic P4 Code Review configuration is now complete.

Important:

If your P4 Server is configured for P4 AS, you can force all of your users to authenticate via your Identity Provider (IdP) by disabling fall-back to passwords. To disable fall-back to passwords on the P4 Server, run the following command:

```
p4 configure set auth.sso.allow.passwd=0
```

5. Do one of the following:
 - **If the P4 Server extension was installed by the configuration script:** Review the post-install configuration options to customize your P4 Code Review installation, see "[Post-install configuration options](#)" on page 214.
 - **If the P4 Server extension was not installed, you must install triggers:** Configure P4 Server for P4 Code Review, see "[Installing triggers](#)" on page 187.

SELinux configuration

P4 Code Review supports SELinux which is an advanced access control mechanism that improves security for Linux distributions.

SELinux operates in one of three modes:

- **enforcing:** this mode blocks and logs any actions that do not match the defined security policy. This is the default mode for SELinux.
- **permissive:** this mode logs actions that do not match the defined security policy but these actions are not blocked.
- **disabled:** in this mode SELinux is off, actions are not blocked and are not logged.

Tip:

To check the mode SELinux is operating in, view the `/etc/selinux/config` file with `vi` or a similar editor:

```
root $ vi /etc/selinux/config
```

SELinux must be configured to enable it to work correctly with P4 Code Review, these configuration steps are shown below.

Note:

You must complete the P4 Code Review package "[Installation](#)" on page 141 steps, and the "[Post-installation configuration](#)" on page 146 steps before configuring SELinux.

Configure SELinux to enforcing mode

Run the following commands as root:

1. Install the `policycoreutils-python` package that contains the **semanage** configuration tool, this is used to configure SELinux

```
root $ yum install policycoreutils-python
```
2. Check the current SELinux mode:

```
root $ getenforce
```
3. SELinux will report its mode as; `enforcing`, `permissive`, or `disabled`.
 - a. If the mode is not set correctly edit the `/etc/selinux/config` file with `vi` or a similar editor.

```
root $ vi /etc/selinux/config
```
 - b. Edit the config file so that `SELinux=` is set to `enforcing`.
 - c. Save the config file.
 - d. Reboot the server to complete the SELinux mode change.
4. Allow content in `/opt/perforce/swarm` to be read and written by the `httpd` process:

```
root $ semanage fcontext -a -t httpd_sys_rw_content_t "/opt/perforce/swarm(/.*)"?"
root $ restorecon -R /opt/perforce/swarm
```
5. Allow the `httpd` process to connect to other networked services, for example P4D and Redis:

```
root $ setsebool -P httpd_can_network_connect 1
```
6. Allow comment attachment thumbnails to be created:

```
root $ setsebool -P httpd_tmp_exec 1
```
7. Allow the files in `p4-bin` to be executed by the `httpd` process:

```
root $ semanage fcontext -a -t httpd_sys_script_exec_t '/opt/perforce/swarm/p4-bin(/.*)"?'
root $ restorecon -R -v /opt/perforce/swarm/p4-bin
```
8. Remove the executable constraints on Redis, allowing it to be started by `systemd` at boot time:

```
root $ semanage fcontext -a -t bin_t /opt/perforce/swarm/sbin/redis-server-swarm
root $ restorecon -v /opt/perforce/swarm/sbin/redis-server-swarm
```
9. Restart the system:

```
root $ systemctl restart httpd
```
10. Check that you can log in to P4 Code Review.
11. **Only if required:** Relabel your filesystem, see note before relabeling:

Important:

Relabeling your file system can be a time consuming process, it is recommended that you only do this if you need to. This depends entirely on your SELinux setup, Perforce cannot give you advice on this.

```
root $ touch /.autorelabel
```

12. Reboot the server.
13. Check that you can log in to P4 Code Review.
14. SELinux is now configured for P4 Code Review.

Note:

If you can not log in to P4 Code Review it is possible that SELinux is blocking P4 Code Review because its configuration is incorrect. You will need to troubleshoot the SELinux configuration to find any issues.

Install the `setroubleshoot` package, this contains `sealert` which is used when troubleshooting SELinux:

```
root $ yum install setroubleshoot
```

`sealert` helps you to interpret the contents of the `audit.log`. Run the following command:

```
root $ sealert -a /var/log/audit/audit.log
```

Error message: If you see an error message with a title similar to the message below, it may be because you are running on a Virtual Machine (VM).

```
root $ SELinux is preventing /usr/sbin/ldconfig from write access on the directory etc.
```

Install `open-vm-tools` on the VM and reboot the VM.

```
root $ yum install open-vm-tools
```

Configure SELinux permissive or disabled mode

Run the following as root:

1. Check the current SELinux mode:


```
root $ getenforce
```
2. SELinux will report its mode as; `enforcing`, `permissive`, or `disabled`.
 - a. If the mode is not set correctly edit the `/etc/selinux/config` file with `vi` or a similar editor.


```
root $ vi /etc/selinux/config
```
 - b. Edit the config file so that `SELinux=` is set to `permissive` or `disabled` as required.
 - c. Save the config file.
 - d. Reboot the server to complete the SELinux mode change.
3. Check that you can log in to P4 Code Review.
4. SELinux is now configured for P4 Code Review.

Uninstall

1. Depending on how you have configured your Helix Core Server events, do one of the following:

- Uninstall the P4 Server extension. To do this, run the following command on your P4 Server:


```
p4 extension --delete Perforce::helix-swarm --yes
```
 - Uninstall the P4 Code Review triggers. As a *super* user, run the `p4 triggers` command from your P4 Server and manually remove all the P4 Code Review trigger code lines.
- 2. Uninstall the P4 Code Review main package, the P4 Code Review triggers package, and the P4 Code Review optional package. The following example assumes that the packages are all installed on the same server (run the command as root):


```
yum remove helix-swarm helix-swarm-triggers helix-swarm-optional
```
- 3. Remove the P4 Code Review trigger scripts from the server hosting your P4 Server.

Important:

If you manually installed the trigger script, perhaps because the server hosting your P4 Server cannot use packages (for example Windows), manually remove the script.

Run P4 Code Review using a Docker container

This section assumes that you have basic knowledge of how to use a Docker container. There are numerous ways of configuring a Docker environment, and the examples outlined in this section are only to get you started. For more information on all of the options for Docker configurations, see [Docker](#).

This section details how to use a Docker container to run P4 Code Review either in a test environment or a production environment.

- ["Prerequisites" on the facing page](#)
- ["P4 Code Review images" on the facing page](#)
- ["Running the Docker container" on the facing page](#)
- ["Persisting Docker containers for a production environment" on page 159](#)
- ["Restarting the Docker container" on page 160](#)
- ["Migrating P4 Code Review to a Docker container" on page 161](#)
- ["Advanced configuration options " on page 162](#)
- ["Example of running Swarm using docker-compose" on page 165](#)

Here is an overview of setting up a Docker container to run P4 Code Review in a production environment:

1. Start P4 Code Review using a Docker container with the `.env` configuration file and a mounted P4 Code Review or data directory.

This stores all the `.env` configuration options into the data directory in the correct format.
2. Stop the Docker container.
3. Start a Docker container using the mounted P4 Code Review or data directory without the `.env` configuration options. The stored configurations in the `config.php` file are used.

Prerequisites

To run P4 Code Review using a Docker container, you must have the following:

- An environment capable of running Docker images. For more information about Docker images, see [Docker](#).
- Access to the internet to download the Docker image. You can download an image and transfer it to a machine with no internet access. To do this, use the `docker save` and `docker load` commands. For more information on how to use these commands, see [Docker](#). By default, the PHP memory limit of the Docker container is set to 2 gigabytes (GB).
- Access to a running P4 Server instance. For more information about the P4 Server, see [P4 Server Administration Documentation](#).
- You must be a *super* user if you are configuring P4 Code Review for the first time. If you are re-using an existing `config.php` file, this is not necessary.
- The P4 Code Review Docker container can run on any host operating system that supports Docker, with official support for RHEL and Ubuntu hosts. The P4 Code Review container is built and delivered with Ubuntu 22.04 (Jammy) as its base image.

P4 Code Review images

To access the P4 Code Review images on Docker Hub, see [Helix Swarm Docker Container](#).

There is a tag for each major version of P4 Code Review, for example, 2022.2. These images are updated for every patch release.

The **latest** tag points to the very latest released version of P4 Code Review. The major version number of P4 Code Review used by the latest tag will change when new major versions are released, this includes the patches for the latest version.

Running the Docker container

This section details how you can configure a fresh installation of P4 Code Review against a P4 Server to get a test instance running. It does not specify how to persist data on an external volume, so should not be used for a production environment. For more information about persisting data and configuration in a production environment, see "[Persisting Docker containers for a production environment](#)" on page 159.

Important: When running Swarm in a Docker container for production systems, your configuration must persist data when it is shutdown.

Create a configuration file

In this section, we create an example `.env` configuration file which is then used to configure a Docker container.

If the `/opt/perforce/swarm/data` directory is mapped outside the Docker container, after the initial configuration run, you can remove the configuration options from the `.env` configuration file as they are stored in `config.php` file.

Create a `.env` configuration file in the current directory as follows:

```
P4D_PORT=ssl:myperforce:1666
P4D_SUPER=super
P4D_SUPER_PASSWD=HelixDockerBay94
SWARM_USER=swarm
SWARM_PASSWD=HelixDockerBay94
```

```
SWARM_HOST=mymachine
```

```
SWARM_REDIS=helix-redis
SWARM_REDIS_PORT=7379
```

```
# If set to 'y', then extensions will be installed even if they already
# exist, overwriting existing configuration.
SWARM_FORCE_EXT=y
```

- **P4D_PORT**

The **P4D_PORT** value must be set to the P4 Server instance.

Important: Ensure that the Docker image is able to communicate with the P4 Server instance.

- **P4D_SUPER and P4D_SUPER_PASSWD**

The **P4D_SUPER** user must be an existing P4 Server user with super privileges. The **P4D_SUPER_PASSWD** must be a password and cannot be a ticket.

- **SWARM_USER and SWARM_PASSWD**

The **SWARM_USER** is created if it does not already exist. If it already exists, the **SWARM_USER** must have at least admin privileges. The **SWARM_PASSWD** must be set to the existing password if the **SWARM_USER** already exists or you can use it to set a new password. The **SWARM_PASSWD** must obey the normal rules for a P4 Server password.

Important: The **P4D_SUPER_PASSWD** and **SWARM_PASSWD** configurables in the **.env** configuration file must be a password, not ticket. The **SWARM_PASSWD** is converted into a host locked ticket before it is stored.

- **SWARM_HOST**

The hostname set by **SWARM_HOST** should be reachable by the P4 Server. In this example setup, we assume that the hostname is the name of the host running the Docker image, rather than the Docker image itself. When the Docker image is started, port 80 and, optionally, port 443 are mapped to the host's port. This way anything that can reach the host can reach P4 Code Review.

To run Swarm on a different port than port 80, you must add the port number to the **SWARM_HOST** parameter.

- **SWARM_REDIS and SWARM_REDIS_PORT**

This is the hostname and port for the Redis server. This example setup assumes that Redis server is running in a Docker container as setup in ["Start a single Docker container" below](#).

- `SWARM_FORCE_EXT`

When set to `y`, `SWARM_FORCE_EXT` will force the extensions to be installed or reinstalled. The existing P4 Code ReviewP4 Serverextension configuration is overwritten.

Start a single Docker container

Running the following command will start a single instance of P4 Code Review running that uses the `.env` configuration file. It assumes that there is a Redis server and P4 Server to connect to. The Redis configurables in the `.env` file in ["Create a configuration file" on page 157](#) section will not work in this example. You must ensure that P4 Code Review points to an existing Redis instance.

```
docker run -d --name helix-swarm --network-alias helix-swarm \
--env-file .env -p 80:80 perforce/helix-swarm
```

You can shut down the P4 Code Review instance as follows:

```
docker stop helix-swarm
```

You can delete the P4 Code Review instance as follows:

```
docker rm helix-swarm
```

Setup a local private network

A simplified option is to run Redis server as a Docker container. You can setup a local private network to run both, the Redis server and P4 Code Review. This makes the Redis server private to P4 Code Review.

The Redis configurables in the `.env` file created in the ["Create a configuration file" on page 157](#) section is used in the example code below. Use the following docker commands to setup a local private network:

```
docker network create helix
docker run -d --name helix-redis --network helix --network-alias helix-redis \
redis redis-server --protected-mode no --port 7379
```

```
docker run -d --name helix-swarm --network helix --network-alias helix-swarm \
--env-file .env -p 80:80 perforce/helix-swarm
```

Persisting Docker containers for a production environment

For a production environment, the P4 Code Review and Redis server cache data should persist even after the Docker container has stopped and deleted. You can do this by using `bind mount` or `volume mount` when starting the Docker containers. This mounts the contents of `/opt/perforce/swarm/data` directory outside the Docker container.

To preserve the P4 Code Review configuration and Redis server cache data outside the Docker container, use the following:

```
docker network create helix
```

```
mkdir -p storage/redis-data
docker run -d --name helix-redis --network helix --network-alias helix-redis \
-v $PWD/storage/redis-data:/data redis \
redis-server --protected-mode no --port 7379 --appendonly yes
```

```
mkdir -p storage/swarm-data
docker run -d --name helix-swarm --network helix --network-alias helix-swarm \
-p 80:80 -v $PWD/storage/swarm-data:/opt/perforce/swarm/data \
--env-file .env perforce/helix-swarm
```

The `--appendonly yes` option for the Redis server will cause it to write its cache data to disc. The `/data` directory can then be mounted externally to persist it beyond the lifetime of the Docker container.

P4 Code Review log files, the worker queue, tokens, and workspaces are also preserved because they are part of the `/opt/perforce/swarm/data` directory.

Restarting the Docker container

When you restart a Docker container, the `/opt/perforce/swarm/data/config.php` file is used to configure the system.

If there is no `config.php` file in the `/opt/perforce/swarm/data/` directory, the `.env` configuration file is used.

When you make any changes to the `config.php` file, you must reload the configuration cache for Swarm to use the updated configurations. If you restart a Docker container, P4 Code Review will not use the updated configuration until the configuration cache has been reloaded. For more information on how to restart configuration cache, see ["Reload Configuration button" on page 720](#) section.

Once you have restarted the Docker container, you must be a `root` user to access files in the `/opt/perforce/swarm/data/` directory.

You can restart the Docker container as follows:

```
docker run -d --name helix-swarm --network helix --network-alias helix-swarm \
-p 80:80 -v $PWD/storage/swarm-data:/opt/perforce/swarm/data \
perforce/helix-swarm
```

Docker directory

When you restart the Docker container, a Docker directory is created `/opt/perforce/swarm/data/docker` for your P4 Code Review installation. The Docker directory includes the Apache site configuration files, P4 Code Review configuration files, P4 Code Review version file, and the Docker container version. All the files in the Docker directory are linked to the original files in their original locations. This allows you to edit files outside the Docker container. Any change to any file in the Docker directory is preserved between restarts.

For example,

- `/etc/apache2/sites-available -> docker/sites-available`

The entire `/etc/apache2/sites-available` directory is soft linked to the `/opt/perforce/swarm/data/docker/sites-available` directory. If a `docker/sites-available` file exists instead of the directory, then no soft link is created. This allows you to mount the Apache directory outside the Docker container.

All the configuration files have `a2ensite` run against them. This allows you to automatically run multiple Apache site configurations.

- `/opt/perforce/swarm/public/custom -> docker/custom`

The entire `/opt/perforce/swarm/public/custom` directory is soft linked to the `docker/custom` directory.

- `/opt/perforce/etc/swarm-cron-hosts.conf -> docker/swarm-cron-hosts.conf`

The entire `/opt/perforce/etc/swarm-cron-hosts.conf` file is soft linked to the `docker/swarm-cron-hosts.conf` file.

Restriction: To enable HTTPS, you must setup a new Apache Virtual Host.

Migrating P4 Code Review to a Docker container

When migrating a P4 Code Review installation to a Docker installation on the same system, you must do the following:

- Use a host unlocked ticket.
- Stop the host cron jobs and use the cron jobs available in the Docker container.
- If your P4 Code Review installation uses the default Redis server, then you must configure a new Redis instance for the dockerised P4 Code Review. To configure a new Redis instance, see ["Create a configuration file" on page 157](#).

You can migrate an existing P4 Code Review server setup into a Docker container. To do this, you must copy the `/opt/perforce/swarm/data` directory to a location that is accessible for the Docker container as follows:

1. Create a `./storage` directory.
2. Copy contents of the `/opt/perforce/swarm/data/` directory to `./storage/swarm-data` directory.
3. Ensure that the Redis connection configuration is defined correctly in the `config.php` file. Below is an example of the Redis connection configuration.

```
<?php
// this block should be a peer of 'p4'
'redis' => array(
    'options' => array(
        'password' => null, // Defaults to null
        'namespace' => 'Swarm',
        'server' => array(
```

```
        'host' => 'localhost', // Defaults to 'localhost' or enter your Redis server
hostname
        'port' => '7379', // Defaults to '7379' or enter your Redis server port
    ),
),
),
```

You can configure additional Redis parameters such as `items_batch_size`, `check_integrity` and `population_lock_timeout`. See ["Redis server" on page 662](#).

4. Ensure that there is an Apache site configuration file at `swarm-data/etc/perforce-swarm-site.conf`. This Apache site configuration file is copied into the Docker when it starts.

```
docker network create helix
```

```
mkdir -p storage/redis-data
docker run -d --name helix-redis --network helix --network-alias helix-redis \
-v $PWD/storage/redis-data:/data redis \
redis-server --protected-mode no --port 7379 --appendonly yes
```

```
mkdir -p storage/swarm-data
docker run -d --name helix-swarm --network helix --network-alias helix-swarm \
-p 80:80 -v $PWD/storage/swarm-data:/opt/perforce/swarm/data \
perforce/helix-swarm
```

A `docker` directory is created in the `/opt/perforce/swarm/data/` location. During the initial configuration, the `docker` directory includes the following files:

- Apache site configuration file
`perforce-swarm-site.conf`
- List of cron worker configuration
`swarm-cron-hosts.conf`
- Directory which contains any custom configurations for Swarm
`custom`

Advanced configuration options

When a `config.php` file exists, all configurations in the `.env` configuration file are ignored.

When no Apache site configuration file is available, the `SWARM_HOST` parameter in the `.env` configuration file is used to configure the Apache server.

We recommend that the `/opt/perforce/swarm/data/docker` directory is mounted outside the Docker container. So changes to any files in the Docker directory are preserved between restarts.

To modify the configuration of the P4 Code Review docker image, you can edit the following parameters in the `.env` configuration file:

Configuration	Default	Description
P4D_SUPER	super	Name of the super user.
P4D_SUPER_PASSWD	<i>no default</i>	Password of the super user.
P4D_PORT	ssl:perforce:1666	Port for the P4 Server (P4D).
SWARM_USER	swarm	Name of the P4 Code Review user.
SWARM_PASSWD	<i>no default</i>	Password of the P4 Code Review user.
SWARM_MAILHOST	localhost	Mail server address
SWARM_HOST	helix-swarm	Hostname of the P4 Code Review server.
SWARM_FORCE_EXT	n	To overwrite P4 Code Review Extensions, set SWARM_FORCE_EXT = 'y'.
SWARM_REDIS	helix-redis	Hostname of the Redis server.
SWARM_REDIS_PORT	7379	Port number of the Redis server.

Configuration	Default	Description
<code>SWARM_REDIS_PASSWD</code>		Password for the Redis server.
<code>SWARM_REDIS_NAMESPACE</code>		Namespace for the Redis server.

After the P4 Code Review setup is complete, you can edit the `config.php` file to include support for Email configuration, JIRA integration, and for Multiple P4 (P4D) instances.

Apache server configuration

By default, the `mod_rewrite` and `mod_ssl` modules are enabled for Apache server. The SSL module is required to run HTTPS from within the docker container.

When using SSL, ensure that you have defined a location for the SSL key and the SSL certificate file. Do this in one of the following ways:

- Save the SSL key and certificate file in the `/opt/perforce/swarm/data/docker` directory and add a reference to them in the Apache site configuration file `perforce-swarm-site.conf`.
- Save the SSL key and certificate file in a different location and mount that location externally. This ensures that the private key file is outside P4 Code Review.
- Save the SSL key and certificate file externally to the docker container. Use a web server as a proxy for the docker container. Enable the proxy web server to provide any HTTPS functionality.

Apache proxy

In cases where the Docker container is behind another web server, you may run into issues with P4 tickets unless they are host unlocked. Generally, the Apache server forwards the IP address of the client to P4 Code Review. This allows a ticket that is obtained on a client machine to work with P4 Code Review, even though P4 Code Review is on a different machine.

If P4 Code Review is behind a web proxy (such as a second Apache server, running SSL), the IP address of the client is lost. In this scenario, host unlocked tickets must be used that can be obtained using `p4 login -ap`.

Alternatively, you can use the `remoteip_module` to allow the container Apache to trust the proxy that enables forwarding the client IP addresses. To do this, add the following code into the `perforce-swarm-site.conf` file, at the same level as the `DocumentRoot`. The `RemoteIPInternalProxy` should be the IP address of the host, as seen by the container.

```
RemoteIPHeader X-Forwarded-For
RemoteIPInternalProxy 172.19.0.1/24
```


Example of running Swarm using docker-compose

This section details an example usage of the docker-compose command. In the following example:

- The docker-compose.yml file is setup in a directory from where the commands are run.
- Creates a network for the two containers to run in.
- Sets up a Redis container to run Redis in, with external storage in the storage directory.
- Sets up a P4 Code Review container with external storage for the data directory and the /var/www directory.

```
#
=====
=== #
# This compose file runs a migrated Swarm instance. Redis and Swarm are in #
# containers, p4d is on the host machine.                                #
#
=====
=== #
version: '2.2'

networks:
  default:
    driver: bridge
    ipam:
      driver: default
      config:
        - subnet: 172.19.0.0/16
          gateway: 172.19.0.1
    name: swarm-net

services:
  helix.swarm:
    image: perforce/helix-swarm:latest
    hostname: helix-swarm
    domainname: helix
    volumes:
      - ./storage/swarm-data:/opt/perforce/swarm/data
      - ./storage/www:/var/www
    ports:
      - 127.0.0.1:80:80
    working_dir: /opt/perforce/swarm
    depends_on:
      - helix.redis
    tty: false

  helix.redis:
    image: "redis"
```

hostname: helix.redis

domainname: helix

command: redis-server --protected-mode no --port 7379 --appendonly yes

volumes:

- ./storage/redis-data:/data

Install and configure P4 Code Review manually from a Tarball

Important:

- Review the runtime dependencies before you install P4 Code Review, see ["Runtime dependencies" on page 91](#).
- P4 Code Review no longer supports PHP 8.0 version.
- **RHEL 8:** Use the Remi repository configuration package (remi-release-8.rpm) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the epel-release-latest-8.noarch.rpm repository configuration package to give P4 Code Review access to EPEL packages.
- **RHEL 9:** Use the Remi repository configuration packages (remi-release-8.rpm and remi-release-9.rpm) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the epel-release-latest-8.noarch.rpm and epel-release-latest-9.noarch.rpm repository configuration package to give P4 Code Review access to EPEL packages.

1. Download the [P4 Code Review tarball](#).
2. Expand the P4 Code Review package (a *compressed tarball*).

From the command line, expand it via the tar command:

```
tar -xzf swarm.tgz
```

The contents of the P4 Code Review package are expanded into a top-level folder named **swarm-*version***, where *version* corresponds to the version downloaded.

Tip:

Many graphical file manager applications (Nautilus on Linux, etc.) can automatically expand the tarball package by simply double-clicking it.

3. Move the contents of the P4 Code Review package to the correct location.

Identify a location for the P4 Code Review files; this should correspond to a location associated to the virtual host configured under Apache (see ["Apache configuration" on page 171](#)). For example:

```
mv /path/to/swarm-version /opt/perforce
```

```
mv /path/to/swarm-version /path/to/vhosts/swarm
```

4. Assign correct ownership and permission for the P4 Code Review files.

The data top-level folder in the P4 Code Review distribution needs to be writeable by the web server. To achieve this effect, simply change ownership of the data folder to the web user:

```
sudo chown -R www /path/to/vhosts/swarm/data
```

The `www` user above is an example of what the web server user name might be. Depending on your distribution, this could be `_www`, `web`, `nobody` or something else entirely.

If your web server is already running, you can discover the user with:

```
ps aux | grep -E 'apache|httpd'
root   3592  0.0  0.5 405240 20708 ?    Ss   May03  4:32 /usr/sbin/apache2 -k start
www    20016  0.0  0.2 405264  9796 ?    S    07:45  0:00 /usr/sbin/apache2 -k start
```

In this example, `www` is the user Apache is running as.

From a security perspective, we recommend that the minimum file permissions should be granted to the user/group under which the web server runs against the P4 Code Review distribution.

5. Configure Redis, see ["Redis configuration" below](#).

Redis configuration

P4 Code Review requires Redis to manage its caches and by default P4 Code Review uses its own Redis server on the P4 Code Review machine. P4 Code Review caches data from the P4 Server to improve the performance of common searches in P4 Code Review and to reduce the load on the P4 Server. Redis is included in the P4 Code Review Tarball installation.

This section describes how to configure the P4 Code Review Redis service on the P4 Code Review machine. Do not change the default values of the following configuration files if you are using the P4 Code Review Redis server.

Tip:

If required, you can use your own Redis server instead of the P4 Code Review Redis server. For instructions on how to configure P4 Code Review to use your own Redis server, see ["Use your own Redis server" on page 224](#).

P4 Code Review has two Redis binaries in `SWARM_ROOT/p4-bin/bin.linux26x86_64/`:

- `redis-server-swarm`
- `redis-cli-swarm`

1. Configure the Redis cache database, enter the following details in the `redis-server.conf` file in `/opt/perforce/etc/`:

```
bind 127.0.0.1
port 7379
supervised auto
save ""
dir /opt/perforce/swarm/redis
```

Default values:

Tip:

- The default settings are shown below, the `redis-server.conf` file contains more detailed information about the Redis configuration for P4 Code Review.
- On P4 Code Review systems with a large number of users, groups, and projects, start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves, see the `redis-server.conf` file comments for detailed information.

- `bind 127.0.0.1` - Redis server IP address (loopback interface)
- `port 7379` - Redis server port number
- `supervised auto` - detects the use of `upstart` or `systemd` automatically to signal that the process is ready to use the supervisors
- `save ""` - background saves disabled, recommended.
- `dir /opt/perforce/swarm/redis` - the directory the Redis cache database is stored in.

Tip:

To fine-tune your Redis server settings, make your changes in the Laminas Redis adapter. For information about the Laminas Redis adapter, see the [Laminas Redis Adapter documentation](#).

2. The Redis cache database must be running before P4 Code Review starts so we recommend setting it up as a service so that it starts automatically at boot time.

The following example script will:

- run the Redis service as the Perforce user
- use the configuration file in `/opt/perforce/etc/redis-server.conf`
- assume the database server is installed in `/opt/perforce/swarm/p4-bin/bin.linux26x86_64/`

To configure Redis as a service, add a script with the following content in `/etc/systemd/system/redis-server-swarm.service`

```
[Unit]
Description=Redis Server for Swarm
After=network.target

[Service]
ExecStart=/opt/perforce/swarm/p4-bin/bin.linux26x86_64/redis-server-swarm
/opt/perforce/etc/redis-server.conf
ExecStop=/opt/perforce/swarm/p4-bin/bin.linux26x86_64/redis-cli-swarm shutdown
Restart=always
RestartSec=10
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=redis
User=perforce
Group=perforce

[Install]
WantedBy=multi-user.target
```

3. Create the `SWARM_ROOT/data/config.php` file if it does not already exist. The `redis` block of the `SWARM_ROOT/data/config.php` file contains the password, namespace, host and port details of the P4 Code Review Redis server:

```
<?php
// this block should be a peer of 'p4'
'redis' => array(
    'options' => array(
        'password' => null, // Defaults to null
        'namespace' => 'Swarm',
        'server' => array(
            'host' => 'localhost', // Defaults to 'localhost' or enter your Redis server
hostname
            'port' => '7379', // Defaults to '7379' or enter your Redis server port
        ),
    ),
    'items_batch_size' => 100000,
    'check_integrity' => '03:00', // Defaults to '03:00' Use one of the following two
formats:
        // 1) The time of day that the integrity check starts each day. Set in
24 hour format with leading zeros and a : separator
        // 2) The number of seconds between each integrity check. Set as a
positive integer. Specify '0' to disable the integrity check.
    'population_lock_timeout' => 300, // Timeout for initial cache population. Defaults to
300 seconds.
),
```

Configurables:

- `password`: Redis server password. Defaults to null and should be left at default if using the P4 Code Review Redis server.
- `namespace`: the prefix used for key values in the Redis cache. Defaults to `Swarm` and should be left at default if using the P4 Code Review Redis server.

Note:

If you have multiple P4 Code Review instances running against a single Redis server, each P4 Code Review server must use a different Redis namespace. This enables the cache data for the individual P4 Code Review instances to be identified. The namespace is limited to ≤ 128 characters.

If one or more of your P4 Code Review instances is connected to multiple P4 Servers, the Redis namespace includes the server label and the character limit is reduced to ≤ 127 characters, see ["Multiple P4 Server instances" on page 638](#).

- `host`: Redis server hostname. Defaults to `localhost` and should be left at default if using the P4 Code Review Redis server.
- `port`: Redis server port number. Defaults to `7379`. P4 Code Review uses port `7379` as its default to avoid clashing with other Redis servers that might be on your network. It should be left at default if using the P4 Code Review Redis server. The default port for a non-P4 Code Review Redis server is `6379`.
- `items_batch_size`: Maximum number of key/value pairs allowed in an `mset` call to Redis. Sets exceeding this will be batched according to this maximum for efficiency. Defaults to `100000`.

Note:

The default value of `100000` was chosen to strike a balance between efficiency and project data complexity. This value should not normally need to be changed, contact support before making a change to this value.

- `check_integrity`: In some circumstances, such as when changes are made in the P4 Server when P4 Code Review is down or if errors occur during updates, the Redis cache can get out of sync with the P4 Server. P4 Code Review can run a regular integrity check to make sure that the Redis caches and P4 Server are in sync. If an integrity check finds an out of sync cache file, P4 Code Review automatically updates the data in that cache.

The `check_integrity` configurable specifies when the Redis cache integrity check is run. Set as a specific time of day (24 hour format with leading zeros) or a number of seconds (positive integer) between checks. Disable the integrity check with `'0'`. Defaults to `'03:00'`.

- `population_lock_timeout`: specifies the timeout, in seconds, for initial cache population. If you have a large P4 Code Review system, increase this time if the initial cache population times out. Defaults to `300` seconds.

Configure Apache, see ["Apache configuration" on the facing page](#).

Apache configuration

The configuration of the Apache HTTP Server (Apache) can vary between OS distributions; see the documentation specific to your installation of Apache.

Important:

The following notes are applicable for RHEL 8 and RHEL 9:

- **P4 Code Review 2020.2 and later:** these versions of P4 Code Review uses the Remi repository for RHEL 8 and RHEL 9. This provides PHP 8.x installed in the standard file system structure. This means that the old httpd24-httpd version of Apache is no longer needed, and the standard system version of Apache is being used again.
The SCL Apache site configuration file was installed at this location for P4 Code Review 2019.1 to 2020.1:
`/opt/rh/httpd24/root/etc/httpd/conf.d/perforce-swarm-site.conf`
If this exists when P4 Code Review is upgraded to 2020.2 and later, this file is copied to `/etc/httpd/conf.d/perforce-swarm-site.conf` if there is no file at the destination. It is also re-written to change references from `/var/log/httpd24` to `/var/log/httpd`
If a site configuration file for P4 Code Review already exists in `/etc/httpd`, the copy and re-write is not performed.
After upgrade, httpd24-httpd is disabled.
- To avoid seeing the Apache HTTP server Linux test page when you start the Apache server, comment out the content of the `welcome.conf` file located in the `/etc/httpd/conf.d/` directory.
- To avoid loading the Apache HTTP server example configuration instead of the P4 Code Review configuration when the Apache server starts, rename the `autoindex.conf` file located in the `/etc/httpd/conf.d/` directory to `z-autoindex.conf` or similar. This is required because Apache runs the first conf file it finds in the `/etc/httpd/conf.d/` directory (alphabetical order) and that must be the `perforce-swarm-site.conf` file.

1. Locate your system's Apache configuration.

Common configuration directories include:

- `/etc/httpd/conf/`
- `/etc/apache2/`
- `/usr/local/apache2/conf/`
- `/Applications/XAMPP/etc/`

Within the configuration path, the main Apache configuration file is usually named one of the following:

- `httpd.conf`
- `apache2.conf`

Tip:

A longer discussion on the possible locations and names of Apache configuration files is available here: [DistrosDefaultLayout](#)

2. Set up an Apache virtual host (vhost) for your installation.

If your Apache configuration directory contains the directories `sites-available` and `sites-enabled`:

- a. Copy the appropriate virtual host definition below into the file `sites-available/swarm`.
- b. Enable the P4 Code Review virtual host definition.

```
sudo a2ensite swarm
```

Otherwise, copy the virtual host definition below into the bottom of the main Apache configuration file, `httpd.conf` or `apache2.conf`.

Virtual host definition example for Apache 2.4:

```
<VirtualHost *:80>
    ServerName myswarm.host
    AllowEncodedSlashes NoDecode
    ServerAlias myswarm
    ErrorLog "/path/to/apache/logs/myswarm.error_log"
    CustomLog "/path/to/apache/logs/myswarm.access_log" common
    DocumentRoot "/path/to/swarm/public"
    <Directory "/path/to/swarm/public">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

See Apache's virtual host documentation for details: [Apache Virtual Host documentation](#)

Note:

If you have installed P4 Code Review on a host that does not provide other web services, you may wish to disable Apache's default site configuration. Doing so means that regardless of the hostname a user might use to reach the web server hosting P4 Code Review, P4 Code Review would be presented.

Be aware that disabling Apache's default site configuration could disable existing web services or content.

Disabling Apache's default site configuration on Ubuntu hosts is easy. Run:

```
$ sudo a2dissite 000-default
```

3. Customize the virtual host definition.
 - a. Replace `myswarm.host` with the hostname for P4 Code Review on your network. This may require adjusting the DNS configuration on your network.

- b. Replace `myswarm` with the name of the subdomain hosting P4 Code Review. Many administrators choose `swarm`.

Note:

The string `myswarm` in the log file paths: this should match the subdomain name and prefix for the log files, to help coordinate the active host with the log files for that host. Doing this is particularly useful when your Apache server hosts multiple instances of P4 Code Review.

- c. Replace `/path/to/apache/logs` with the path where your Apache stores its log files. Apache's log files are typically named `access_log` and `error_log`.
 - d. Replace `/path/to/swarm` with the path to the P4 Code Review directory.
4. Verify that the correct Apache modules are enabled.
 - To query whether the PHP and Rewrite modules are active, use the `apachectl` utility to list all of the active modules (this may be named `apache2ctl` on your system):


```
apachectl -t -D DUMP_MODULES
```
 - Simply look for `php8_module` and `rewrite_module` in the output. If you see them, skip ahead to step 5.
 - If the Apache utility `a2enmod` is installed, use it to enable the PHP and Rewrite modules:


```
sudo a2enmod php8 rewrite
```
 - Without the `a2enmod` utility, edit the Apache configuration file by hand. Locate your Apache configuration file for modules and either uncomment or add the following lines:


```
LoadModule php8_module libexec/apache2/libphp8.so
LoadModule rewrite_module libexec/apache2/mod_rewrite.so
```
 - Note that your Apache installation may have different paths for the location of its modules (the `.so` files).
 5. Restart your web server.
 - To ensure that the Apache configuration changes you made become active, restart the web server.


```
sudo apachectl restart
```
 - Query Apache's active virtual hosts and modules to confirm your changes are in effect:


```
apachectl -t -D DUMP_VHOSTS
apachectl -t -D DUMP_MODULES
```

Important:

Apache must be configured to use the prefork MPM because P4PHP does not support threaded operation.

The prefork *MPM* is the default for Linux Apache installations, so you may not have to do anything.

For more information on Apache MPMs and configuration, see: [Multi-Processing Modules \(MPMs\)](#)

6. Configure PHP, see "[PHP configuration](#)" below.

PHP configuration

PHP can vary between OS distributions; see the documentation specific to your installation of PHP.

1. First determine which PHP `.ini` file is in use by the PHP Apache module.

Tip: The `php.ini` used by your web server may differ from the one used by the command-line interface. Running `php --ini` only shows the CLI configuration.

If you are having trouble determining which `.ini` file the PHP Apache module is using, create a PHP file that can be served through Apache with the following contents:

```
<?php phpinfo();?>
```

Point your browser to this file and look for this table row in the resulting table:

Loaded Configuration File

2. Open the `php.ini` configuration file and increase the `memory_limit` to 512M, default value is 256M.

Large changelists may fail to submit if the `memory_limit` is too low.

An example `memory_limit` setting in `php.ini`:

```
memory_limit = 512M
```

If you are still having memory limitation issues, see "[Troubleshooting out of memory \(OOM\) issues](#)" on page 178

3. Ensure that `date.timezone` is set correctly for your system.

Some distributions do not make a default timezone available to PHP, so the best practice to set the timezone for PHP explicitly. See the [list of supported timezones](#).

An example `date.timezone` setting in `php.ini`:

```
date.timezone = America/Vancouver
```

4. Verify that the `iconv`, `json`, and `session` extensions are present.

They are usually enabled by default, although you may have to install packages for them through your OS distribution. Verify they are present by searching for their respective names in the `phpinfo` output above.

5. Enable P4 API for PHP, the Perforce extension for PHP:

For P4 Code Review to communicate with the P4 Server, it needs the P4 API for PHP extension. P4 API for PHP variants are available for each [PHP 8 version supported](#) by P4 Code Review and each variant is available as either openssl 1.02 or openssl 1.1.1. All of the P4 API for PHP variants are supplied with the P4 Code Review package and tarball installations.

Note:

- For Linux, the default variants are compiled with glibc 2.11.
- P4 API for PHP 2019.1 or later is required for P4 Code Review 2019.1 and later.

P4 Code Review package and tarball installations: two versions of P4 API for PHP are supplied for PHP 8 version supported by P4 Code Review. They are located in the `p4-bin/bin.linux26x86_64` directory.

- `perforce-php8x.so` compatible with systems using SSL 1.0.2
- `perforce-php8x-ssl1.1.1.so` compatible with systems using SSL 1.1.1
- `perforce-php8x-ssl3.so` compatible with systems using SSL 3.0.0 (by default, Ubuntu 22.04 and 24.04 uses SSL 3.0.0)

Where x is the version of PHP 8.

P4 Code Review no longer supports PHP 8.0 version.

If the `perforce.ini` file is not pointing at the correct version of P4 API for PHP and you connect to an SSL enabled P4 Server:

- The P4 Code Review web-page will not load and you might see a Connection Reset error.
- There might be an undefined symbol: `SSLey` message in the Apache error log

To enable P4 API for PHP, edit the web server's `php.ini` file and add the following line:

```
extension=/path/to/swarm/p4-bin/bin.<platform>/perforce-<php8.x><ssl>.so
```

Replace `<php8.x>` with the version of PHP 8 you are running.

P4 Code Review no longer supports PHP 8.0 version.

Supported versions of PHP 8 are as follows:

- For PHP 8.1, use `php81`
- For PHP 8.2, use `php82`
- For PHP 8.3, use `php83`
- For PHP 8.4, use `php84`

Replace `<ssl>` with the variant of openssl your system is running. Enter the following into `<ssl>` for the variant of openssl you are running:

- For SSL 1.0.2, you do not need to enter a variant. See Example 1 below.
- For SSL 1.1.1, use `ssl1.1.1`
- For SSL 3.0.0, use `ssl3`

See the examples below:

Example 1: for a 64-bit Linux system running PHP 8.1 and SSL 1.0.2:

```
extension=/path/to/swarm/p4-bin/bin.linux26x86_64/perforce-php81.so
```

Example 2: for a 64-bit Linux system running PHP 8.3 and SSL 1.1.1:

```
extension=/path/to/swarm/p4-bin/bin.linux26x86_64/perforce-php83-ssl1.1.1.so
```

Example 3: for a 64-bit Linux system running PHP 8.4 and SSL 3.0.0:

```
extension=/path/to/swarm/p4-bin/bin.linux26x86_64/perforce-php84-ssl3.so
```

Alternatively, copy the extension file to the default location for PHP extensions, and then just add this line instead:

```
extension=perforce-<php8.x><ssl>.so
```

- Restart Apache for the changes to become active.
- To verify that P4 API for PHP is active, navigate to the `phpinfo` file you [created above](#). You should then see a `perforce` section (search for "Perforce Module"). It should report that the module is enabled and display the version information.

Note:

Be aware that any operating system upgrades on the machine hosting P4 Code Review may involve updates to PHP. If this occurs, the PHP `.ini` file needs to be updated to point to the correct *variant* of P4 API for PHP to match the version of PHP that the upgraded operating system is using.

- In addition, P4 Code Review greatly benefits from the following optional extension: ImageMagick extension for PHP to improve P4 Code Review's ability to preview graphic formats that web browsers typically cannot display, see "[ImageMagick \(imagemagick\) extension for PHP](#)" below.
- Configure P4 Code Review, see "[P4 Code Review configuration](#)" on page 179.

ImageMagick (imagemagick) extension for PHP

Imagick is a PHP extension that integrates the ImageMagick graphics library's API for the creation and manipulation of images. Enabling Imagick improves P4 Code Review's ability to preview graphics formats that web browsers typically cannot display and to create comment attachment thumbnails.

For more information, see:

- [Image Processing \(ImageMagick\)](#)
- [imagemagick](#)

Follow the instructions for your OS distribution:

RHEL and Rocky Linux

1. We recommend that you install Imagick from your OS distribution, via `yum`, etc. If your distribution does not offer the imagick package for PHP, install it via PECL (although you may have to resolve system dependencies):

```
sudo pecl install imagick
```
2. Verify that imagick is enabled in your PHP Apache module's PHP `.ini` file (as determined in the section above for P4 API for PHP). You may need to add the following line:

```
extension=imagick.so
```
3. Configure the ImageMagick `policy.xml` file to allow P4 Code Review to create thumbnails if it is not already configured:
 - a. Open the ImageMagick `policy.xml` file for editing:

```
/etc/ImageMagick-6/policy.xml
```
 - b. Disable lines with the pattern below for the following `<filetype>s`: PS, PS2, EPS, PDF, or XPS:

```
<policy domain="coder" rights="none" pattern="<filetype>" />
```
 - c. Save the `policy.xml` file.
4. Restart Apache for the changes to become active.
5. To verify that imagick is active, navigate to the `phpinfo` file you [created earlier](#). You should then see an imagick section. It should report its version information and a table for its directives, supported image file formats, and more.

Warning:

Once you have completed installing and enabling P4 API for PHP and imagick, we recommend that you remove the `phpinfo` file you created to avoid disclosing information about your installation.

Ubuntu

1. Install the generic `php-imagick` module.
`# sudo apt install php-imagick`
2. Or install a specific version.
`# php --version`
PHP 8.1.2-1ubuntu2.19 (cli) (built: Sep 30 2024 16:25:25) (NTS)
`# sudo apt install php8.1-imagick`
3. Restart Apache for the changes to become active.
`# sudo systemctl restart apache2`
4. To verify that imagick is active, navigate to the `phpinfo` file you [created earlier](#). You should then see an imagick section. It should report its version information and a table for its directives, supported image file formats, and more.

Warning:

Once you have completed installing and enabling P4 API for PHP and imagick, we recommend that you remove the `phpinfo` file you created to avoid disclosing information about your installation.

Troubleshooting out of memory (OOM) issues

If you are running P4 Code Review servers running PHP 8.x on a RHEL platform, some PHP processes (called `php-fpm` threads) may get stuck or idle and fail to release memory. These are known as zombie or idle threads. Over time, these idle threads caused the server to run out of memory, sometimes crashing or requiring frequent restarts of the `php-fpm` service.

Workaround: Stop using `php-fpm` and instead run P4 Code Review with `mod_php` under **Apache prefork MPM**. To switch to `mod_php`, follow these steps:

1. Edit the Apache MPM configuration:
`sudo nano /etc/httpd/conf.modules.d/00-mpm.conf`
2. Enable prefork MPM by uncommenting this line:
`LoadModule mpm_prefork_module modules/mod_mpm_prefork.so`
3. Disable event MPM by commenting out this line:
`#LoadModule mpm_event_module modules/mod_mpm_event.so`
4. Restart Apache:
`sudo systemctl restart httpd`
5. Verify the active MPM:
`httpd -V | grep MPM`

You will see the following if you have successfully switched to `mod_php`:

Server MPM: prefork

P4 Code Review configuration

Now that P4 Code Review is ready for use, you need to configure it to work in your environment.

Important:

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

P4 Code Review can be connected to P4 Servers (P4D) and commit servers.

- To configure P4 Code Review to connect to more than one P4 Server (P4D), see ["Multiple P4 Server instances" on page 638](#).
- To configure P4 Code Review to connect to a P4 Server configured to use *commit-edge architecture*, see ["Commit-edge deployment" on page 582](#).

Important: Do not connect P4 Code Review to P4 Broker, P4 Proxy, an edge server, forwarding replica, or read-only replica servers.

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

Swarm configuration file

Important:

- If your P4 Server is configured for P4 AS, the P4 Server must be temporarily configured to allow fall-back to passwords while you establish a connection to the P4 Server. Run the following command on the P4 Server to enable fall-back to passwords:
`p4 configure set auth.sso.allow.passwd=1`

Edit the [SWARM_ROOT/data/config.php](#) file so that it contains the following configuration block:

```
<?php
return array(
    'p4' => array(
        'port'    => 'my-helix-core-server:1666',
        'user'     => 'admin_userid',
        'password' => 'admin user ticket or password',
        'sso'      => 'disabled', // 'disabled'|'optional'|'enabled'
                        // default value is 'disabled'
    ),
    'log' => array(
        'priority' => 3, // 7 for max, defaults to 3
    ),
    'mail' => array(
        'transport' => array(
```

```
        'host' => 'my.mx.host',  
    ),  
),  
);
```

port

For the port value, replace *my-helix-core-server:1666* with the P4PORT value used to connect to your P4 Server.

Important:

If your P4 Server is deployed using the commit-edge architecture, ensure that Swarm's port value points to the commit server.

For more information, see [Commit-edge](#) in the [P4 Server Administration Documentation](#).

Warning:

If the port points to a P4 Broker, ensure that the broker does not delegate commands to different replicas, edge servers, or proxies. Such delegation can cause odd problems or outright failures in Swarm.

Swarm needs to have a consistent, current view of the state of P4 Server, and works best when it connects to a central/commit server.

user

For the user value, replace `admin_userid` with a normal P4 Server userid that has `admin`-level access to P4 Server.

Note:

- This user is used by P4 Code Review to communicate with the P4 Server. It should not be used to perform everyday P4 Code Review reviewing tasks.
- P4 Server 2020.1 and later, permissions have changed for viewing and [editing stream spec files](#) in P4 Code Review. To view and edit stream spec files in P4 Code Review, the P4 Code Review user must have *admin* permissions for the entire depot `//...`

password

For the password value, while a plain-text password works, we recommend that you use a ticket value instead. Obtain the ticket value for the `admin_userid` during login with this command:

```
p4 -p my-helix-core-server:1666 -u admin_userid login -p
```


Important:

- If your P4 Server authentication is configured in the one of the following ways, ticket-based authentication is required:
 - Authentication configured with security level 3.
 - Authentication configured for LDAP.
 - Authentication configured for P4 AS.
- You must use a long-lived login ticket for the P4 Code Review user.

You can determine when the admin userid ticket will expire with:

```
p4 -p my-helix-core-server:1666 -u admin_userid -P ticket_value login -s
```

For more information about tickets, see the section [Ticket-based authentication](#) in the [P4 Server Administration Documentation](#).

sso

P4 Code Review can be configured for the P4 AS when the P4 Server is also configured for the P4 AS.

Set `sso` to one of the following:

- `enabled` all users must use P4 AS to log in to P4 Code Review.
The login page displays an input box to enter the email or username. Once the input box is filled, the **Log in with SSO** button is activated. See ["Log in with SSO" on page 384](#).
- `optional` P4 AS is available for users to log in to P4 Code Review but is not enforced.
The login page displays an input box to enter the email or username. Once the input box is filled, the **Log in with SSO** button is enabled. To log in with SSO, see ["Log in with SSO" on page 384](#).
The user can also manually log in to P4 Code Review using the **Log in with credentials** button. To log in manually, see ["Log in with a password" on page 383](#).
If you switch to **Log in with credentials** and then decide to use single sign-on, click **Log in with SSO**.
- `disabled` P4 AS is not available to P4 Code Review. This is the default value.
The login page displays two input boxes to enter the email or username, and to enter the password. Once the input boxes are filled, the **Log in with credentials** button is activated. See ["Log in with a password" on page 383](#).

Important:

From P4 Code Review 2021.1, the `sso_enabled` configurable is deprecated but remains supported. It is replaced with the more flexible `sso` configurable. If the `sso_enabled` configurable and `sso` configurable are both present in the `p4` configuration block, P4 Code Review uses the `sso` configurable value.

host

For the host value, replace `my.mx.host` with the hostname of the mail exchange service that P4 Code Review should use to send its email notifications.

Note:

Since this configuration file contains the credentials for a P4 Server *admin*-level user, we recommend that this file's ownership and permissions be adjusted such that only the web server user can read the file, and that no user can write the file.

Next step

Configure the P4 Server to notify P4 Code Review about P4 Server events, see ["Configuring P4 Server event notification" below](#).

Configuring P4 Server event notification

Tip:

Triggers are still supported, but we recommend you use P4 Server Extensions. P4 Server Extensions are easier to install and maintain than Triggers.

P4 Code Review needs to know about a number of P4 Server events to operate correctly, this can be done by using P4 Server Extensions (recommended) or P4 Server Triggers. P4 Code Review installs include the P4 Server extension file and trigger scripts required for P4 Code Review to get the events it needs from your P4 Server.

You must install and configure P4 Server Extensions or P4 Server Triggers to complete the P4 Code Review installation.

Do one of the following so that P4 Code Review is notified about events on the P4 Server:

- Install and configure the P4 Server extension, see ["Installing the P4 Code Review extension for P4 Server \(recommended\)" below](#)
- Install and configure P4 Server Triggers, see ["Installing triggers" on page 187](#)

Installing the P4 Code Review extension for P4 Server (recommended)

Important:

- If you are using the P4 Code Review extension to connect to the P4 Server, do not install P4 Code Review triggers.
- You must be a user with *super* user permissions to install and configure P4 Server Extensions.

P4 Code Review needs to know about a number of P4 Server events to operate correctly, this is done by using P4 Server Extensions. P4 Server Extensions must be installed and configured on your P4 Server to complete your P4 Code Review installation. The P4 Server extension is included with the P4 Code Review product download.

Prerequisites

To install the P4 Server extension you need:

A compatible version of P4 Server for the extension:

- **Linux:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.
- **Windows:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.

You will also need:

- P4 Server Extensions installed and configured on your P4 Server.
- A user with *super* permissions to install and configure the P4 Server extension.

P4 Server extension installation

Tip:

The following commands can be run on the P4 Code Review server or the P4 Server. You must be a P4 Server *super* user to install and configure P4 Server Extensions.

The P4 Server extension file is included for all of the P4 Code Review installation types. The `helix-swarm.p4-extension` file is located in "`swarm_root`" on [page 712](#)/p4-bin/extensions

Run the following commands as a P4 Server user with *super* permissions:

1. Install the P4 Server extension on the P4 Server with:

```
p4 extension --yes --install helix-swarm.p4-extension
```

Example response:

```
Extension 'Perforce::helix-swarm' version '2022.1.20221215' installed successfully.
Perform the following steps to turn on the Extension:
```

```
# Create a global configuration if one doesn't already exist.
p4 extension --configure Perforce::helix-swarm
```

```
# Create an instance configuration to enable the Extension.
p4 extension --configure Perforce::helix-swarm --name Perforce::helix-swarm-
instanceName
```

For more information, visit:
<https://www.perforce.com/manuals/extensions/Content/Extensions/Home-extensions.html>

2. Create a global configuration if one doesn't already exist:

```
p4 extension --configure Perforce::helix-swarm
```

The spec file opens in your text editor, for example:

```
ExtName:    helix-swarm
ExtDescription:
    Helix Swarm Extension

ExtVersion: 2022.1.DEV.20220120

ExtUUID:    4532BC59-7BC8-478F-ADF6-0A563C42563D

ExtRev: 1

ExtMaxScriptTime:    unset

ExtMaxScriptMem:     unset

ExtAllowedGroups:

ExtEnabled:  true

ExtP4USER:  sampleExtensionsUser

Name: helix-swarm

Owner: super

Update: 2022/01/21 10:43:46

Description:
    The description of your config.

ExtConfig:
    Debug:
        2
    Swarm-Secure:
        true
    Swarm-Token:
        SWARM-TOKEN
    Swarm-URL:
        http://localhost/
```

3. Edit the spec file to match your system:

- ExtP4USER: Set to an existing user with *super* permissions.
- ExtConfig: Check and edit the values in this block:
 - Debug:
Debug levels 0 to 3 control the amount of debug information sent to the P4 Server Extensions log.

Important:

A debug level of 10 and higher sends all debug information to the client. This is useful for debugging, but should not be run in a production environment.

- **SSL-CA-File:** (optional) Set the absolute path of the [Certificate Authority \(CA\) Privacy-Enhanced Mail \(PEM\)](#) file to validate the P4 Server's certificate for P4 Code Review. Ensure that this file is accessible to the P4 Server process owner.

If a suitable certificate is not found in the local SSL certificate store maintained by the operating system or if there are verification issues from using a self-signed certificate then the default CA file is used.

By default, the local certificate stored in the default CA file is used if the absolute path for the [Certificate Authority \(CA\)](#) is not specified.

- **Swarm-Secure:**
 - `true` will only accept secure SSL certificates.
 - `false` will accept insecure SSL certificates.
- **Swarm-Token:** Set to the P4 Code Review trigger token value.

Important:

To set multiple P4 Server token settings for Extensions, repeat the following steps for each P4 Server. The process to obtain an extension token of your P4 Code Review instance is similar to obtaining a trigger token of your P4 Code Review instance.

To obtain the trigger token of your P4 Code Review instance:

- a. Log in to P4 Code Review as a *super* user.
 - b. Click your *userid*, found at the right of the main toolbar.
 - c. Select **About P4 Code Review**.
 - d. The **About P4 Code Review** dialog is displayed and P4 Code Review generates an API token if it doesn't already exist.
 - e. Copy the trigger token value displayed at the bottom of the dialog and paste the token into the spec file.
- **Swarm-URL:** Set to the URL of your P4 Code Review instance.

4. When you have finished editing the spec file, save it in your text editor.

5. Create an instance configuration to enable the extension:

```
p4 extension --configure Perforce::helix-swarm --name swarm
```

The spec file opens in your text editor, for example:

ExtName: helix-swarm

ExtDescription:
Helix Swarm Extension

ExtVersion: 2022.1.DEV.20220120

ExtUUID: 4532BC59-7BC8-478F-ADF6-0A563C42563D

ExtRev: 1

ExtMaxScriptTime: unset

ExtMaxScriptMem: unset

ExtEnabled: true

ExtDebug: none

Name: swarm

Owner: super

Update: 2022/01/21 10:58:41

Description:
The description of your config.

ExtConfig:
depot-path:
//...
enableStrict:
true
enableWorkflow:
true
httpTimeout:
30
ignoreErrors:
false

We recommend that these settings are left at their default values, but the configuration needs to be run to set the default values.

- `enableStrict`: Enable strict checking of reviews for workflow. Verifies that the file content in a commit matches the file content of its associated approved review. If one or more files in a commit do not match the content of the file in its associated review, the commit is rejected.

- `enableWorkflow`: Enable workflow.
 - `httpTimeout`: Timeout (in seconds) when communicating with the P4 Code Review server.
 - `ignoreErrors`: If the server returns an error (timeout, not there), then default to allowing a request. It means workflow rules can be skipped, but if the P4 Code Review server is down it will not block submits.
6. Save the spec file in your text editor.
 7. Confirm your P4 Server extension is installed and configured by running the following command on the P4 Server:


```
p4 extension --run swarm ping
```

 - If the P4 Server extension is working, the P4 Server responds with `OK`
 - If the P4 Server extension is not working, the P4 Server responds with an error message. For example, `BAD (Cannot reach https://swarm-example.com/)`

For more information about P4 Server Extensions, see:

 - [Extension overview](#) in the [P4 Server Administration Documentation](#).
 - [p4 extension](#) in the [P4 CLI Reference](#).
 8. Depending on your installation type, do one of the following:
 - **P4 Code Review package installation:** Review the post-install configuration options to customize your P4 Code Review installation, see "[Post-install configuration options](#)" on page 214.
 - **P4 Code Review tarball installation:** Set up a recurring task to spawn workers, see "[Set up a recurring task to spawn workers](#)" on page 208.

Add P4 Code Review as an HTML tab in the P4 Visual Client (P4V)

To add Swarm as an HTML tab in the P4 Visual Client (P4V), see [HTML Tabs](#) section in the [P4VJS Developer Guide](#).

Installing triggers

With your P4 Code Review instance configured, the final step is to set up your P4 Server to notify your instance of important events. This is achieved using triggers.

Tip: We recommend using Extensions instead of installing triggers as these are easier to install and manage. For more information, see "[Installing the P4 Code Review extension for P4 Server \(recommended\)](#)" on page 182.

This document covers how to install triggers on P4 Code Review. For more information on how to configure P4 Server triggers, see the [P4 Server documentation](#).

P4 Code Review can be connected to P4 Servers (P4D) and commit servers.

- To configure P4 Code Review to connect to more than one P4 Server (P4D), see "[Multiple P4 Server instances](#)" on page 638.

- To configure P4 Code Review to connect to a P4 Server configured to use *commit-edge architecture*, see ["Commit-edge deployment" on page 582](#).

Important: Do not connect P4 Code Review to P4 Broker, P4 Proxy, an edge server, forwarding replica, or read-only replica servers.

P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

Prerequisites

Before installing P4 Code Review triggers, check the following conditions:

- Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.
- P4 Code Review does not support P4 Servers that are configured to use P4AUTH. For more information, see [Centralized authorization server \(P4AUTH\)](#) in the [P4 Server Administration Documentation](#).

Setup up P4 Code Review triggers

Use P4 Server triggers to customize the operation of the server. For example, use P4 Server triggers to automatically run custom scripts or commands when certain events happen, such as submitting a changelist or creating a branch.

P4 Code Review provides a trigger script, written in Perl, that notifies P4 Code Review about activity within the P4 Server.

For more information on configuring the Perl trigger script, see ["Trigger options" on page 723](#)

The operating system which hosts your P4 Server determines how to set up P4 Code Review triggers:

- ["Copy the Perl trigger script on to your P4 Server" on the facing page](#)
- ["Copy the Perl trigger script on to your P4 Server" on page 200](#)

Set up P4 Code Review triggers with a Windows-hosted P4 Server

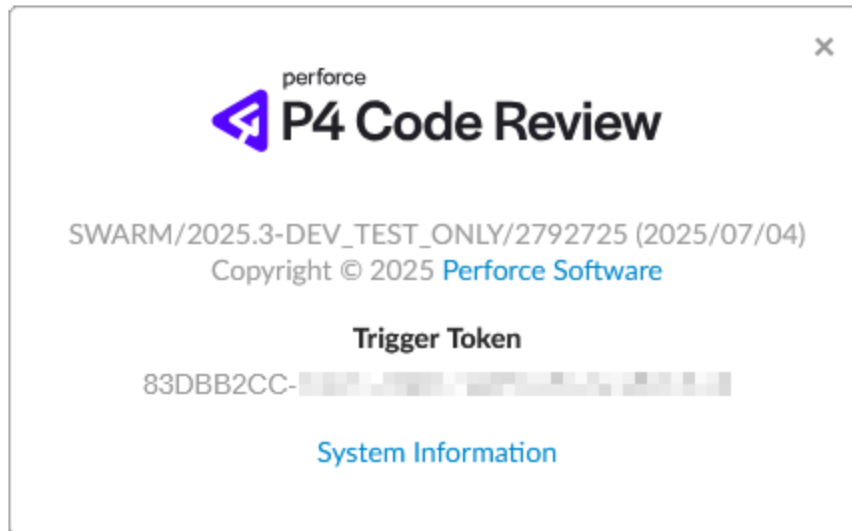
The operating system which hosts your P4 Server determines how to set up P4 Code Review triggers. This guide outlines how to set up P4 Code Review triggers with a Windows-hosted P4 Server.

Prerequisites

Complete the following steps before setting up P4 Code Review triggers:

- Check the required dependencies have been installed on the machine hosting the P4 Server. For more information, see ["Trigger dependencies" on page 100](#).
- Find your P4 Code Review API token. You need this token you configure P4 Server for P4 Code Review. To find this API token, follow these steps:
 1. Log in to P4 Code Review as a *super* user. For more information, see ["Log in/Log out" on page 383](#).

2. Click your *userid*, found at the right of the main toolbar.
3. Select **About P4 Code Review**. The **About P4 Code Review** dialog is displayed and P4 Code Review generates an API token if it doesn't already exist.



4. Copy the API token displayed at the bottom of the dialog. This token is the trigger token to enter in the `swarm-trigger.conf` file when you configure P4 Server for P4 Code Review.

Set up P4 Code Review triggers

As there are many steps involved in setting up P4 Code Review triggers, these have been broken into the following sections:

- "Copy the Perl trigger script on to your P4 Server" below
- "Configure the API token for the P4 Code Review trigger" on the next page
- "Ensure trigger script works" on page 192
- "Update the P4 Server triggers table" on page 193
- "Configure the P4 Server to promote all shelved changes" on page 198
- "Optional installation steps" on page 198

Copy the Perl trigger script on to your P4 Server

Follow these steps to locate and copy the Perl trigger script:

1. Find the P4 Code Review Perl trigger script, `swarm-trigger.pl`, in the following location: `p4-bin/scripts/swarm-trigger.pl`
2. Copy this Perl script to the following depot on your P4 Server: `/.swarm/triggers/swarm-trigger.pl`

Important: If you are using a version of P4 Code Review that is older than 2020.1, [contact support](#) for assistance on installing triggers.

Configure the API token for the P4 Code Review trigger

This section outlines how to add the API token to the `swarm-trigger.conf` configuration file. Ensure both the configuration file and the `swarm-trigger.pl` trigger script are in the following depot location on your P4 Server: `//.swarm/triggers/`

The following steps assume you have installed and configured P4 Code Review from a package. For more information, see ["Install and configure P4 Code Review from a package \(recommended\)"](#) on [page 106](#).

1. Find the P4 Code Review configuration file in the following location:
`/opt/perforce/etc/swarm-trigger.conf`.
2. Copy the `swarm-trigger.conf` file to the depot. The recommended depot location is `//.swarm/triggers/swarm-trigger.conf`

Important: If you installed P4 Code Review manually, you will also need to manually create the `swarm-trigger.conf` file.

Ensure the configuration file is in the same location as the `swarm-trigger.pl` file. The recommended depot location is `//.swarm/triggers/swarm-trigger.conf`

3. Modify the `swarm-trigger.conf` configuration file to update the following:
 - **SWARM_HOST:** Update this with the host name of the P4 Code Review instance. This starts with either `http://` or `https://`, for example: `SWARM_HOST="http://my-swarm-host"`
 - **SWARM_TOKEN:** Update this with the API token you copied before. To find your API token, see [the prerequisites above](#).

Below is a sample of the `swarm-trigger.conf`:

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or "https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional) Do not use if the Workflow feature is enabled (default)
# For enforcing reviewed changes, optionally specify the normal Perforce user
# with admin privileges (to read keys); if not set, will use whatever Perforce
# user is set in environment.
ADMIN_USER=
```

```
# ADMIN_TICKET_FILE (optional) Do not use if the Workflow feature is enabled
(default)
# For enforcing reviewed changes, optionally specify the location of the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very long
# timeout; then, manually login as this user from the Perforce server machine to
# set the ticket.
ADMIN_TICKET_FILE=

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls whether
# the SSL certificate is validated or not. By default this is set to 1, which
# means any SSL certificates must be valid. If the web server is using a self
# signed certificate, then this must be set to 0.
VERIFY_SSL=1
```

Troubleshooting: If workflows are disabled

From P4 Code Review version 2019.2 and later, workflows are enabled by default. However, if workflows are disabled, you need to provide an `ADMIN_USER` in the `swarm-trigger.conf` file.

The login for the `ADMIN_USER` is authenticated using a ticket file which is found in either of the following locations:

- On Windows, the file path is: `%USERPROFILE%\p4tickets.txt`
- In the `swarm-trigger.conf` file, the file path is set using `ADMIN_TICKET_FILE`.

If you provide an `ADMIN_USER`, the login ticket contain the exact same port number used by your P4 Server.

For example, if P4 Server is initiated with the following code:

```
p4d -p my-helix-core-server:1666 ...
```

Then the ticket for the user specified with `ADMIN_USER` should be established with:

```
p4 -p my-helix-core-server:1666 -u admin_userid login
```

If the ticket was established using the wrong port, the error message you encounter includes the port that the trigger is attempting to use. Below is an example of the error message:

```
'swarm.strict.1' validation failed: Invalid login credentials to [port] within this trigger script;
please contact your administrator
```

Depending on if workflows are enabled or disabled determines if certain triggers in the trigger table need to be commented out. For more information, see ["Update the P4 Server triggers table" on the facing page](#).

Troubleshooting: Copying `swarm-trigger.conf` when using edge servers

If you copied the `swarm-trigger.pl` trigger script to the commit server and all relevant edge servers in the previous step, also copy the `swarm-trigger.conf` configuration file to the commit server as well as the edge servers, making sure that the files exist in the same path on all servers.

Troubleshooting: HTTP::Tiny module is not installed.

If you do not have the HTTP::Tiny Perl module installed, edit the `swarm-trigger.pl` trigger script and specify the full path to `curl.exe`.

Ensure trigger script works

Verify that the trigger script executes correctly. On windows, open the command line and run the following:

```
C:\> perl "C:\<file_path>\swarm-trigger.pl" -t ping -v 0
```

Replace `<file_path>` with the file path to the `swarm-trigger.pl` trigger script.

If you do not see any output, then the trigger script works.

If you see an error, the trigger script needs to be reconfigured. For more information, see ["Configure the API token for the P4 Code Review trigger" on page 190](#).

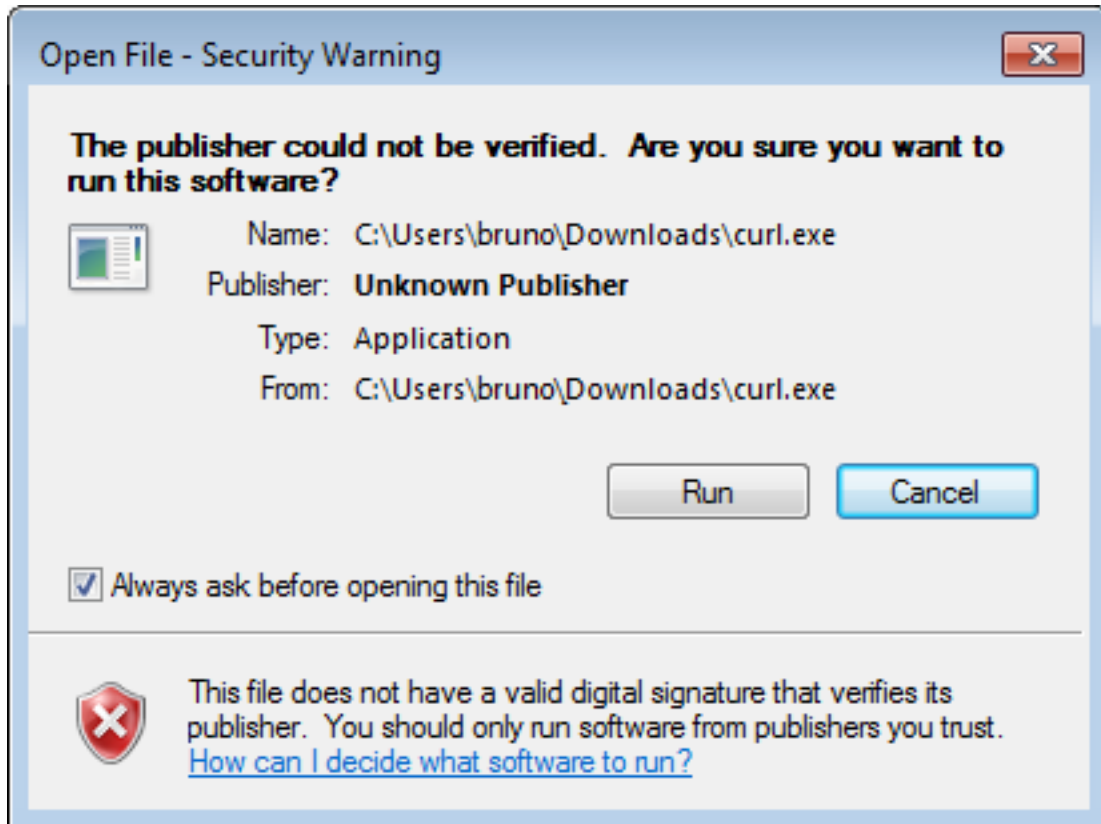
Troubleshooting: Curl may cause security warning.

Running the triggers script may cause a security warning dialog to appear when `curl.exe` is run.

This security prompt causes the `swarm-trigger.pl` trigger script to hang and creates zombie Perl processes (these are processes that are not doing any work but still exist).

To resolve this issue:

1. In the security prompt window, clear the **Always ask before opening this file** check box and click **Run**.



2. Go to the folder that contains `curl.exe`.
3. Right-click on `curl.exe`, select **Properties**, and click **Unblock**.
4. Re-run the `swarm-trigger.pl` trigger script.

Update the P4 Server triggers table

Update the P4 Server triggers table to run the trigger script. To access the table, enter `p4 triggers` in your command line.

Important: You must be a P4 Server user with *super* privileges to edit the table.

Add the following lines (make sure to include the initial tab character at the start of each line) to the trigger table, replacing the following variables with the corresponding file path:

- `<perl_path>` - the file path to `perl.exe`.
- `<trigger_path>` - the file path to `swarm-trigger.pl`.
- `<config_path>` - the file path to `swarm-trigger.conf`.

Depending on whether you committed the trigger script and configuration file to the P4 Server or copied them to common paths on all servers, you will need to update the trigger table accordingly.

Important: When updating the trigger table:

- An initial tabbed indent is required for each trigger line.
- Commented out trigger lines are not stored.

If you have committed both the trigger script and the configuration file to the P4 Server, add the following lines to the trigger table:

```

    swarm.job    form-commit job "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t job    -
v %formname%"
    swarm.user   form-commit user "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
user    -v %formname%"
    swarm.userdel form-delete user "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
userdel  -v %formname%"
    swarm.group  form-commit group "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
group    -v %formname%"
    swarm.groupdel form-delete group "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
groupdel -v %formname%"
    swarm.changesave form-save  change "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
changesave -v %formname%"
    swarm.shelve shelve-commit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
shelve   -v %change%"
    swarm.commit change-commit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
commit   -v %change%"
    swarm.shelvedel shelve-delete //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
shelvedel -v %change% -w %client% -u %user% -d
%quote%%clientcwd%^^^%quote% -a %quote%%argsQuoted%%quote% -s
%quote%%serverVersion%%quote%"
# The following three triggers are used by workflows.
# If workflows are enabled in P4 Code Review,
# then these triggers should also be disabled.
    swarm.enforce change-submit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
checkenforced -v %change% -u %user%"

```

```

swarm.strict  change-content //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
checkstrict -v %change% -u %user%"
swarm.shelvesub  shelve-submit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
checkshelve -v %change% -u %user%"
# The following triggers are only used to prevent a commit
# without an approved review. They predate the workflow
# functionality and should only be used if workflow is disabled.
# Support for these will be dropped in a later release.
# swarm.enforce.1 change-submit //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t enforce -v %change% -p %serverport%"
# swarm.enforce.2 change-submit //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t enforce -v %change% -p %serverport%"
# swarm.strict.1 change-content //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t strict -v %change% -p %serverport%"
# swarm.strict.2 change-content //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t strict -v %change% -p %serverport%"

```

If you have copied the trigger script and configuration file to common paths on all servers:

```

swarm.job  form-commit job "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t job -v %formname%"
swarm.user  form-commit user "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t user -v %formname%"
swarm.userdel  form-delete user "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t userdel -v %formname%"
swarm.group  form-commit group "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t group -v %formname%"
swarm.groupdel  form-delete group "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t groupdel -v %formname%"
swarm.changesave  form-save change "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t changesave -v %formname%"

```

```

swarm.shelve shelve-commit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t shelve -v %change%"
swarm.commit change-commit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t commit -v %change%"
swarm.shelvedel shelve-delete //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t shelvedel -v %change% -w %client% -u %user% -d
%quote%%clientcwd%^^^%quote% -a %quote%%argsQuoted%%%quote% -s
%quote%%serverVersion%%%quote%"
# The following three triggers are used by workflows.
# If workflows are enabled in P4 Code Review,
# then these triggers should also be disabled.
swarm.enforce change-submit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t checkenforced -v %change% -u %user%"
swarm.strict change-content //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t checkstrict -v %change% -u %user%"
swarm.shelvesub shelve-submit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t checkshelve -v %change% -u %user%"
# The following triggers are only used to prevent a commit
# without an approved review. They predate the workflow
# functionality and should only be used if workflow is disabled.
# Support for these will be dropped in a later release.
# swarm.enforce.1 change-submit //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t enforce -v %change% -p
%serverport%"
# swarm.enforce.2 change-submit //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t enforce -v %change% -p
%serverport%"
# swarm.strict.1 change-content //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t strict -v %change% -p
%serverport%"
# swarm.strict.2 change-content //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t strict -v %change% -p
%serverport%"

```


Important: If you are using a version of P4 Code Review that is older than 2020.1, [contact support](#) for assistance on installing triggers.

Troubleshooting: Workflows require triggers to be enabled or disabled

P4 Code Review workflow feature is enabled by default from version 2019.2 onwards. The trigger lines you need to configure in the P4 Server depend on whether this feature is enabled or disabled.

If you disable the workflow feature in the P4 Code Review `config.php` file, workflows are not processed by P4 Code Review but a small overhead is still incurred by the P4 Server each time it runs a workflow trigger script. This overhead can be eliminated by commenting out the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelfe-submit` workflow triggers.

If Workflow is enabled (default)

Ensure these trigger lines are present in the trigger table:

- `swarm.enforce change-submit`
- `swarm.strict change-content`
- `swarm.shelvesub shelfe-submit`

Comment out these trigger lines if they exist:

- `swarm.enforce.1`
- `swarm.enforce.2`
- `swarm.strict.1`
- `swarm.strict.2`

If Workflow is disabled

Comment out these trigger lines if they exist:

- `swarm.enforce change-submit`
- `swarm.strict change-content`
- `swarm.shelvesub shelfe-submit`

The following trigger lines are optional and should be commented out unless configured properly:

- `swarm.enforce.1`
- `swarm.enforce.2`
- `swarm.strict.1`
- `swarm.strict.2`

Notes on these optional triggers:

- The lines `swarm.enforce.1` and `.2` enforce that submitted changes must be tied to an approved review.

- The lines `swarm.strict.1` and `.2` enforce that changelist contents must match their approved review contents.
- You must configure the `DEPOT_PATH1` and `DEPOT_PATH2` values correctly for these triggers to work.
- Support for these triggers will be removed in future P4 Code Review releases.

Tip: If you want to apply the triggers for `enforce` or `strict` to additional depot paths, copy the existing lines and update the depot paths accordingly.

Configure the P4 Server to promote all shelved changes

Configuring this setting ensures P4 Code Review has access to shelved changelists (which is required for pre-commit reviews). Run the following in the command line to configure promote shelved changes:

```
p4 configure set dm.shelve.promote=1
```

To verify this setting, enter the following into the command line:

```
p4 configure show dm.shelve.promote
```

If the setting is configured correctly, the following text is returned.

```
dm.shelve.promote=1
```

If this setting is not configured and you are using an edge server, you must use the `-p` option when promoting shelved files to the commit server when initiating a pre-commit review. An example of which is shown below:

```
p4 shelve -p -c <changelist_number>
```

Optional installation steps

Depending on how you installed P4 Code Review and what other P4 products are available to you, consider the following:

- If you intend to use P4V and its P4 Code Review integration, consider using forwarding logins to the commit server. For more information, see ["P4V Authentication" on page 584](#).
- If you installed P4 Code Review using a package installation, review the post-install configuration options to customize your P4 Code Review installation. For more information, see ["Post-install configuration options" on page 214](#).

- If you installed P4 Code Review using a tarball installation, set up a recurring task to spawn workers. For more information, see ["Set up a recurring task to spawn workers" on page 208](#).

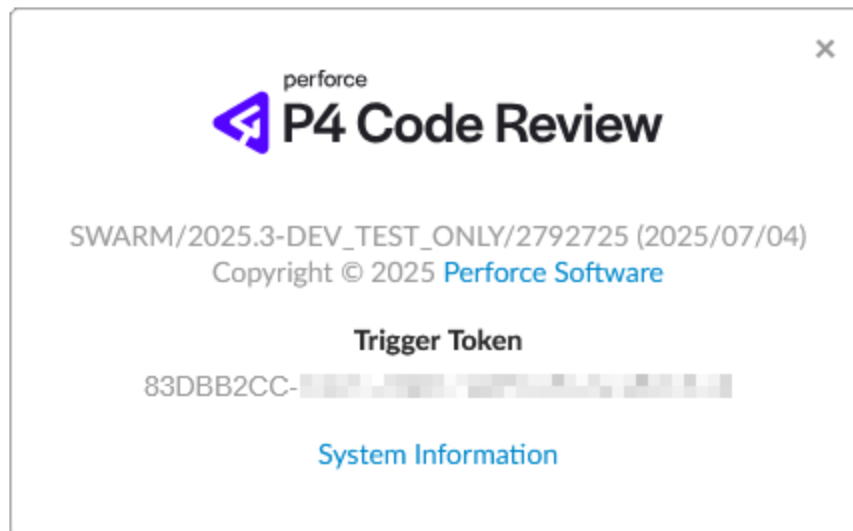
Set up P4 Code Review triggers with a Linux-hosted P4 Server

The operating system which hosts your P4 Server determines how to set up P4 Code Review triggers. This guide outlines how to set up P4 Code Review triggers with a Linux-hosted P4 Server.

Prerequisites

Complete the following steps before setting up P4 Code Review triggers:

- Check the required dependencies have been installed on the machine hosting the P4 Server. For more information, see ["Trigger dependencies" on page 100](#).
- Find your P4 Code Review API token. You need this token you configure P4 Server for P4 Code Review. To find this API token, follow these steps:
 1. Log in to P4 Code Review as a *super* user. For more information, see ["Log in/Log out" on page 383](#).
 2. Click your *userid*, found at the right of the main toolbar.
 3. Select **About P4 Code Review**. The **About P4 Code Review** dialog is displayed and P4 Code Review generates an API token if it doesn't already exist.



4. Copy the API token displayed at the bottom of the dialog. This token is the trigger token to enter in the `swarm-trigger.conf` file when you configure P4 Server for P4 Code Review.

Set up P4 Code Review triggers

As there are many steps involved in setting up P4 Code Review triggers, these have been broken into the following sections:

- "Copy the Perl trigger script on to your P4 Server" below
- "Configure the API token for the P4 Code Review trigger" below
- "Ensure trigger script works" on page 202
- "Update the P4 Server triggers table" on page 203
- "Configure the P4 Server to promote all shelved changes" on page 207
- "Optional installation steps" on page 208

Copy the Perl trigger script on to your P4 Server

Follow these steps to locate and copy the Perl trigger script:

1. Find the P4 Code Review Perl trigger script, `swarm-trigger.pl`, in the following location: `p4-bin/scripts/swarm-trigger.pl`
2. Copy this Perl script to the following depot on your P4 Server: `/.swarm/triggers/swarm-trigger.pl`

Important: If you are using a version of P4 Code Review that is older than 2020.1, [contact support](#) for assistance on installing triggers.

Configure the API token for the P4 Code Review trigger

This section outlines how to add the API token to the `swarm-trigger.conf` configuration file. Both the configuration file and the `swarm-trigger.pl` trigger script should be in the depot.

The following steps assume you have installed and configured P4 Code Review from a package. For more information, see ["Install and configure P4 Code Review from a package \(recommended\)"](#) on page 106.

1. Find the P4 Code Review configuration file in the following location:
`/opt/perforce/etc/swarm-trigger.conf`.
2. Copy the `swarm-trigger.conf` file to the depot. The recommended depot location is `/.swarm/triggers/swarm-trigger.conf`

Important: If you installed P4 Code Review manually, you will also need to manually create the `swarm-trigger.conf` file.

Ensure the configuration file is in the same location as the `swarm-trigger.pl` file. The recommended depot location is `/.swarm/triggers/swarm-trigger.conf`

3. Modify the `swarm-trigger.conf` configuration file to update the following:
4.
 - **SWARM_HOST:** Update this with the host name of the P4 Code Review instance. This starts with either `http://` or `https://`, for example: `SWARM_HOST="http://my-swarm-host"`
 - **SWARM_TOKEN:** Update this with the API token you copied before. To find your API token, see [the prerequisites above](#).

Below is a sample of the `swarm-trigger.conf`:

```

# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or "https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional) Do not use if the Workflow feature is enabled (default)
# For enforcing reviewed changes, optionally specify the normal Perforce user
# with admin privileges (to read keys); if not set, will use whatever Perforce
# user is set in environment.
ADMIN_USER=

# ADMIN_TICKET_FILE (optional) Do not use if the Workflow feature is enabled
# (default)
# For enforcing reviewed changes, optionally specify the location of the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very long
# timeout; then, manually login as this user from the Perforce server machine to
# set the ticket.
ADMIN_TICKET_FILE=

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls whether
# the SSL certificate is validated or not. By default this is set to 1, which
# means any SSL certificates must be valid. If the web server is using a self
# signed certificate, then this must be set to 0.
VERIFY_SSL=1

```

When using Linux, `swarm-trigger.pl` looks for configuration in the following files.

- `/etc/perforce/swarm-trigger.conf`
- `/opt/perforce/etc/swarm-trigger.conf`
- The `swarm-trigger.conf` file stored in the same directory as `swarm-trigger.pl`.
- Any file passed to the `swarm-trigger.pl` script using the `-c` command line argument.

Variables defined in the later files override the earlier defined variables of the same name, for example, variables set inside the `swarm-trigger.pl` script itself.

Troubleshooting: If workflows are disabled

From P4 Code Review version 2019.2 and later, workflows are enabled by default. When workflows are enabled, you need to provide an `ADMIN_USER` in the `swarm-trigger.conf` file.

The login for the `ADMIN_USER` is authenticated using a ticket file which is found in either of the following locations:

- On Windows, the file path is: `%USERPROFILE%\p4tickets.txt`
- In the `swarm-trigger.conf` file, the file path is set using `ADMIN_TICKET_FILE`.

If you provide an `ADMIN_USER`, the login ticket contain the exact same port number used by your P4 Server.

For example, if P4 Server is initiated with the following code:

```
p4d -p my-helix-core-server:1666 ...
```

Then the ticket for the user specified with `ADMIN_USER` should be established with:

```
p4 -p my-helix-core-server:1666 -u admin_userid login
```

If the ticket was established using the wrong port, the error message you encounter includes the port that the trigger is attempting to use. Below is an example of the error message:

'swarm.strict.1' validation failed: Invalid login credentials to [port] within this trigger script; please contact your administrator

Troubleshooting: Copying `swarm-trigger.conf` when using edge servers

If you copied the `swarm-trigger.pl` trigger script to the commit server and all relevant edge servers in the previous step, also copy the `swarm-trigger.conf` configuration file to the commit server as well as the edge servers, making sure that the files exist in the same path on all servers.

Ensure trigger script works

Important: Skip this step if you have committed the trigger script to the P4 Server.

The `swarm-trigger.pl` trigger script requires execute permissions. Run the following command in the terminal to grant the script the relevant permissions:

```
$ chmod +x /<file_path>/swarm-trigger.pl
```

Replace `<file_path>` with the file path to the `swarm-trigger.pl` trigger script.

Verify that the trigger script executes correctly. On Linux, open the terminal and run the following:

```
$ /<file_path>/swarm-trigger.pl -t ping -v 0
```

If you do not see any output, then the trigger script works.

If you see an error, the trigger script is needs to be reconfigured. For more information, see ["Configure the API token for the P4 Code Review trigger" on page 200](#).

Update the P4 Server triggers table

Update the P4 Server triggers table to run the trigger script. To access the table, enter `p4 triggers` in your command line.

Important: You must be a P4 Server user with *super* privileges to edit the table.

Add the following lines (make sure to include the initial tab character at the start of each line) to the trigger table, replacing the following variables with the corresponding file path:

- `<perl_path>` - the file path to `perl.exe`.
- `<trigger_path>` - the file path to `swarm-trigger.pl`.
- `<config_path>` - the file path to `swarm-trigger.conf`.

Depending on whether you committed the trigger script and configuration file to the P4 Server or copied them to common paths on all servers, you will need to update the trigger table accordingly.

Important: When updating the trigger table:

- An initial tabbed indent is required for each trigger line.
- Commented out trigger lines are not stored.

If you have committed both the trigger script and the configuration file to the P4 Server, add the following lines to the trigger table

```
swarm.job    form-commit job "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t job    -
v %formname%"
swarm.user   form-commit user "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
user       -v %formname%"
swarm.userdel form-delete user "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
userdel    -v %formname%"
swarm.group  form-commit group "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
group      -v %formname%"
```

```

    swarm.grouppdel form-delete group "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
grouppdel -v %formname%"
    swarm.changesave form-save change "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
changesave -v %formname%"
    swarm.shelve shelve-commit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
shelve -v %change%"
    swarm.commit change-commit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
commit -v %change%"
    swarm.shelvedel shelve-delete //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
shelvedel -v %change% -w %client% -u %user% -d
%quote%%clientcwd%^^^%quote% -a %quote%%argsQuoted%%quote% -s
%quote%%serverVersion%%quote%"
# The following three triggers are used by workflows.
# If workflows are enabled in P4 Code Review,
# then these triggers should also be disabled.
    swarm.enforce change-submit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/triggers/swarm-trigger.conf% -t
checkenforced -v %change% -u %user%"
    swarm.strict change-content //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
checkstrict -v %change% -u %user%"
    swarm.shelvesub shelve-submit //... "%quote%<perl_path>\perl.exe%quote%
%<trigger_path>/swarm-trigger.pl% -c %<config_path>/swarm-trigger.conf% -t
checkshelve -v %change% -u %user%"
# The following triggers are only used to prevent a commit
# without an approved review. They predate the workflow
# functionality and should only be used if workflow is disabled.
# Support for these will be dropped in a later release.
# swarm.enforce.1 change-submit //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t enforce -v %change% -p %serverport%"
# swarm.enforce.2 change-submit //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t enforce -v %change% -p %serverport%"
# swarm.strict.1 change-content //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t strict -v %change% -p %serverport%"

```



```
# swarm.strict.2 change-content //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %<trigger_path>/swarm-trigger.pl% -c %<config_
path>/swarm-trigger.conf% -t strict -v %change% -p %serverport%"
```

If you have copied the trigger script and configuration file to common paths on all servers:

```
swarm.job    form-commit job "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t job    -v %formname%"
swarm.user   form-commit user "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t user   -v %formname%"
swarm.userdel form-delete user "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t userdel -v %formname%"
swarm.group  form-commit group "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t group  -v %formname%"
swarm.groupdel form-delete group "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t groupdel -v %formname%"
swarm.changesave form-save change "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t changesave -v %formname%"
swarm.shelve  shelve-commit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t shelve  -v %change%"
swarm.commit  change-commit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t commit   -v %change%"
swarm.shelvedel shelve-delete //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t shelvedel -v %change% -w %client% -u %user% -d
%quote%%clientcwd%^^^%quote% -a %quote%%argsQuoted%%quote% -s
%quote%%serverVersion%%quote%"
# The following three triggers are used by workflows.
# If workflows are enabled in P4 Code Review,
# then these triggers should also be disabled.
swarm.enforce change-submit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t checkenforced -v %change% -u %user%"
```

```

    swarm.strict  change-content //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t checkstrict -v %change% -u %user%"
    swarm.shelvesub shelve-submit //... "%quote%<perl_path>\perl.exe%quote%
%quote%<trigger_path>/swarm-trigger.pl%quote% -c %quote%<config_path>/swarm-
trigger.conf%quote% -t checkshelve -v %change% -u %user%"
# The following triggers are only used to prevent a commit
# without an approved review. They predate the workflow
# functionality and should only be used if workflow is disabled.
# Support for these will be dropped in a later release.
# swarm.enforce.1 change-submit //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t enforce -v %change% -p
%serverport%"
# swarm.enforce.2 change-submit //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t enforce -v %change% -p
%serverport%"
# swarm.strict.1 change-content //DEPOT_PATH1/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t strict -v %change% -p
%serverport%"
# swarm.strict.2 change-content //DEPOT_PATH2/... "%quote%<perl_
path>\perl.exe%quote% %quote%<trigger_path>/swarm-trigger.pl%quote% -c
%quote%<config_path>/swarm-trigger.conf%quote% -t strict -v %change% -p
%serverport%"

```

Important: If you are using a version of P4 Code Review that is older than 2020.1, [contact support](#) for assistance on installing triggers.

Troubleshooting: Workflows require triggers to be enabled or disabled

P4 Code Review workflow feature is enabled by default from version 2019.2 onwards. The trigger lines you need to configure in the P4 Server depend on whether this feature is enabled or disabled.

If you disable the workflow feature in the P4 Code Review `config.php` file, workflows are not processed by P4 Code Review but a small overhead is still incurred by the P4 Server each time it runs a workflow trigger script. This overhead can be eliminated by commenting out the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` workflow triggers.

If Workflow is enabled (default)

Comment out these trigger lines if they exist:

- `swarm.enforce.1`
- `swarm.enforce.2`
- `swarm.strict.1`
- `swarm.strict.2`

Ensure these trigger lines are present in the trigger table:

- `swarm.enforce change-submit`
- `swarm.strict change-content`
- `swarm.shelvesub shelve-submit`

If Workflow is disabled

Comment out these trigger lines if they exist:

- `swarm.enforce change-submit`
- `swarm.strict change-content`
- `swarm.shelvesub shelve-submit`

The following trigger lines are optional and should be commented out unless configured properly:

- `swarm.enforce.1`
- `swarm.enforce.2`
- `swarm.strict.1`
- `swarm.strict.2`

Notes on these optional triggers:

- The lines `swarm.enforce.1` and `.2` enforce that submitted changes must be tied to an approved review.
- The lines `swarm.strict.1` and `.2` enforce that changelist contents must match their approved review contents.
- You must configure the `DEPOT_PATH1` and `DEPOT_PATH2` values correctly for these triggers to work.
- Support for these triggers will be removed in future P4 Code Review releases.

Tip: If you want to apply the triggers for enforce or strict to additional depot paths, copy the existing lines and update the depot paths accordingly.

Configure the P4 Server to promote all shelved changes

Configuring this setting ensures P4 Code Review has access to shelved changelists (which is required for pre-commit reviews). Run the following in the command line to configure promote shelved changes:

```
p4 configure set dm.shelve.promote=1
```

To verify this setting, enter the following into the command line:

```
p4 configure show dm.shelve.promote
```

If the setting is configured correctly, the following text is returned.

```
dm.shelve.promote=1
```

If this setting is not configured and you are using an edge server, you must use the `-p` option when promoting shelved files to the commit server when initiating a pre-commit review. An example of which is shown below:

```
p4 shelve -p -c <changelist_number>
```

Optional installation steps

Depending on how you installed P4 Code Review and what other P4 products are available to you, consider the following:

- If you intend to use P4V and its P4 Code Review integration, consider using forwarding logins to the commit server. For more information, see ["P4V Authentication" on page 584](#).
- If you installed P4 Code Review using a package installation, review the post-install configuration options to customize your P4 Code Review installation. For more information, see ["Post-install configuration options" on page 214](#).
- If you installed P4 Code Review using a tarball installation, set up a recurring task to spawn workers. For more information, see ["Set up a recurring task to spawn workers" below](#).

Set up a recurring task to spawn workers

To ensure that incoming P4 Server events are automatically processed by P4 Code Review, it is important to set up a cron job to do this. We recommend that the cron job is installed on the P4 Code Review host machine.

Install curl or wget

You must install either **curl** or **wget** on the P4 Code Review server machine to run the recurring task that spawns the P4 Code Review workers:

- To install **curl**, see ["Install curl" on the facing page](#).
- To install **wget**, see ["Install wget" on the facing page](#).

Install curl

1. Download **curl** from: [curl download page](#)
2. Install **curl**.
3. Verify that **curl** is installed, see ["Verify that curl or wget is installed" below](#).

If **curl** cannot execute as expected, trigger execution may block or fail. Prior to configuring the [triggers](#), verify that **curl** executes, run:

```
$ curl -h
```

Below is an example of the start of the output:

```
Usage: curl [options...] <url>
Options: (H) means HTTP/HTTPS only, (F) means FTP only
--anyauth    Pick "any" authentication method (H)
-a, --append  Append to target file when uploading (F/SFTP)
--cacert FILE CA certificate to verify peer against (SSL)
--capath DIR  CA directory to verify peer against (SSL)
...[truncated for brevity]...
```

For a more thorough test that fetches content over a network, try the following test:

```
$ curl https://www.perforce.com/
```

The output should look like HTML.

Install wget

1. Download **wget** from: [wget download page](#)
2. Install **wget**.
3. Verify that **wget** is installed, see ["Verify that curl or wget is installed" below](#).

Verify that curl or wget is installed

Important: curl or wget must be installed or workers do not spawn and P4 Code Review cannot process any events.

Verify that curl or wget is installed:

1. Use the **which** command.
For example to verify that **curl** is installed:

```
$ which curl
```

2. If you see any output, the referenced command is installed.

Create a recurring task to spawn workers

Note:

P4 Code Review package installation:

- The `helix-swarm`, `swarm-cron-hosts.conf`, and `swarm-cron.sh` files are automatically created with the correct content including the correct URL for P4 Code Review. They do not need to be edited.
- If you want to run P4 Code Review in a sub-folder of an existing website, or with a custom port, the `swarm-cron-host.conf` file must be edited. For instructions on how to configure P4 Code Review to run in a sub-folder, or with a custom port, see [Run Swarm in a sub-folder of an existing web site](#), or [Run Swarm's virtual host on a custom port](#).

helix-swarm cron file

The `helix-swarm` file is located in the `/etc/cron.d` and points to the `swarm-cron.sh` script file.

1. Create a file named `helix-swarm` in `/etc/cron.d` if it does not already exist.
2. Edit the `helix-swarm` file so that it has the following content:

```
#
# Cron job to start Swarm workers every minute
#
* * * * * nobody [ -x /opt/perforce/swarm/p4-bin/scripts/swarm-cron.sh ] &&
/opt/perforce/swarm/p4-bin/scripts/swarm-cron.sh
```

3. Save the `helix-swarm` file.

swarm-cron.sh script

The `swarm-cron.sh` file in `/opt/perforce/swarm/p4-bin/scripts` ensures that a worker is fired up every minute.

By default, the `DEFAULT_CONFIG_FILE=` configuration line is set to `/opt/perforce/etc/swarm-cron-hosts.conf`. The `swarm-cron.sh` script takes the P4 Code Review hostname from the `swarm-cron-hosts.conf` file.

The `swarm-cron.sh` script contains the correct content for P4 Code Review installations using `wget`, or `curl`.

Tip:

You must leave the `wget`, and `curl` configuration lines in the script. The script is written so that it can be used with P4 Code Review installations that use `wget`, and P4 Code Review installations that use `curl`. The `wget` configuration line is tried first, if that fails the `curl` configuration line is tried.

- `wget` configuration line for HTTP, and HTTPS:

```
wget --quiet --no-check-certificate --output-document /dev/null --timeout 5 "${SWARM_
HOST}/queue/worker"
```

- `curl` configuration line for HTTP, and HTTPS:

```
curl --silent --insecure --output /dev/null --max-time 5 "${SWARM_
HOST}/queue/worker"
```

In the `wget` and `curl` configuration lines above, where you see `--timeout 5` or `--max-time 5`, the `5` is the number of seconds that the cron task will wait for a response from the P4 Code Review host. When the cron task is installed on the P4 Code Review host, that value could be reduced to `1` second (e.g. `--timeout 1` or `--max-time 1`).

Note:

If you configure P4 Code Review to use HTTPS, and you install a self-signed certificate, the cron jobs avoid the certificate validity test which could cause silent failures to process events. The command to avoid the certificate validity test is included by default for `wget`, and `curl`:

- `wget` command: `--no-check-certificate`
- `curl` command: `--insecure`

If you have configured an HTTP connection to P4 Code Review, the avoid validity check command is ignored. For more information about the risks of using a self-signed SSL certificate, see ["Security risks of using a self-signed certificate" on page 287](#).

`swarm-cron-hosts.conf` configuration file

The `swarm-cron-hosts.conf` file in `/opt/perforce/etc` specifies the connection type (HTTP or HTTPS), hostname, and port number for P4 Code Review cron jobs.

1. Edit the `swarm-cron-hosts.conf` file so that it contains the actual P4 Code Review hostname, and port you have configured for P4 Code Review (which may include a [sub-folder](#) or a [custom port](#)). The following format is used:

```
[http[s]://]<swarm-host>[:<port>]
```

Default if value not specified:

- `[http[s]://]` http
 - `<swarm-host>` must be specified
 - `:<port>` 80
2. Save the edited file. If the recurring task is disabled, or stops functioning for any reason, logged-in users will see an error message when P4 Code Review detects that no workers are running.

Tip:
To check or modify P4 Code Review worker configuration, see ["Workers" on page 740](#).

3. The basic P4 Code Review configuration is now complete.

Important:
If your P4 Server is configured for P4 AS, you can force all of your users to authenticate via your Identity Provider (IdP) by disabling fall-back to passwords. To disable fall-back to passwords on the P4 Server, run the following command:
`p4 configure set auth.sso.allow.passwd=0`

4. Review the post-install configuration options to customize your P4 Code Review installation, see ["Post-install configuration options" on page 214](#).

Set up a cron job to delete AI summaries

AI summaries are stored as P4 keys in the Perforce database. By default, the AI summaries are kept for 30 days before being permanently deleted from P4 Code Review.

To ensure that the AI summaries are removed on a schedule, a cron job is required. The cron job that deletes these AI summaries runs at 6:00 PM on the 28th of every month and runs in the P4 Code Review server's time zone.

P4 Code Review admin credentials must be added to the cron job in order to successfully delete the AI summaries.

Do I need to manually add the cron job to P4 Code Review?

Use the table below to determine if you need to manually add the cron job to the AI summaries.

P4 Code Review version	Installation method	Do I need to manually add cron job?
Upgrading to 2025.2	Package	Yes
Upgrading to 2025.2	Docker	No
Upgrading to 2025.2	Tarball	Yes
Installing 2025.2 and onwards	Package	No

P4 Code Review version	Installation method	Do I need to manually add cron job?
Installing 2025.2 and onwards	Docker	No
Installing 2025.2 and onwards	Tarball	Yes

Prerequisites

Before adding the cron job to delete the AI summaries, ensure you have completed the following.

- Installed either **curl** or **wget** on the P4 Code Review server machine that runs the cron job.
 - To install **curl**, see [Install curl](#).
 - To install **wget**, see [Install wget](#).
- Create a file named `helix-swarm` in `/etc/cron.d` if it does not already exist.
- Ensure you have the following information:
 - `<ADMIN_USER>` - This is the user name of the P4 Code Review admin.
 - `<TICKET>` - This is the ticket associated with the P4 Code Review admin user. This is 30 characters hexadecimal string. For example, `12524CCEA6C43BC6EC081C9EA73WE3`.

Tip: To find the `<TICKET>`, enter the following into a command prompt.

```
p4 -u <username> login -p
```

Replace `<username>` with your P4 Code Review user name.

- `<DOMAIN>` - This is either `http` or `https`.
 - If you have SSL enabled in your P4 Code Review instance, enter `https`.
 - If you have not enabled SSL, enter `http`.

Create a cron for removal of AI summaries

1. Find and open the `helix-swarm` file. This file is located in the `/etc/cron.d`.
2. Edit the `helix-swarm` file so that it has the following content:

```
#  
# Cron job to delete AI summaries  
#  
0 18 28 * * nobody curl -X DELETE -u <ADMIN_USER>:<TICKET>  
<DOMAIN>://<HOSTNAME>[:<PORT>]/api/v11/AiAnalysis/removeAiSummaries
```

Tip: The <PORT>credential may not be applicable in your API URL.

3. Save the helix-swarm file.

The cron to remove the AI summaries is now set up and will run at 6:00 PM on the 28th of every month. This will run in the server's timezone.

Common post-installation options

This chapter covers the post-installation options that are common for all installation processes.

In this section:

Post-install configuration options

There are a few options for customizing your P4 Code Review installation's operation. This section covers the options that are officially supported.

This section contains:

- ["No customization required" below](#)
- ["Back up your P4 Code Review virtual host first" on the facing page](#)
- ["HTTPS" on the facing page](#)
- ["Run P4 Code Review in a sub-folder of an existing web site" on page 219](#)
- ["Run the P4 Code Review virtual host on a custom port" on page 222](#)
- ["Use your own Redis server" on page 224](#)
- ["Hiding P4 Code Review storage from regular users" on page 226](#)
- ["Handling Exclusive Locks" on page 227](#)

No customization required

If you do not need to customize your P4 Code Review installation, your P4 Code Review installation is complete but you must check that it is working correctly before using it in production. For instructions on how to check your P4 Code Review installation, see ["Validate your P4 Code Review installation" on page 227](#).

Back up your P4 Code Review virtual host first

Before undertaking any of the following customization options, ensure that you have backed up your P4 Code Review virtual host configuration. Choose the most appropriate option:

- If your Apache configuration directory contains the directories `sites-available` and `sites-enabled`:

```
$ cd /path/to/apache/configuration/..
$ cp -a sites-available sites-available.bak
```

Important:

If the `sites-enabled` directory contains files, and not just symbolic links, you need to backup this folder as well:

```
$ cd /path/to/apache/configuration/..
$ cp -a sites-enabled sites-enabled.bak
```

- For RHEL systems, if you used the [P4 Code Review packages](#) to install P4 Code Review:

```
$ cd /path/to/apache/configuration/..
$ cp -a conf.d conf.d.bak
```

- Otherwise, back up your Apache configuration.

HTTPS

Important:

Back up your P4 Code Review virtual host configuration before customizing your P4 Code Review installation, see ["Back up your P4 Code Review virtual host first"](#) above.

This section describes how to make your P4 Code Review installation more secure by using HTTPS.

Before you begin the following procedure, locate your system's Apache configuration. Common configuration directories include:

- `/etc/httpd/conf/`
- `/etc/apache2/`
- `/usr/local/apache2/conf/`
- `/Applications/XAMPP/etc/`

Within the Apache configuration path, the main Apache configuration file is usually named one of the following:

- `httpd.conf`
- `apache2.conf`

A longer discussion on the possible locations and names of Apache configuration files is available here: [DistrosDefaultLayout](#)

1. Enable SSL in Apache.

If the Apache utility `a2enmod` is installed:

```
$ sudo a2enmod ssl
```

Without the `a2enmod` utility, edit the Apache configuration file by hand. Locate your Apache configuration file for modules and either uncomment or add the following lines:

```
LoadModule ssl_module libexec/apache2/mod_ssl.so
```

2. Create a directory to store certificates.

```
$ sudo mkdir -p /etc/apache2/ssl
```

3. Create a certificate/key pair.

```
$ cd /etc/apache2/ssl
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout apache.key -
out apache.crt
```

This command generates a private key and a certificate. To form the certificate, `openssl` prompts you for several details:

```
Generating a 2048 bit RSA private key
.....+++
.....+++
writing new private key to 'apache.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:CA
State or Province Name (full name) [Some-State]:British Columbia
Locality Name (eg, city) []:Victoria
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Perforce Software
Organizational Unit Name (eg, section) []:P4 Code Review development team
Common Name (e.g. server FQDN or YOUR name) []:myswarm.host
Email Address []:admin@myswarm.host
```

The output above includes some example details. You should replace anything in italics with your own details. Since the certificate request details that can help users determine whether your certificate is valid, enter legitimate information whenever possible.

Important:

The Common Name field must match the hostname for your P4 Code Review installation exactly.

4. Secure the certificate directory.

```
$ sudo chmod 600 /etc/apache2/ssl
```

5. Edit the virtual host configuration.

Note:

The virtual host configuration should be in the file you [backed up initially](#).

Edit the virtual host configuration to match:

```
<VirtualHost *:80>
    ServerName myswarm
    AllowEncodedSlashes NoDecode
    ServerAlias myswarm.host
    ErrorLog "/path/to/apache/logs/myswarm.error_log"
    CustomLog "/path/to/apache/logs/myswarm.access_log" common
    DocumentRoot "/path/to/swarm/public"
    <Directory "/path/to/swarm/public">
        AllowOverride All
        Require all granted
    </Directory>

    Redirect / https://myswarm.host/
</VirtualHost>

<VirtualHost *:443>
    SSLEngine on
    SSLCertificateFile /etc/apache2/ssl/apache.crt
    SSLCertificateKeyFile /etc/apache2/ssl/apache.key

    ServerName myswarm.host
    AllowEncodedSlashes NoDecode
    ServerAlias myswarm
    ErrorLog "/path/to/apache/logs/myswarm.error_log"
    CustomLog "/path/to/apache/logs/myswarm.access_log" common
    DocumentRoot "/path/to/swarm/public"
    <Directory "/path/to/swarm/public">
        AllowOverride All
        Require all granted
    </Directory>
</VirtualHost>
```

See Apache's virtual host documentation for details: [Apache Virtual Host documentation](#)

6. Customize the virtual host definition.
 - a. Replace `myswarm.host` with the hostname for P4 Code Review on your network.
 - b. Replace `myswarm` with the name of the subdomain hosting P4 Code Review. Many administrators choose swarm.

Note:

The string `myswarm` in the log file paths: this should match the subdomain name and prefix for the log files, to help coordinate the active host with the log files for that host. Doing this is particularly useful when your Apache server hosts multiple instances of P4 Code Review.

- c. Replace `/path/to/apache/logs` with the path where your Apache store its log files. Apache's log files are typically named `access_log` and `error_log`.
 - d. Replace `/path/to/swarm` with the path to the P4 Code Review directory.

7. Restart your web server.

```
$ sudo apachectl restart
```

8. Adjust your firewall configuration to allow connections to the standard SSL port for web servers.

- For RHEL:

```
$ sudo firewall-cmd --zone=public --add-port=443/tcp --permanent
$ sudo systemctl reload firewalld
```

- For other distributions, consult with your network administrator or operating system documentation to determine how to adjust your firewall configuration.

9. Test your HTTPS URL from a web browser.

Important:

If the `myswarm.host` value in the virtual host configuration and the certificate do not match, the P4V integration with P4 Code Review fails with the message `SSL handshake failed`.

Also, when a reverse DNS lookup is performed, `myswarm.host` should be the answer when querying for the P4 Code Review server's IP address.

10. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

Secure your P4 Code Review installation

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.

- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

Run P4 Code Review in a sub-folder of an existing web site

Warning:

Multiple P4 Server instances on a single P4 Code Review instance: You cannot run P4 Code Review in a sub-folder if P4 Code Review is connected to multiple P4 Servers.

Issue: P4 Code Review will lose connection to all of the P4 Servers if you edit the `base_url` configurable value in the environment block of `<swarm_root>/data/config.php`. This will stop your system working.

Fix: Remove the `base_url` configurable from the environment block of `<swarm_root>/data/config.php`.

Important:

Back up your P4 Code Review virtual host configuration before customizing your P4 Code Review installation, see ["Back up your P4 Code Review virtual host first" on page 215](#).

If you cannot run P4 Code Review in its own virtual host, which might be necessary when you do not control the hostname to be used with P4 Code Review, installing P4 Code Review in a sub-folder of an existing virtual host configuration can be a good solution.

Installing P4 Code Review in a sub-folder requires modification of the previous installation steps covered in this chapter:

- The ["Apache configuration" on page 171](#) is entirely different; instead of establishing a new virtual host, you need to [modify an existing virtual host configuration](#). Often, this would be Apache's default site.
- [P4 Code Review's configuration file](#) requires an [extra item](#).

The following sections cover the specifics of sub-folder installation.

See ["base_url" on page 604](#) for more details.

Important:

If you used the [P4 Code Review packages](#) to install P4 Code Review, you can adjust P4 Code Review's configuration using the package configuration script `/opt/perforce/swarm/sbin/configure-swarm.sh`.

`configure-swarm.sh` does not read any existing P4 Code Review configuration; you must provide all of the configuration details each time you execute `configure-swarm.sh`:

```
$ sudo /opt/perforce/swarm/sbin/configure-swarm.sh -n -p myp4host:1666 -u swarm -
w password -e mx.example.com -H myhost -B /swarm
```

In the example above, the `-B` option is used to specify the name of the sub-folder.

If you use `configure-swarm.sh` to adjust the Swarm configuration, you only need to follow the "[Apache configuration](#)" below steps described below; all of the changes listed in the "[P4 Code Review configuration](#)" on the facing page section below have been completed by `configure-swarm.sh`.

Apache configuration

1. Ensure that the `SWARM_ROOT` is not within the document root of the intended virtual host.
This step ensures that P4 Code Review's source code and configuration is impossible to browse, preventing access to important details such as stored credentials, and active sessions and workspaces.
2. Adjust the virtual host configuration that you are already using.

Note:

Depending on the method used to install P4 Code Review, the filename for virtual host configuration you need to edit is:

- For [P4 Code Review package](#) installations, edit `perforce-swarm-site.conf`.
- For manual installations following P4 Code Review's recommended "[Apache configuration](#)" on page 171 edit `swarm`.
- For other installations, you may have to edit `httpd.conf` or nearby files.

Add the following lines to the virtual host definition:

```
Alias "/swarm" "SWARM_ROOT/public"
```

```
DocumentRoot "/var/www/html"
```

```
<Directory "SWARM_ROOT/public">
  AllowOverride All
  Require all granted
</Directory>
```


The `Alias` line configures Apache to respond to requests to `https://myhost/swarm` with content from P4 Code Review's `public` folder. You can change the `/swarm` portion of the `Alias` line to anything you want.

The `DocumentRoot` line configures the Apache `DocumentRoot` directory location.

The `<Directory>` block grants access to everything within P4 Code Review's `public` folder. Replace `SWARM_ROOT` with the actual path to P4 Code Review.

3. Restart your web server.

```
$ sudo apachectl restart
```

P4 Code Review configuration

To successfully operate within a sub-folder, the `"swarm_root"` on [page 712](#)/`data/config.php` file needs to be adjusted to contain the following lines (as a peer of the `p4` item):

```
'environment' => array(
    'base_url' => '/swarm'
),
```

Ensure that `/swarm` matches the first item in the `Alias` line in the virtual host configuration.

See ["Environment" on page 602](#) for more details.

P4 Code Review trigger configuration

The `swarm-trigger.conf` file must be updated to include the `base_url` in the `SWARM_HOST` path.

Tip:

The location of the `swarm-trigger.conf` depends on your installation. For information on the location of the `swarm-trigger.conf` file, see ["Installing triggers" on page 187](#).

1. Edit the `swarm-trigger.conf` file.
2. Replace:

```
SWARM_HOST="http://my-swarm-host"
```

With:

```
SWARM_HOST="http://website-hostname/swarm"
```

3. Save the file.

Cron configuration

P4 Code Review's [recurring task](#) configuration must be updated to reflect the sub-folder that you have configured in [Apache's](#) and [P4 Code Review's](#) configurations. This is configured in the `swarm-cron-hosts.conf` file.

1. Edit `/opt/perforce/etc/swarm-cron-hosts.conf`.
2. Replace:

`https://swarm-host`

with:

`https://swarm-host/swarm/`

Where *swarm-host* is the hostname of your P4 Code Review installation, and *swarm* is the sub-folder you want to use.

3. Save the edited file.

New workers should be started at the start of the next minute.

4. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

Run the P4 Code Review virtual host on a custom port

Important:

Back up your P4 Code Review virtual host configuration before customizing your P4 Code Review installation, see ["Back up your P4 Code Review virtual host first" on page 215](#).

If you cannot run Swarm on port 80 (or port 443 for HTTPS), perhaps because you do not have root access, it is possible to run P4 Code Review on a custom port.

Installing P4 Code Review to use a custom port requires modification of the previous installation steps covered in this chapter: The [Apache configuration](#) is slightly different, requiring [modification of P4 Code Review's virtual host definition](#).

The following section covers the specifics of the custom port configuration.

Note:

In addition to the following instructions, you may also need to apply the `external_url` item described in the ["Environment" on page 602](#) section if your P4 Code Review is behind a proxy, or you have multiple P4 Code Review instances connected to P4 Server.

Important:

If you used the [P4 Code Review packages](#) to install P4 Code Review, you can adjust P4 Code Review's configuration using the package configuration script `/opt/perforce/swarm/sbin/configure-swarm.sh`. `configure-swarm.sh` does not read any existing P4 Code Review configuration; you must provide all of the configuration details each time you execute `configure-swarm.sh`:

```
$ sudo /opt/perforce/swarm/sbin/configure-swarm.sh -n -p myp4host:1666 -u
swarm -w password -e mx.example.com -H myhost -P 8080
```

In the example above, the `-P` option is used to specify the custom port that P4 Code Review should use.

If you use `configure-swarm.sh` to adjust P4 Code Review's configuration, follow the additional steps that it describes. Once those steps are complete, do not perform any of the steps described below.

Apache configuration

1. Edit the virtual host configuration.

Note:

Depending on the method used to install P4 Code Review, the filename for virtual host configuration you need to edit is:

- For [P4 Code Review package](#) installations, edit `performce-swarm-site.conf`.
- For manual installations following P4 Code Review's [recommended Apache configuration](#), edit `swarm`.
- For other installations, you may have to edit `httpd.conf` or nearby files.

- a. Add the following line *outside* of the `<VirtualHost>` block:

```
Listen 8080
```

- b. Edit the `<VirtualHost *:80>` line to read:

```
<VirtualHost *:8080>
```

For both lines, replace `8080` with the custom port you wish to use.

Important:

If you choose a port that is already in use, Apache refuses to start.

2. Restart your web server.

```
$ sudo apachectl restart
```

3. Adjust your firewall configuration to allow connections to the custom port.

- For RHEL:

```
$ sudo firewall-cmd --zone=public --add-port=8080/tcp --permanent
$ sudo systemctl reload firewalld
```

Replace `8080` with the custom port you wish to use.

- For other distributions, consult with your network administrator or operating system documentation to determine how to adjust your firewall configuration.

Cron configuration

P4 Code Review's [recurring task](#) configuration must be updated to reflect the custom port that you have configured in [Apache's](#) configuration. This is configured in the `swarm-cron-hosts.conf` file.

1. Edit `/opt/perforce/etc/swarm-cron-hosts.conf`.
2. Replace:

```
https://swarm-host:80
```

with:

```
https://swarm-host:8080
```

Where `swarm-host` is the hostname of your P4 Code Review installation, and `8080` is the custom port you want to use.

3. Save the edited file.
New workers should be started at the start of the next minute.
4. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

Use your own Redis server

By default P4 Code Review uses its own Redis server on the P4 Code Review machine.

This section describes how to connect P4 Code Review to your own Redis server.

To use your own Redis server:

1. Add the redis block to the `SWARM_ROOT/data/config.php` file, it contains the password, namespace, host, and port details of your Redis server

```
<?php
// this block should be a peer of 'p4'
'redis' => array(
    'options' => array(
        'password' => null, // Defaults to null
        'namespace' => 'Swarm',
        'server' => array(
            'host' => 'MyRedisServerHostname', // Defaults to 'localhost' or enter your
Redis server hostname
            'port' => 'MyRedisServerPortNumber', // Defaults to '7379' or enter your Redis
server port
        ),
    ),
    'items_batch_size' => 100000,
    'check_integrity' => '03:00', // Defaults to '03:00' Use one of the following two
formats:
```

```
    // 1) The time of day that the integrity check starts each day. Set in
```

24 hour format with leading zeros and a : separator

// 2) The number of seconds between each integrity check. Set as a positive integer. Specify '0' to disable the integrity check.

'population_lock_timeout' => 300, // Timeout for initial cache population. Defaults to 300 seconds.

),

- password: set to the password of your Redis server, defaults to null
- namespace: the prefix used for key values in the Redis cache. Defaults to Swarm.

Note:

If you have multiple P4 Code Review instances running against a single Redis server, each P4 Code Review server must use a different Redis namespace. This enables the cache data for the individual P4 Code Review instances to be identified. The namespace is limited to ≤ 128 characters.

If one or more of your P4 Code Review instances is connected to multiple P4 Servers, the Redis namespace includes the server label and the character limit is reduced to ≤ 127 characters, see ["Multiple P4 Server instances" on page 638](#).

- host `MyRedisServerHostname`: Redis server host name
- port `MyRedisServerPortNumber`: Redis server port number
- items_batch_size: Maximum number of key/value pairs allowed in an mset call to Redis. Sets exceeding this will be batched according to this maximum for efficiency. Defaults to 100000.

Note:

The default value of 100000 was chosen to strike a balance between efficiency and project data complexity. This value should not normally need to be changed, contact support before making a change to this value.

- check_integrity: In some circumstances, such as when changes are made in the P4 Server when P4 Code Review is down or if errors occur during updates, the Redis cache can get out of sync with the P4 Server. P4 Code Review can run a regular integrity check to make sure that the Redis caches and P4 Server are in sync. If an integrity check finds an out of sync cache file, P4 Code Review automatically updates the data in that cache.

The check_integrity configurable specifies when the Redis cache integrity check is run. Set as a specific time of day (24 hour format with leading zeros) or a number of seconds (positive integer) between checks. Disable the integrity check with '0'. Defaults to '03:00'.

- population_lock_timeout: specifies the timeout, in seconds, for initial cache population. If you have a large P4 Code Review system, increase this time if the initial cache population times out. Defaults to 300 seconds.

2. Edit the Redis server configuration in the `/opt/perforce/etc/redis-server.conf` file:

```
bind MyRedisServerHostname
port MyRedisServerPortNumber
supervised auto
save ""
dir /MyRedis/CacheDatabase/Directory/Location
```

- `bind MyRedisServerHostname` - Redis server host name
- `port MyRedisServerPortNumber` - Redis server port number
- `supervised auto` - detects the use of `upstart` or `systemd` automatically to signal that the process is ready to use the supervisors
- `save ""` - background saves disabled, recommended.
- `dir /MyRedis/CacheDatabase/Directory/Location` - the directory the Redis cache database is stored in

Tip:

- The `redis-server.conf` file contains more detailed information about the Redis configuration for P4 Code Review.
- On P4 Code Review systems with a large number of users, groups, and projects, start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves, see the `redis-server.conf` file comments for detailed information.

P4 Code Review is now connected to your Redis server and your P4 Code Review installation is complete.

3. Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" on the facing page](#).

Hiding P4 Code Review storage from regular users

P4 Code Review information storage uses P4 Server's `keys` facility. By default, users with `list`-level access can search keys and potentially obtain information they would not otherwise have access to, and users with `review`-level access can write or modify keys potentially corrupting or destroying data.

We recommend that you set the `dm.keys.hide` configurable to 2 to require `admin`-level access for searching and modifying keys. Note that `dm.keys.hide` is available in P4 Server versions 2013.1 and newer.

When `dm.keys.hide` is set to 2, both the `p4 keys` and `p4 key` commands require `admin`-level access in the P4 Server. When `dm.keys.hide` is set to 1, only the `p4 keys` command requires `admin`-level access in the P4 Server. When `dm.keys.hide` is set to 1, or is not set, users who know (or can deduce) key names can read values (if they have `list`-level access) or write values (if they have `review`-level access) with the `p4 key` command.

To set `dm.keys.hide`:

```
p4 configure set dm.keys.hide=2
```

To confirm the current value of `dm.keys.hide`:

```
p4 configure show dm.keys.hide
```

To unset `dm.keys.hide`:

```
p4 configure unset dm.keys.hide
```

Next step

Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" below](#).

Handling Exclusive Locks

Tip:

If this setting is not enabled in P4 Server, P4 Code Review will report exceptions when working with exclusively opened files similar to Cannot unshelve review (x). One or more files are exclusively open, and that you must have the `filetype.bypasslock` configurable enabled.

P4 Code Review takes copies of files when it is creating reviews. Some of the files managed by P4 Server may be limited to 'exclusive open' by having the filetype modifier '+l' set. This file-level setting ensures only one user at a time can open the file for editing.

To allow P4 Code Review to work with these 'exclusive open' files, you must enable `filetype.bypasslock` in the P4 Server configuration.

To set `filetype.bypasslock`:

```
p4 configure set filetype.bypasslock=1
```

To confirm the current value of `filetype.bypasslock`:

```
p4 configure show filetype.bypasslock
```

To unset `filetype.bypasslock`:

```
p4 configure unset filetype.bypasslock
```

Next step

Validate that P4 Code Review is working correctly before using it in production, see ["Validate your P4 Code Review installation" below](#).

Validate your P4 Code Review installation

When P4 Code Review starts for the first time it verifies the Redis cache. During this verification you cannot log in to P4 Code Review. The time required to verify the Redis cache depends on the number of users, groups, and projects present in P4 Code Review.

Tip: To improve start-up times, persist the memory cache. To do this, disable background saves and enable append saves in the `redis-server.conf` file. For more information, see ["Redis server configuration file" on page 665](#).

With P4 Code Review is fully installed, check the P4 Code Review installation is working correctly. To check the installation, follow these steps:

1. Create a new changelist that:
 - a. Contains at least one modified file
 - b. Contains the #review [keyword](#) in the changelist description
2. Right click on the new changelist in P4V and click **Shelve Files...**

Important:

Do not select **Request New Review...** because this method uses the API and will not fully test the P4 Server extension.

This is also true if you are using P4 Code Review triggers instead of the P4 Server extension.

3. Check that a new review is created for the changelist.
 - If a review is created, the P4 Server extension is working. If you are using P4 Code Review triggers instead of the P4 Server extension and the review is created, the triggers are working.
 - If a review is not created, see ["Troubleshooting: Review not created"](#) below.

P4 Code Review is now installed and ready to use. For more information on the next steps, see ["Quickstart" on page 23](#).

Tip: P4 Code Review offers several optional integrations. For more information on what integrations are available, see ["Integrations" on page 535](#).

Troubleshooting: Azure Application Gateway issues with P4 Code Review and P4V

When running P4 Code Review behind an Azure Application Gateway (such as when using a load balancer with P4V), you may encounter authentication issues due to missing client IP information.

To resolve this, you need to configure the Azure Application Gateway to include an X-Forwarded-For header. Specifically, add a rewrite rule to insert or preserve the X-Forwarded-For header so that P4 Code Review can correctly identify the original client IP address. This can only be done by the Azure administrator managing your environment. It is not performed by P4 Remote Administration.

For more information, see [Configuring Azure Application Gateway for accessing Kibana](#).

Troubleshooting: Review not created

If you are using the P4 Server extension

If a new review is not created when you validate your installation, your P4 Server extension is probably not working correctly on the P4 Server.

Option 1: Check the P4 Server extension is working

To check that the P4 Server extension is working, run the following command on your P4 Server:

```
p4 extension --run swarm ping
```


Example success message:

OK

Example failure message:

BAD (Cannot reach https://swarm-example.com/)

For information about extension on your P4 Server, see the [P4 Server Administration Documentation](#).

Option 2: Check installed versions

To check your P4 Code Review version and your P4 Server extension version, run the following command on your P4 Server:

```
p4 extension --run swarm version
```

Example response:

```
Swarm Version: SWARM/2022.1/2226999 (2022/03/20)
Extension Version: 2022.1.20221215
```

If you are using P4 Code Review triggers

If a new review is not created when you validate your installation, your P4 Code Review triggers are probably not installed correctly on the P4 Server. For instructions on how to install the P4 Code Review triggers on your P4 Server, see ["Installing triggers" on page 187](#).

Troubleshooting: Review cannot be updated using triggers

It is unlikely reviews cannot be updated using triggers, but it can happen if your P4 Code Review server is configured for SSL.

Cause of this issue

The `swarm-trigger.pl` script uses the Perl `HTTP::Tiny` module for HTTP requests when the module is available. In certain environments, `HTTP::Tiny` may encounter issues related to SSL/TLS certificate validation, which can result in a HTTP 599 error.

The `HTTP::Tiny` module requires two other Perl modules to work properly with HTTPS. These modules are:

- `IO::Socket::SSL` (version 1.42 or higher)
- `Net::SSLeay` (version 1.49 or higher)

If these modules are missing or out of date, you may encounter a HTTP 599 error.

To verify if this is the cause of the issue, review the operating system log of your P4 Server for any error messages logged by `swarm-trigger.pl`. Depending on which operating system you are using, find the system logs in the following locations:

- For Windows operating systems, find the system logs using [Event Viewer](#).
- For Linux operating systems, find the system logs in the following locations:

- For Ubuntu or Debian operating systems, look in /var/log/syslog.
- For Red Hat operating systems, look in /var/log/messages.

The error message logged by `swarm-trigger.pl` starts with the following text: Error: (599/Internal Exception) (probably invalid SSL certificate)

Workaround for issue

If you see the 599 error, there are two available fixes:

- Fix 1: Edit the P4 Code Review trigger script file on the P4 Server so that it uses `Curl` instead of `HTTP::Tiny`.
- Fix 2: Install the required Perl modules. This fixes `HTTP::Tiny` so it can work as intended.

Fix 1: Modify the trigger script to use Curl:

There are two lines in the `swarm-trigger.pl` script that enable the use of `HTTP::Tiny`. Modify these two lines so the script reverts to using `Curl`:

1. In the `swarm-trigger.pl`, find the following line:
`if ($HAVE_TINY) {`
2. Add `!` to `$HAVE_TINY`. The new line should look like the following:
`if (!$HAVE_TINY) {`
3. Save the trigger script file.

Fix 2: Install the required Perl modules:

1. Run the following script to check if the required modules are installed or are using an outdated version

```
use HTTP::Tiny;

my $response = HTTP::Tiny->new->get('<instance_URL>');
print $response->{content} if length $response->{content};
die "Failed!\n" unless $response->{success};
```

Replace `<instance_URL>` with the URL to your P4 Code Review instance.

2. If these modules are missing or outdated, you will see the following output.

```
IO::Socket::SSL 1.42 must be installed for https support
Net::SSLeay 1.49 must be installed for https support
```

3. To install these two modules on your P4 Server, use the appropriate method for your OS and Perl installation, for example using Perl's CPAN module on the command line.

Upgrading P4 Code Review

The section describes how to upgrade a P4 Code Review package or tarball installation to a newer release.

Important:

- When upgrading to P4 Code Review 2022.3 or later, ensure that you upgrade P4 Visual Client (P4V) to 2021.3 or later.
- In the P4 Code Review 2024.3 release, Redis binary has been upgraded to version 7.2.4. As a result of this upgrade, the kernel and glibc requirements have automatically changed from version 2.6 to 3.10. If you need to run Redis under kernel version 2.6, you can do either of the following:
 - Compile Redis from sources on the target host meeting the target kernel.
 - Or use your own older Redis installation.

Tip:

If you are not already running P4 Code Review, none of these instructions apply to you. Instead, see the [Swarm installation instructions](#).

Choose your upgrade process:

- ["Upgrading a Ubuntu package installation" below](#).
- ["Upgrading a RHEL package installation" on page 243](#).
- ["Upgrading a Docker container" on page 258](#).
- ["Upgrading a tarball installation" on page 260](#).

Upgrading a Ubuntu package installation

The section describes how to upgrade a P4 Code Review Ubuntu package installation to a newer release.

Important:

- P4 Code Review runtime dependencies change between releases, you must check that your system satisfies the P4 Code Review runtime dependencies before starting the upgrade, see ["Runtime dependencies" on page 91](#).
- Review the PHP requirements before you upgrade P4 Code Review, see ["PHP" on page 94](#).
P4 Code Review no longer supports PHP 8.0 version.
- Review the P4 Server requirements before you upgrade P4 Code Review, see ["P4 Server requirements" on page 98](#).
- P4 Server 2020.1 and later, permissions have changed for viewing and [editing stream spec files](#) in P4 Code Review. To view and edit stream spec files in P4 Code

Review, the P4 Code Review user must have *admin* permissions for the entire depot //...

- If you are upgrading from P4 Code Review 2020.2 or earlier and have userids that contain the forward slash (/) character, add `AllowEncodedSlashes NoDecode` to the `VirtualHost` block of your `/etc/apache2/sites-enabled/perforce-swarm-site.conf` file. For more information about the `VirtualHost` block, see ["Apache configuration" on page 171](#).

Upgrade P4 Code Review

Tip:

From P4 Code Review 2021.1, the P4 Code Review package upgrade installs logrotate to manage your P4 Code Review log rotation. If the package upgrade finds an existing custom logrotate configuration file for P4 Code Review, the upgrade will notify you and give you details on how to disable the new logrotate configuration.

For information about the logrotate configuration, see ["Logrotate" on page 627](#).

The following process attempts to minimize downtime, but a short period of downtime for P4 Code Review users is unavoidable. There should be no downtime for your P4 Server. After a successful upgrade, all P4 Code Review users are logged out.

If you are using P4 Code Review in a production environment, we encourage you to test this upgrade process in a non-production environment first.

1. Run the following commands:

```
sudo apt-get update
sudo apt-get install helix-swarm helix-swarm-triggers helix-swarm-optional
```

2. P4 Code Review generally has several major updates each year, and may occasionally have a patch update between major updates. To determine whether a P4 Code Review update is available, run the following commands:

```
sudo apt-get update
sudo apt-get -s upgrade | grep swarm
```

Tip:

P4 Code Review uses a Redis server to manage its caches. This is installed and configured on the P4 Code Review machine during the upgrade. If you prefer to use your own Redis server, see ["Use your own Redis server" on page 224](#).

Configuring P4 Server event notification

P4 Code Review needs to know about a number of P4 Server events to operate correctly, this can be done by using P4 Server Extensions or P4 Server Triggers. P4 Code Review installs include the P4 Server extension file and trigger scripts required for P4 Code Review to get the events it needs from your P4 Server.

You must install the P4 Server extension or update your P4 Code Review triggers to complete the P4 Code Review upgrade.

Do one of the following so that P4 Code Review is notified about events on the P4 Server:

- Install and configure the P4 Server extension, see ["Installing the P4 Server extension" below](#)
- Update your P4 Code Review Triggers, see ["Updating your triggers" on page 237](#)

Installing the P4 Server extension

Important:

- If you are using the P4 Code Review extension to connect to the P4 Server, do not install P4 Code Review triggers.
- You must be a user with *super* user permissions to install and configure P4 Server Extensions.

Prerequisites

To install the P4 Server extension you need:

A compatible version of P4 Server for the extension:

- **Linux:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.
- **Windows:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.

You will also need:

- P4 Server Extensions installed and configured on your P4 Server.
- Your P4 Code Review user password
- A user with *super* permissions to install and configure P4 Server Extensions.

To install the P4 Server extension

There are two ways to install the P4 Server extension:

- Run the interactive post-installation configuration script. See [Install process using the interactive post-installation configuration script](#).

Note:

The P4 Code Review post-installation configuration script can be used in a few different ways. The steps below outline the most straightforward configuration using an interactive install, but you can review the options by running:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh -h
```

- Alternatively, use the `swarm-extctl.sh` script file located in `/opt/perforce/swarm/sbin` directory. The `swarm-extctl.sh` script file specifically handles the extension upgrade. See [Install process using the swarm-extctl.sh script](#).

Install process using the interactive post-installation configuration script

1. Run the interactive post-installation configuration script to install the P4 Server extension:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh
```

The configuration script displays a summary for your P4 Code Review instance:

```
-----
configure-swarm.sh: Thu Feb 17 14:20:49 PDT 2021: commencing configuration of
Swarm
Summary of arguments passed:
Interactive?    [yes]
Force?         [no]
P4PORT         [(not specified, will prompt)]
Swarm user     [(not specified, will prompt, will suggest swarm)]
Swarm password [(not specified, will prompt)]
Email host     [(not specified, will prompt)]
Use Extensions? [(not specified, will prompt)]
Swarm host     [(not specified, will prompt, will suggest myhost)]
Swarm port     [(default (80))]
Swarm base URL [(default (empty))]
Create Swarm user? [yes]
Super user     [(not specified, will prompt)]
Super password [(not specified, will prompt)]
```

2. The configuration script prompts you for your P4 Code Review configuration details.

Enter your P4 Code Review user password when prompted, all of the other details are pre-filled with your existing P4 Code Review configuration. Press the **[Enter]** key to accept each of them.

3. The script prompts you to use the P4 Server extension.

```
Do you want to use Swarm's Helix Core server extension?
Configuring Server extensions requires super user access to the Helix Server.
If you install the Swarm server extension, do not install the Swarm triggers.
Server extensions are supported for:
* Linux: Helix server 19.2 and later.
* Windows: Helix server 21.2 and later.
```

```
Use server extensions? (y/n) [n]
```

Type y to use P4 Server Extensions.

4. Sign in as a user with *super* permissions when prompted.
5. The script checks that P4 Server Extensions can be installed. The script will:

- Check your P4 Server supports P4 Server Extensions
- Check if P4 Server Extensions are installed and configured on your P4 Server. If they are, P4 Code Review does not need to do anything
- Check if P4 Code Review triggers are installed on the server

If any of these checks fail, P4 Code Review will not install the P4 Server extension and will report the issues on the configuration summary screen.

6. If P4 Code Review triggers are installed, you are prompted to remove them.

P4 Code Review triggers are installed on this server and must be removed before installing extensions. You can either have this script automatically remove them, or you can quit and do it manually.
Do you want to automatically remove the triggers? (y/n) [n]

- **Recommended:** Type `y` to get the script to automatically remove the P4 Code Review triggers.

The P4 Code Review triggers are not deleted immediately, the triggers to be removed are listed so you can review them before removing them.

- Type `n` to manually remove the P4 Code Review triggers from the trigger spec file.

7. The script requests confirmation that it is okay to remove the triggers:

Are you sure it is okay to remove these triggers? (y/n) [n]

- **Recommended:** Type `y` to automatically remove the triggers if the list of triggers looks correct. The script will:

- Save a timestamped copy of your old trigger spec to `/opt/perforce/swarm/triggers.saved.yyyymmdd-hhmmss`
- Save a timestamped copy of the new trigger spec to `/opt/perforce/swarm/triggers.noswarm.yyyymmdd-hhmmss`
- Remove the P4 Server triggers from the trigger spec and save it

If either of the trigger specs contain sensitive information, move them to a secure location.

- Type `n` if you see something wrong with the P4 Code Review triggers marked for removal or if you want to remove the triggers from the trigger spec manually. You are offered the option of automatically opening your trigger spec file in your default text editor or opening the file manually in a text editor. Save your changes and rerun the post-installation configuration script to complete the installation of the P4 Serverextension.

8. The script will now install and configure the P4 Server extension for you. When it has completed the configuration, the configuration summary screen is displayed, for example:

```
.....
- installing instance configuration
Extension config swarm saved
configure-swarm.sh: Thu Feb 17 14:31:36 PDT 2021: completed configuration of Helix
Swarm
```

```
.....
::
:: Swarm is now configured and available at:
::
:: http://myhost/
::
:: Ensure that you have configured the P4 Code Review hostname in your
:: network's DNS, or have added an IP address-to-hostname
:: mapping to your computer's hosts configuration so that you
:: can access P4 Code Review.
::
:: You may login as the Swarm user [swarm] using the password
:: you specified.
::
:: Server side extensions are installed and configured
:: on your P4D server.
::
:: Documentation for optional post-install configuration, such as
:: configuring Swarm to use HTTPS, operate in a sub-folder, or on a
:: custom port, is available:
::
:: https://www.perforce.com/perforce/doc.current/manuals/swarm/setup.post.html
::
.....
```

9. The P4 Code Review upgrade is now complete.

Install process using the `swarm-extctl.sh` script

1. Navigate to the the `swarm-extctl.sh` script file located in `/opt/perforce/swarm/sbin` directory.
2. Run the following commands as root:
 - a. Backup the current configuration. Ensure that the current configuration is correctly backed up before continuing.

```
sudo ./swarm-extctl.sh -p <masterserver|p:port> -u <username> save
```
 - b. Remove the existing extension from the master.

```
sudo ./swarm-extctl.sh -p <masterserver|p:port> -u <username> delete
```
 - c. Install the latest version of P4 Server extension located on the P4 Code Review server on the master. The following command will check for all saved configurations and apply them during install.

```
sudo ./swarm-extctl.sh -p <masterserver|p:port> -u <username> install
```
3. The P4 Code Review upgrade is now complete.

Next step

Now ["Validate your upgrade" on page 240](#).

Updating your triggers**Important:**

Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.

1. Copy the new P4 Code Review trigger script to your P4 Server machine. The trigger script is `SWARM_ROOT/p4-bin/scripts/swarm-trigger.pl`, and requires installation of Perl 5.08+ (use the latest available) on the P4 Server machine. If P4 Code Review is using SSL, then the triggers also require the `IO::Socket::SSL` Perl module.

Warning:

Do not overwrite any existing trigger script at this time. Give the script a new name, for example: `swarm-trigger-new.pl`.

2. Configure the P4 Code Review trigger script by creating, in the same directory on the P4 Server machine, `swarm-trigger.conf`. It should contain:

Note:

If you already have a `swarm-trigger.conf` file, no additional configuration is required.

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or "https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional) Do not use if the Workflow feature is enabled (default)
# For enforcing reviewed changes, optionally specify the normal Perforce user
# with admin privileges (to read keys); if not set, will use whatever Perforce
# user is set in environment.
ADMIN_USER=

# ADMIN_TICKET_FILE (optional) Do not use if the Workflow feature is enabled
(default)
# For enforcing reviewed changes, optionally specify the location of the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very long
```

```
# timeout; then, manually login as this user from the Perforce server machine to
# set the ticket.
ADMIN_TICKET_FILE=

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls whether
# the SSL certificate is validated or not. By default this is set to 1, which
# means any SSL certificates must be valid. If the web server is using a self
# signed certificate, then this must be set to 0.
# set the ticket.
VERIFY_SSL=1
```

Fill in the required `SWARM_HOST` and `SWARM_TOKEN` variables with the configuration from any previous P4 Code Review trigger script, typically `swarm-trigger.pl`.

Tip:

The `ADMIN_USER` and `ADMIN_TICKET` variables were used by the 'enforce triggers' in P4 Code Review 2019.1 and earlier. They can be removed unless you are explicitly [disabling workflow](#) and using the deprecated 'enforce triggers'.

Note:

P4 Code Review 2015.4 and earlier: P4 Code Review trigger script files were available as shell scripts in these earlier P4 Code Review versions, typically `swarm-trigger.sh`.

P4 Code Review must now use a Perl trigger script file, typically `swarm-trigger.pl`.

3. **On Linux:** ensure that the script is executable:

```
sudo chmod +x swarm-trigger-new.pl
```

4. Rename the new trigger script:

On Linux:

```
mv swarm-trigger-new.pl swarm-trigger.pl
```

On Windows:

```
ren swarm-trigger-new.pl swarm-trigger.pl
```

5. Update the triggers in your P4 Server.

Warning:

- The `swarm.shelvedel shelve-delete` trigger line was added to P4 Code Review in version 2018.1 and updated in version 2020.1.
 - **Upgrading from P4 Code Review 2017.4 and earlier:** add the `swarm.shelvedel shelve-delete` trigger line to the P4 Server trigger table if it is not already present, see ["Installing triggers" on page 187](#)

- **Upgrading from P4 Code Review 2018.x and 2019.x:** replace the existing `swarm.shelvedel` `shelve-delete` trigger line in the P4 Server trigger table with the one supplied in the P4 Code Review version you are upgrading to.

- **Workflow feature:**

The [Workflow feature](#) is enabled by default in P4 Code Review 2019.2 and later. The trigger lines required when workflow is enabled are different to those required when workflow is disabled:

- **Workflow feature [enabled](#) (default):**

- Comment out the `swarm.enforce.1`, `swarm.enforce.2`, `swarm.strict.1`, and `swarm.strict.2` trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)
- Add the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` trigger lines to the P4 Server trigger table if they are not already present, see ["Installing triggers" on page 187](#)

- **Workflow feature [disabled](#):**

Comment out the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)

- Run the P4 Code Review trigger script to capture (using **Ctrl+C** on Windows and Linux) the trigger lines that should be included in the Perforce trigger table:

On Linux:

```
./swarm-trigger.pl -o
```

On Windows:

```
path/to/perl swarm-trigger.pl -o
```

- As a Perforce user with *super* privileges, update the Perforce trigger table by running `p4 triggers` command and replacing any `swarm.*` lines with the previously captured trigger line output (using **Ctrl+V** on Windows and Linux).

Important:

If you previously customized the P4 Code Review trigger lines, perhaps to apply various ["Trigger options" on page 723](#), be sure to repeat those customizations within the updated trigger lines.

Next step

Now ["Validate your upgrade" on the next page](#).

Validate your upgrade

Tip:

When P4 Code Review starts it verifies the Redis cache, during this time you cannot log in to P4 Code Review. The time taken to verify the Redis cache depends on the number of users, groups, and projects P4 Code Review has. Start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves in the redis-server.conf file, see ["Redis server configuration file" on page 665](#).

Check that your upgraded P4 Code Review instance is working correctly by doing the following:

1. Create a new changelist that:
 - a. Contains at least one modified file
 - b. Contains the `#review keyword` in the changelist description
2. Right click on the new changelist in P4V and click **Shelve Files...**

Important:

Do not select **Request New Review...** because this method uses the API and will not fully test the P4 Server extension.

This is also true if you are using P4 Code Review triggers instead of the P4 Server extension.

3. Check that a new review is created for the changelist.
 - If a review is created, the P4 Server extension is working. If you are using P4 Code Review triggers instead of the P4 Server extension and the review is created, the triggers are working.
 - If a review is not created, see ["Troubleshooting: Review not created" on page 228](#).

P4 Code Review index upgrade

If you are upgrading from P4 Code Review 2017.2 or earlier you should run the index upgrade, this ensures that the review activity history is displayed in the correct order on the [Dashboard](#), and [Reviews list](#) pages.

Note:

If you are upgrading from P4 Code Review version 2017.3 or later, the index upgrade step is not required.

The index upgrade process can be configured to suit your P4 Code Review system specifications. See [Upgrade index](#) for details.

Run the upgrade as an *Admin* user by visiting the following URL:

`http://SWARM-HOST/upgrade`

Note:

After the P4 Code Review upgrade, on the first visit to some P4 Code Review pages, users might see a message to perform a browser hard refresh. This happens because we are updating the UI and content of some P4 Code Review pages, so the user's page cache is no longer valid and requires a hard refresh to load the upgraded page. For example, for Chrome on Windows/Linux **[CTRL]+[F5]**.

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers" on page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

4. Modify the P4 Server's `P4.Swarm.URL` property. For more information about P4 Server integration, see ["Client integration" on page 573](#).

If your Apache server is listening on both HTTPS and HTTP in `perforce-swarm-site.conf` file, you must set the `auto_register_urlconfigurable` in the `p4` block to `false` and correctly configure the `P4.Swarm.URL` property.

If your Apache server is listening only on HTTPS and if the `auto_register_urlconfigurable` in the `p4` block is set to `true` (default value), an Apache restart will correct the property.

To get all your current values for `P4.Swarm.URL` property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the `P4.Swarm.URL` property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.
- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.

- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

All done!

Upgrading a RHEL package installation

The section describes how to upgrade a P4 Code Review RHEL package installation to a newer release.

Important:

- P4 Code Review runtime dependencies change between releases, you must check that your system satisfies the P4 Code Review runtime dependencies before starting the upgrade, see ["Runtime dependencies" on page 91](#).
- Review the PHP requirements before you upgrade P4 Code Review, see ["PHP" on page 94](#).
P4 Code Review no longer supports PHP 8.0 version.
- Review the P4 Server requirements before you upgrade P4 Code Review, see ["P4 Server requirements" on page 98](#).
- P4 Server 2020.1 and later, permissions have changed for viewing and [editing stream spec files](#) in P4 Code Review. To view and edit stream spec files in P4 Code Review, the P4 Code Review user must have *admin* permissions for the entire depot //...

Note:

- If you are upgrading from P4 Code Review 2017.2 or earlier, run the P4 Code Review index upgrade after you have validated your upgrade. This is the last step of the upgrade and ensures that the review activity history is displayed in the correct order on the [Dashboard](#), and [Reviews list](#) pages.
- If you are upgrading from P4 Code Review 2020.2 or earlier and have userids that contain the forward slash (/) character, add AllowEncodedSlashes NoDecode to the VirtualHost block of your `/etc/apache2/sites-enabled/perforce-swarm-site.conf` file. For more information about the VirtualHost block, see ["Apache configuration" on page 171](#).

Upgrade P4 Code Review

Important:

The following notes are applicable for RHEL 8 and RHEL 9:

- As part of the PHP upgrade process your Apache 2.2 server will be stopped and disabled, if you are currently using the Apache 2.2 server for any other applications they will stop working. You will either need to upgrade these applications to work with PHP 8 and Apache 2.4 or move them to another server.

- **P4 Code Review 2020.2 and later:** these versions of P4 Code Review uses the Remi repository for RHEL 8 and RHEL 9. This provides PHP 8.x installed in the standard file system structure. This means that the old httpd24-httpd version of Apache is no longer needed, and the standard system version of Apache is being used again.
The SCL Apache site configuration file was installed at this location for P4 Code Review 2019.1 to 2020.1:
`/opt/rh/httpd24/root/etc/httpd/conf.d/perforce-swarm-site.conf`
If this exists when P4 Code Review is upgraded to 2020.2 and later, this file is copied to `/etc/httpd/conf.d/perforce-swarm-site.conf` if there is no file at the destination. It is also re-written to change references from `/var/log/httpd24` to `/var/log/httpd`
If a site configuration file for P4 Code Review already exists in `/etc/httpd`, the copy and re-write is not performed.
After upgrade, httpd24-httpd is disabled.
- To avoid seeing the Apache HTTP server Linux test page when you start the Apache server, comment out the content of the `welcome.conf` file located in the `/etc/httpd/conf.d/` directory.
- To avoid loading the Apache HTTP server example configuration instead of the P4 Code Review configuration when the Apache server starts, rename the `autoindex.conf` file located in the `/etc/httpd/conf.d/` directory to `z-autoindex.conf` or similar. This is required because Apache runs the first conf file it finds in the `/etc/httpd/conf.d/` directory (alphabetical order) and that must be the `perforce-swarm-site.conf` file.

Tip:

From P4 Code Review 2021.1, the P4 Code Review package upgrade installs logrotate to manage your P4 Code Review log rotation. If the package upgrade finds an existing custom logrotate configuration file for P4 Code Review, the upgrade will notify you and give you details on how to disable the new logrotate configuration.

For information about the logrotate configuration, see "[Logrotate](#)" on page 627.

The following process attempts to minimize downtime, but a short period of downtime for P4 Code Review users is unavoidable. There should be no downtime for your P4 Server. After a successful upgrade, all P4 Code Review users are logged out.

If you are using P4 Code Review in a production environment, we encourage you to test this upgrade process in a non-production environment first.

1. Install the main P4 Code Review package on the server to host P4 Code Review.

Follow the instructions for your OS distribution, see [RHEL 8](#) or [RHEL 9](#) below:

- **RHEL 8** (run these commands as root):

The full PHP and Apache upgrade process described below is only required the first time you upgrade to PHP 8.x. For future P4 Code Review upgrades just run the P4 Code Review upgrade steps, the ImageMagick steps if you want to install ImageMagick, and check that SELinux is working correctly.

The upgrade process will deploy `epel-release-latest-8.noarch.rpm` to give P4 Code Review access to the EPEL repository, deploy `remi-release-8.rpm` to give P4 Code Review access to PHP 8.x and Apache 2.4, upgrade P4 Code Review, stop and disable the Apache 2.2 server, copy and edit some P4 Code Review files so they work with Apache 2.4, and finally start and enable the Apache 2.4 server.

- a. Deploy the `epel-release-latest-8.noarch.rpm` repository configuration package to give P4 Code Review access to EPEL packages:

```
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

- b. Deploy the Remi repository configuration package to give P4 Code Review access to PHP 8.x (only required the first time you upgrade to PHP 8.x):

```
dnf install https://rpms.remirepo.net/enterprise/remi-release-8.rpm
```

Tip:

If you don't deploy the Remi repository, you will see dependency errors when you do the next steps.

- c. Install the `yum-utils` package to give access to the `yum-config-manager` command:

```
dnf install yum-utils
```

- d. Enable the optional channel for some dependencies:

```
subscription-manager repos --enable=rhel-8-server-optional-rpms
```

- e. Install the **Default/Single version** of PHP:

- i. Enable the module stream for PHP 8.3:

```
dnf module reset php
```

- ii. Install PHP 8.3:

```
dnf module install -y php:remi-8.3
```

- iii. Run an upgrade for PHP, this will also upgrade the P4 Code Review packages:

```
dnf update
```

- f. If you didn't run `dnf update` in the previous step, run the P4 Code Review package upgrade now:

```
yum install helix-swarm helix-swarm-triggers helix-swarm-optional
```

Important:

If you are upgrading from P4 Code Review 2019.3 to P4 Code Review 2021.1 or later, remove the P4 Code Review PHP 7.1 files:

```
yum remove $(yum list installed | grep php71 | awk '{print $1}')
```

- g. If you are upgrading from Apache 2.2, disable your Apache 2.2 HTTP server so that it does not automatically start:

```
systemctl disable httpd
```

- h. If you are upgrading from Apache 2.2, stop your Apache 2.2 HTTP server:

```
systemctl stop httpd
```

- i. If you have any special `php.ini` configuration options set, copy the `php.ini` file to the `/etc/php.d/php.ini/` directory, for example:

```
cp /etc/opt/rh/rh-php72/php.d/php.ini /etc/php.d/php.ini
```

- j. Copy the `perforce-swarm-site.conf` file to the `/etc/httpd/conf.d/` directory if it is not already in there, for example:

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/perforce-swarm-site.conf  
/etc/httpd/conf.d/perforce-swarm-site.conf
```

- k. Replace the `/log/httpd24filepath` with `/log/httpd/` in the `perforce-swarm-site.conf` file using the `sed` command:

```
sed -i "s#/log/httpd24/#/log/httpd/#g" /etc/httpd/conf.d/perforce-swarm-site.conf
```

- l. Enable your Apache 2.4 HTTP server so that it automatically starts:

```
systemctl enable httpd
```

- m. Start the Apache 2.4 HTTP server:

```
systemctl start httpd
```

- n. Enable your Apache 2.4 HTTP server so that it automatically starts:

```
systemctl enable httpd
```

- o. Start the Apache 2.4 HTTP server:

```
systemctl start httpd
```

- p. **Optional, ImageMagick:** when ImageMagick is enabled, P4 Code Review can display the following image formats: BMP, EPS, PSD, TGA, TIFF.

- i. Install ImageMagick:

```
yum install ImageMagick
```

- ii. Restart the web server so that ImageMagick is picked up.

```
systemctl restart httpd
```

- q. If you are upgrading from P4 Code Review 2020.1, restart your Redis server:

```
systemctl restart redis-server-swarm
```

- r. Check that SELinux is working correctly for your system. For instructions on configuring SELinux on RHEL, see "[SELinux configuration](#)" on page 137.

- **RHEL 9** (run these commands as root):

- a. Run an upgrade for PHP, this will also upgrade the P4 Code Review packages:

```
dnf update
```

- b. If you didn't run `dnf update` in the previous step, run the P4 Code Review package upgrade now:

```
yum install helix-swarm helix-swarm-triggers helix-swarm-optional
```

Important:

If you are upgrading from P4 Code Review 2019.3 to P4 Code Review 2021.1 or later, remove the P4 Code Review PHP 7.1 files:

```
yum remove $(yum list installed | grep php71 | awk '{print $1}')
```

- c. **Optional, ImageMagick:** when ImageMagick is enabled, P4 Code Review can display the following image formats: BMP, EPS, PSD, TGA, TIFF.
 - i. Install ImageMagick:


```
yum install ImageMagick
```
 - ii. Restart the web server so that ImageMagick is picked up.


```
systemctl restart httpd
```
- d. Check that SELinux is working correctly for your system. For instructions on configuring SELinux on RHEL, see ["SELinux configuration" on page 137](#).
2. P4 Code Review generally has several major updates each year, and may occasionally have a patch update between major updates. To determine whether a P4 Code Review update is available.

Run the following command as root:

```
yum list updates | grep swarm
```

Tip:

P4 Code Review uses a Redis server to manage its caches. This is installed and configured on the P4 Code Review machine during the upgrade. If you prefer to use your own Redis server, see ["Use your own Redis server" on page 224](#).

Configuring P4 Server event notification

P4 Code Review needs to know about a number of P4 Server events to operate correctly, this can be done by using P4 Server Extensions or P4 Server Triggers. P4 Code Review installs include the P4 Server extension file and trigger scripts required for P4 Code Review to get the events it needs from your P4 Server.

You must install the P4 Server extension or update your P4 Code Review triggers to complete the P4 Code Review upgrade.

Do one of the following so that P4 Code Review is notified about events on the P4 Server:

- Install and configure the P4 Server extension, see ["Installing the P4 Code Review extension for P4 Server" on the next page](#)
- Update your P4 Code Review triggers, see ["Updating your triggers" on page 252](#)

Installing the P4 Code Review extension for P4 Server

Important:

- If you are using the P4 Code Review extension to connect to the P4 Server, do not install P4 Code Review triggers.
- You must be a user with *super* user permissions to install and configure P4 Server Extensions.

Prerequisites

To install the P4 Server extension you need:

A compatible version of P4 Server for the extension:

- **Linux:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.
- **Windows:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.

You will also need:

- P4 Server Extensions installed and configured on your P4 Server.
- Your P4 Code Review user password
- A user with *super* permissions to install and configure P4 Server Extensions.

To install the P4 Server extension

There are two ways to install the P4 Server extension:

- Run the interactive post-installation configuration script. See [Install process using the interactive post-installation configuration script](#).

Note:

The P4 Code Review post-installation configuration script can be used in a few different ways. The steps below outline the most straightforward configuration using an interactive install, but you can review the options by running:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh -h
```

- Alternatively, use the `swarm-extctl.sh` script file located in `/opt/perforce/swarm/sbin` directory. The `swarm-extctl.sh` script file specifically handles the extension upgrade. See [Install process using the swarm-extctl.sh script](#).

Install process using the interactive post-installation configuration script

1. Run the interactive post-installation configuration script to install the P4 Server extension:

```
sudo /opt/perforce/swarm/sbin/configure-swarm.sh
```

The configuration script displays a summary for your P4 Code Review instance:

```

-----
configure-swarm.sh: Thu Feb 17 14:20:49 PDT 2021: commencing configuration of
Swarm
Summary of arguments passed:
Interactive?    [yes]
Force?         [no]
P4PORT         [(not specified, will prompt)]
Swarm user      [(not specified, will prompt, will suggest swarm)]
Swarm password  [(not specified, will prompt)]
Email host     [(not specified, will prompt)]
Use Extensions? [(not specified, will prompt)]
Swarm host      [(not specified, will prompt, will suggest myhost)]
Swarm port      [(default (80))]
Swarm base URL  [(default (empty))]
Create Swarm user? [yes]
Super user      [(not specified, will prompt)]
Super password  [(not specified, will prompt)]

```

2. The configuration script prompts you for your P4 Code Review configuration details.

Enter your P4 Code Review user password when prompted, all of the other details are pre-filled with your existing P4 Code Review configuration. Press the **[Enter]** key to accept each of them.

3. The script prompts you to use the P4 Server extension.

```

Do you want to use Swarm's Helix Core server extension?
Configuring Server extensions requires super user access to the Helix Server.
If you install the Swarm server extension, do not install the Swarm triggers.
Server extensions are supported for:
* Linux: Helix server 19.2 and later.
* Windows: Helix server 21.2 and later.

```

```

Use server extensions? (y/n) [n]

```

Type **y** to use P4 Server Extensions.

4. Sign in as a user with *super* permissions when prompted.
5. The script checks that P4 Server Extensions can be installed. The script will:
 - Check your P4 Server supports P4 Server Extensions
 - Check if P4 Server Extensions are installed and configured on your P4 Server. If they are, P4 Code Review does not need to do anything
 - Check if P4 Code Review triggers are installed on the server

If any of these checks fail, P4 Code Review will not install the P4 Server extension and will report the issues on the configuration summary screen.

6. If P4 Code Review triggers are installed, you are prompted to remove them.

P4 Code Review triggers are installed on this server and must be removed before installing extensions. You can either have this script automatically remove them, or you can quit and do it manually. Do you want to automatically remove the triggers? (y/n) [n]

- **Recommended:** Type `y` to get the script to automatically remove the P4 Code Review triggers.

The P4 Code Review triggers are not deleted immediately, the triggers to be removed are listed so you can review them before removing them.

- Type `n` to manually remove the P4 Code Review triggers from the trigger spec file.

7. The script requests confirmation that it is okay to remove the triggers:

Are you sure it is okay to remove these triggers? (y/n) [n]

- **Recommended:** Type `y` to automatically remove the triggers if the list of triggers looks correct. The script will:

- Save a timestamped copy of your old trigger spec to `/opt/perforce/swarm/triggers.saved.yyyymmdd-hhmmss`
- Save a timestamped copy of the new trigger spec to `/opt/perforce/swarm/triggers.noswarm.yyyymmdd-hhmmss`
- Remove the P4 Server triggers from the trigger spec and save it

If either of the trigger specs contain sensitive information, move them to a secure location.

- Type `n` if you see something wrong with the P4 Code Review triggers marked for removal or if you want to remove the triggers from the trigger spec manually. You are offered the option of automatically opening your trigger spec file in your default text editor or opening the file manually in a text editor. Save your changes and rerun the post-installation configuration script to complete the installation of the P4 Serverextension.

8. The script will now install and configure the P4 Server extension for you. When it has completed the configuration, the configuration summary screen is displayed, for example:

```

.....
- installing instance configuration
Extension config swarm saved
configure-swarm.sh: Thu Feb 17 14:31:36 PDT 2021: completed configuration of Helix
Swarm

```

```

.....
::
:: Swarm is now configured and available at:
::
:: http://myhost/
::
:: Ensure that you have configured the P4 Code Review hostname in your
:: network's DNS, or have added an IP address-to-hostname
:: mapping to your computer's hosts configuration so that you
:: can access P4 Code Review.
::
:: You may login as the Swarm user [swarm] using the password
:: you specified.
::
:: Server side extensions are installed and configured
:: on your P4D server.
::
:: Documentation for optional post-install configuration, such as
:: configuring Swarm to use HTTPS, operate in a sub-folder, or on a
:: custom port, is available:
::
:: https://www.perforce.com/perforce/doc.current/manuals/swarm/setup.post.html
::
.....

```

9. The P4 Code Review upgrade is now complete.

Install process using the `swarm-extctl.sh` script

1. Navigate to the the `swarm-extctl.sh` script file located in `/opt/perforce/swarm/sbin` directory.
2. Run the following commands as root:
 - a. Backup the current configuration. Ensure that the current configuration is correctly backed up before continuing.


```
sudo ./swarm-extctl.sh -p <masterserver|p:port> -u <username> save
```
 - b. Remove the existing extension from the master.


```
sudo ./swarm-extctl.sh -p <masterserver|p:port> -u <username> delete
```
 - c. Install the latest version of P4 Server extension located on the P4 Code Review server on the master. The following command will check for all saved configurations and apply them during install.


```
sudo ./swarm-extctl.sh -p <masterserver|p:port> -u <username> install
```
3. The P4 Code Review upgrade is now complete.

Next step

Now ["Validate your upgrade" on page 255](#).

Updating your triggers**Important:**

Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.

1. Copy the new P4 Code Review trigger script to your P4 Server machine. The trigger script is `SWARM_ROOT/p4-bin/scripts/swarm-trigger.pl`, and requires installation of Perl 5.08+ (use the latest available) on the P4 Server machine. If P4 Code Review is using SSL, then the triggers also require the `IO::Socket::SSL` Perl module.

Warning:

Do not overwrite any existing trigger script at this time. Give the script a new name, for example: `swarm-trigger-new.pl`.

2. Configure the P4 Code Review trigger script by creating, in the same directory on the P4 Server machine, `swarm-trigger.conf`. It should contain:

Note:

If you already have a `swarm-trigger.conf` file, no additional configuration is required.

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or "https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional) Do not use if the Workflow feature is enabled (default)
# For enforcing reviewed changes, optionally specify the normal Perforce user
# with admin privileges (to read keys); if not set, will use whatever Perforce
# user is set in environment.
ADMIN_USER=

# ADMIN_TICKET_FILE (optional) Do not use if the Workflow feature is enabled
(default)
# For enforcing reviewed changes, optionally specify the location of the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very long
```



```
# timeout; then, manually login as this user from the Perforce server machine to
# set the ticket.
ADMIN_TICKET_FILE=

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls whether
# the SSL certificate is validated or not. By default this is set to 1, which
# means any SSL certificates must be valid. If the web server is using a self
# signed certificate, then this must be set to 0.
# set the ticket.
VERIFY_SSL=1
```

Fill in the required `SWARM_HOST` and `SWARM_TOKEN` variables with the configuration from any previous P4 Code Review trigger script, typically `swarm-trigger.pl`.

Tip:

The `ADMIN_USER` and `ADMIN_TICKET` variables were used by the 'enforce triggers' in P4 Code Review 2019.1 and earlier. They can be removed unless you are explicitly [disabling workflow](#) and using the deprecated 'enforce triggers'.

Note:

P4 Code Review 2015.4 and earlier: P4 Code Review trigger script files were available as shell scripts in these earlier P4 Code Review versions, typically `swarm-trigger.sh`.

P4 Code Review must now use a Perl trigger script file, typically `swarm-trigger.pl`.

3. **On Linux:** ensure that the script is executable:

```
sudo chmod +x swarm-trigger-new.pl
```

4. Rename the new trigger script:

On Linux:

```
mv swarm-trigger-new.pl swarm-trigger.pl
```

On Windows:

```
ren swarm-trigger-new.pl swarm-trigger.pl
```

5. Update the triggers in your P4 Server.

Warning:

- The `swarm.shelvedel shelve-delete` trigger line was added to P4 Code Review in version 2018.1 and updated in version 2020.1.
 - **Upgrading from P4 Code Review 2017.4 and earlier:** add the `swarm.shelvedel shelve-delete` trigger line to the P4 Server trigger table if it is not already present, see ["Installing triggers" on page 187](#)

- **Upgrading from P4 Code Review 2018.x and 2019.x:** replace the existing `swarm.shelvedel shelve-delete` trigger line in the P4 Server trigger table with the one supplied in the P4 Code Review version you are upgrading to.
 - **Workflow feature:**
The [Workflow feature](#) is enabled by default in P4 Code Review 2019.2 and later. The trigger lines required when workflow is enabled are different to those required when workflow is disabled:
 - **Workflow feature [enabled](#) (default):**
 - Comment out the `swarm.enforce.1`, `swarm.enforce.2`, `swarm.strict.1`, and `swarm.strict.2` trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)
 - Add the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` trigger lines to the P4 Server trigger table if they are not already present, see ["Installing triggers" on page 187](#)
 - **Workflow feature [disabled](#):**
Comment out the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)
- a. Run the P4 Code Review trigger script to capture (using **Ctrl+C** on Windows and Linux) the trigger lines that should be included in the Perforce trigger table:

On Linux:

```
./swarm-trigger.pl -o
```

On Windows:

```
path/to/perl swarm-trigger.pl -o
```
- b. As a Perforce user with *super* privileges, update the Perforce trigger table by running `p4 triggers` command and replacing any `swarm.*` lines with the previously captured trigger line output (using **Ctrl+V** on Windows and Linux).

Important:

If you previously customized the P4 Code Review trigger lines, perhaps to apply various ["Trigger options" on page 723](#), be sure to repeat those customizations within the updated trigger lines.

Next step

Now ["Validate your upgrade" on the facing page](#).

Validate your upgrade

Tip:

When P4 Code Review starts it verifies the Redis cache, during this time you cannot log in to P4 Code Review. The time taken to verify the Redis cache depends on the number of users, groups, and projects P4 Code Review has. Start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves in the redis-server.conf file, see ["Redis server configuration file" on page 665](#).

Check that your upgraded P4 Code Review instance is working correctly by doing the following:

1. Create a new changelist that:
 - a. Contains at least one modified file
 - b. Contains the `#review keyword` in the changelist description
2. Right click on the new changelist in P4V and click **Shelve Files...**

Important:

Do not select **Request New Review...** because this method uses the API and will not fully test the P4 Server extension.

This is also true if you are using P4 Code Review triggers instead of the P4 Server extension.

3. Check that a new review is created for the changelist.
 - If a review is created, the P4 Server extension is working. If you are using P4 Code Review triggers instead of the P4 Server extension and the review is created, the triggers are working.
 - If a review is not created, see ["Troubleshooting: Review not created" on page 228](#).

P4 Code Review index upgrade

If you are upgrading from P4 Code Review 2017.2 or earlier you should run the index upgrade, this ensures that the review activity history is displayed in the correct order on the [Dashboard](#), and [Reviews list](#) pages.

Note:

If you are upgrading from P4 Code Review version 2017.3 or later, the index upgrade step is not required.

The index upgrade process can be configured to suit your P4 Code Review system specifications. See [Upgrade index](#) for details.

Run the upgrade as an *Admin* user by visiting the following URL:

`http://SWARM-HOST/upgrade`

Note:

After the P4 Code Review upgrade, on the first visit to some P4 Code Review pages, users might see a message to perform a browser hard refresh. This happens because we are updating the UI and content of some P4 Code Review pages, so the user's page cache is no longer valid and requires a hard refresh to load the upgraded page. For example, for Chrome on Windows/Linux **[CTRL]+[F5]**.

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers" on page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the "Reload Configuration button" on page 720.

4. Modify the P4 Server's `P4.Swarm.URL` property. For more information about P4 Server integration, see "Client integration" on page 573.

If your Apache server is listening on both HTTPS and HTTP in `perforce-swarm-site.conf` file, you must set the `auto_register_urlconfigurable` in the `p4` block to false and correctly configure the `P4.Swarm.URL` property.

If your Apache server is listening only on HTTPS and if the `auto_register_urlconfigurable` in the `p4` block is set to true (default value), an Apache restart will correct the property.

To get all your current values for `P4.Swarm.URL` property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the `P4.Swarm.URL` property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.
- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.

- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

All done!

Upgrading a Docker container

This section describes how to upgrade a Docker container that is running P4 Code Review to a newer release.

The following process attempts to minimize downtime, but a short period of downtime for P4 Code Review users is unavoidable. There should be no downtime for your P4 Server. After a successful upgrade, all P4 Code Review users are logged out.

If you are using P4 Code Review in a production environment, we encourage you to test this upgrade process in a non-production environment first.

To upgrade a Docker container, run the following commands in your Docker setup directory:

1. Pull down the latest version of the P4 Code Review image. In the below code example, you can change the *latest* tag to the current version of your P4 Code Review installation, for example, 2022.2.

```
docker pull perforce/helix-swarm:latest
```

2. View what is currently running in your Docker container using `docker ps` command.

```
docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED        STATUS
PORTS
NAMES
0b3baaf10387   perforce/helix-swarm "/bin/sh -c /opt/per..." 22 seconds ago Up 20
seconds  0.0.0.0:80->80/tcp,
:::80->80/tcp
helix-swarm
f2030d449679   redis                "docker-entrypoint.s..." 38 seconds ago Up 37
seconds  6379/tcp helix-redis
```

3. Stop and delete the running P4 Code Review image.

```
docker stop helix-swarm
docker rm helix-swarm
```

4. Start a new P4 Code Review image using a `docker run` command. For more information on how to run a P4 Code Review image on a Docker container, see ["Run P4 Code Review using a Docker container" on page 156](#).

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers" on page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

4. Modify the P4 Server's P4.Swarm.URL property. For more information about P4 Server integration, see ["Client integration" on page 573](#).

If your Apache server is listening on both HTTPS and HTTP in `performce-swarm-site.conf` file, you must set the `auto_register_urlconfigurable` in the `p4` block to `false` and correctly configure the P4.Swarm.URL property .

If your Apache server is listening only on HTTPS and if the `auto_register_urlconfigurable` in the `p4` block is set to `true` (default value), an Apache restart will correct the property.

To get all your current values for P4.Swarm.URL property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the P4.Swarm.URL property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.
- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.
- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

Upgrading a tarball installation

The section describes how to upgrade a P4 Code Review tarball installation to a newer release.

The following process attempts to minimize downtime, but a short period of downtime for P4 Code Review users is unavoidable. There should be no downtime for your P4 Server. After a successful upgrade, all P4 Code Review users are logged out.

If you are using P4 Code Review in a production environment, we encourage you to test this upgrade process in a non-production environment first.

Warning:

- P4 Code Review runtime dependencies change between releases, you must check that your system satisfies the P4 Code Review runtime dependencies before starting the upgrade, see ["Runtime dependencies" on page 91](#).
- P4 Code Review no longer supports PHP 8.0 version.
- P4PHP should be upgraded to the version included in the new P4 Code Review release.
 - If you have already configured PHP to use the P4 Code Review-provided P4 API for PHP ([as recommended](#)), this happens automatically.
 - If you have manually installed P4 API for PHP in some other fashion, configure P4 Code Review to use the version of P4 API for PHP included in the new P4 Code Review tarball before you perform any of the upgrade steps below. See ["PHP configuration" on page 174](#) for details.

P4 Code Review package and tarball installations: two versions of P4 API for PHP are supplied for PHP 8 version supported by P4 Code Review. They are located in the `p4-bin/bin.linux26x86_64` directory.

- `perforce-php8x.so` compatible with systems using SSL 1.0.2
- `perforce-php8x-ssl1.1.1.so` compatible with systems using SSL 1.1.1
- `perforce-php8x-ssl3.so` compatible with systems using SSL 3.0.0 (by default, Ubuntu 22.04 and 24.04 uses SSL 3.0.0)

Where x is the version of PHP 8.

P4 Code Review no longer supports PHP 8.0 version.

If the `perforce.ini` file is not pointing at the correct version of P4 API for PHP and you connect to an SSL enabled P4 Server:

- The P4 Code Review web-page will not load and you might see a Connection Reset error.
- There might be an undefined symbol: `SSLey` message in the Apache error log
- Review the PHP requirements before you upgrade P4 Code Review, see ["PHP" on page 94](#).
- Review the P4 Server requirements before you upgrade P4 Code Review, see ["P4 Server requirements" on page 98](#).
- P4 Server 2020.1 and later, permissions have changed for viewing and [editing stream spec files](#) in P4 Code Review. To view and edit stream spec files in P4 Code Review, the P4 Code Review user must have *admin* permissions for the entire depot //...

Note:

- If you are upgrading from P4 Code Review 2017.2 or earlier, run the P4 Code Review index upgrade after you have validated your upgrade. This is the last step of the upgrade and ensures that the review activity history is displayed in the correct order on the [Dashboard](#), and [Reviews list](#) pages.
- If you are upgrading from P4 Code Review 2020.2 or earlier and have userids that contain the forward slash (/) character, add AllowEncodedSlashes NoDecode to the VirtualHost block of your `/etc/apache2/sites-enabled/perforce-swarm-site.conf` file. For more information about the VirtualHost block, see "[Apache configuration](#)" on [page 171](#).

PHP and Apache 2.4 installation

RHEL does not have PHP 8 and Apache 2.4 by default so you must upgrade your system before you can upgrade P4 Code Review. This process is only required the first time you upgrade to PHP 8. If you have already upgraded to PHP 8 and Apache 2.4, see "[Configure Redis on the P4 Code Review machine](#)" on [page 265](#).

Carry out the following steps:

Important:

The following notes are applicable for RHEL 8 and RHEL 9:

- As part of the PHP upgrade process your Apache 2.2 server will be stopped and disabled, if you are currently using the Apache 2.2 server for any other applications they will stop working. You will either need to upgrade these applications to work with PHP 8 and Apache 2.4 or move them to another server.
- **RHEL 8:** Use the Remi repository configuration package (`remi-release-8.rpm`) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the `epel-release-latest-8.noarch.rpm` repository configuration package to give P4 Code Review access to EPEL packages.
- **RHEL 9:** Use the Remi repository configuration packages (`remi-release-8.rpm` and `remi-release-9.rpm`) to give P4 Code Review access to PHP 8 and to LibreOffice which is part of the optional package install. Use the `epel-release-latest-8.noarch.rpm` and `epel-release-latest-9.noarch.rpm` repository configuration package to give P4 Code Review access to EPEL packages.
- **P4 Code Review 2020.2 and later:** these versions of P4 Code Review uses the Remi repository for RHEL 8 and RHEL 9. This provides PHP 8.x installed in the standard file system structure. This means that the old `httpd24-httpd` version of Apache is no longer needed, and the standard system version of Apache is being used again.

The SCL Apache site configuration file was installed at this location for P4 Code Review 2019.1 to 2020.1:

```
/opt/rh/httpd24/root/etc/httpd/conf.d/perforce-swarm-site.conf
```

If this exists when P4 Code Review is upgraded to 2020.2 and later, this file is copied to `/etc/httpd/conf.d/perforce-swarm-site.conf` if there is no file at the destination. It is also re-written to change references from `/var/log/httpd24` to `/var/log/httpd`

If a site configuration file for P4 Code Review already exists in `/etc/httpd`, the copy and re-write is not performed.

After upgrade, `httpd24-httpd` is disabled.

- To avoid seeing the Apache HTTP server Linux test page when you start the Apache server, comment out the content of the `welcome.conf` file located in the `/etc/httpd/conf.d/` directory.
- To avoid loading the Apache HTTP server example configuration instead of the P4 Code Review configuration when the Apache server starts, rename the `autoindex.conf` file located in the `/etc/httpd/conf.d/` directory to `z-autoindex.conf` or similar. This is required because Apache runs the first `conf` file it finds in the `/etc/httpd/conf.d/` directory (alphabetical order) and that must be the `performer-swarm-site.conf` file.

This section describes how to install PHP 8 and Apache 2.4 on your system using a repository .

1. This process is only required the first time you upgrade to PHP 8. Install PHP 8 and migrate the P4 Code Review Apache configuration to use Apache 2.4.

Follow the instructions for your OS distribution, see [RHEL 8](#) or [RHEL 9](#) below:

- **RHEL 8** (run these commands as root):
 - a. Deploy the `epel-release-latest-8.noarch.rpm` repository configuration package to give P4 Code Review access to EPEL packages:


```
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```
 - b. Deploy the Remi repository configuration package to give P4 Code Review access to PHP 8 (only required the first time you upgrade to PHP 8):


```
dnf install https://rpms.remirepo.net/enterprise/remi-release-8.rpm
```
 - c. Install the `yum-utils` package to give access to the `yum-config-manager` command:


```
dnf install yum-utils
```
 - d. Enable the optional channel for some dependencies:


```
subscription-manager repos --enable=rhel-8-server-optional-rpms
```
 - e. Install the **Default/Single version** of PHP 8:
 - i. Enable the module stream for PHP:


```
dnf module reset php
```
 - ii. Install PHP.

If you want to install PHP 8.4 use the following command:

```
dnf module install php:remi-8.4
```

To install a specific version of PHP, replace the PHP 8.4 reference in the above command.

- iii. Run an upgrade for PHP:

```
dnf update
```

Important:

If you are upgrading from P4 Code Review 2019.3 to P4 Code Review 2021.1 or later, remove the P4 Code Review PHP 7.1 files:

```
yum remove $(yum list installed | grep php71 | awk '{print $1}')
```

- f. If you are upgrading from Apache 2.2, disable your Apache 2.2 HTTP server so that it does not automatically start:

```
systemctl disable httpd
```

- g. If you are upgrading from Apache 2.2, stop your Apache 2.2 HTTP server:

```
systemctl stop httpd
```

- h. If you are upgrading from PHP 7.2, copy the 30-perforce.ini file to /etc/php.d/ directory, this step also changes the ini filename to perforce.ini:

```
cp /etc/opt/rh/rh-php72/php.d/30-perforce.ini /etc/php.d/perforce.ini
```

- i. If you are upgrading from PHP 7.2, replace the PHP 7.2 references with PHP version number in the perforce.ini file using the sed command. For example, to upgrade from PHP 7.2 to PHP 7.4 use the following sed command:

```
sed -i "s/72/74/g" /etc/php.d/perforce.ini
```

- j. If you have any special php.ini configuration options set, copy the php.ini file to the /etc/php.d/ directory, for example:

```
cp /etc/opt/rh/rh-php72/php.d/php.ini /etc/php.d/php.ini
```

- k. Copy the perforce-swarm-site.conf file to the /etc/httpd/conf.d/ directory if it is not already in there, for example:

```
cp /opt/rh/httpd24/root/etc/httpd/conf.d/perforce-swarm-site.conf  
/etc/httpd/conf.d/perforce-swarm-site.conf
```

- l. Replace the /log/httpd24filepath with /log/httpd/ in the perforce-swarm-site.conf file using the sed command:

```
sed -i "s#/log/httpd24/#/log/httpd/#g" /etc/httpd/conf.d/perforce-swarm-site.conf
```

- m. Enable your Apache 2.4 HTTP server so that it automatically starts:

```
systemctl enable httpd
```

- n. Start the Apache 2.4 HTTP server:

```
systemctl start httpd
```

- o. Check that SELinux is working correctly for your system. For instructions on configuring SELinux on RHEL, see ["SELinux configuration" on page 137](#).

■ **RHEL 9** (run these commands as root):

- a. Run an upgrade for PHP, this will also upgrade the P4 Code Review packages:

```
dnf update
```

- b. If you didn't run `dnf update` in the previous step, run the P4 Code Review package upgrade now:

```
yum install helix-swarm helix-swarm-triggers helix-swarm-optional
```

Important:

If you are upgrading from P4 Code Review 2019.3 to P4 Code Review 2021.1 or later, remove the P4 Code Review PHP 7.1 files:

```
yum remove $(yum list installed | grep php71 | awk '{print $1}')
```

- c. **Optional, ImageMagick:** when ImageMagick is enabled, P4 Code Review can display the following image formats: BMP, EPS, PSD, TGA, TIFF.
 - i. Install ImageMagick:


```
yum install ImageMagick
```
 - ii. Restart the web server so that ImageMagick is picked up.


```
systemctl restart httpd
```
 - d. Check that SELinux is working correctly for your system. For instructions on configuring SELinux on RHEL, see ["SELinux configuration" on page 137](#).
2. PHP 8 and Apache 2.4 are now installed, configure the P4 Code Review Redis service on the P4 Code Review machine, see ["Configure Redis on the P4 Code Review machine" below](#).

Configure Redis on the P4 Code Review machine

Follow the instructions for the P4 Code Review version you are upgrading from:

- ["Upgrade from P4 Code Review 2019.1 and earlier" below](#)
- ["Upgrade from P4 Code Review 2019.2 and later" on page 269](#)

Upgrade from P4 Code Review 2019.1 and earlier

P4 Code Review requires Redis to manage its caches and by default P4 Code Review uses its own Redis server on the P4 Code Review machine. P4 Code Review caches data from the P4 Server to improve the performance of common searches in P4 Code Review and to reduce the load on the P4 Server. Redis is included in the P4 Code Review Tarball installation.

This section describes how to configure the P4 Code Review Redis service on the P4 Code Review machine. Do not change the default values of the following configuration files if you are using the P4 Code Review Redis server.

Tip:

If required, you can use your own Redis server instead of the P4 Code Review Redis server. For instructions on how to configure P4 Code Review to use your own Redis server, see ["Use your own Redis server" on page 224](#).

P4 Code Review has two Redis binaries in `SWARM_ROOT/p4-bin/bin.linux26x86_64/`:

- `redis-server-swarm`
- `redis-cli-swarm`

1. Configure the Redis cache database, enter the following details in the `redis-server.conf` file in `/opt/perforce/etc/`:

```
bind 127.0.0.1
port 7379
supervised auto
save ""
dir /opt/perforce/swarm/redis
```

Default values:

Tip:

- The default settings are shown below, the `redis-server.conf` file contains more detailed information about the Redis configuration for P4 Code Review.
- On P4 Code Review systems with a large number of users, groups, and projects, start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves, see the `redis-server.conf` file comments for detailed information.

- `bind 127.0.0.1` - Redis server IP address (loopback interface)
- `port 7379` - Redis server port number
- `supervised auto` - detects the use of `upstart` or `systemd` automatically to signal that the process is ready to use the supervisors
- `save ""` - background saves disabled, recommended.
- `dir /opt/perforce/swarm/redis` - the directory the Redis cache database is stored in.

Tip:

To fine-tune your Redis server settings, make your changes in the Laminas Redis adapter. For information about the Laminas Redis adapter, see the [Laminas Redis Adapter documentation](#).

2. The Redis cache database must be running before P4 Code Review starts so we recommend setting it up as a service so that it starts automatically at boot time.

The following example script will:

- run the Redis service as the Perforce user
- use the configuration file in `/opt/perforce/etc/redis-server.conf`
- assume the database server is installed in `/opt/perforce/swarm/p4-bin/bin.linux26x86_64/`

To configure Redis as a service, add a script with the following content in `/etc/systemd/system/redis-server-swarm.service`

```
[Unit]
Description=Redis Server for Swarm
After=network.target

[Service]
ExecStart=/opt/perforce/swarm/p4-bin/bin.linux26x86_64/redis-server-swarm
/opt/perforce/etc/redis-server.conf
ExecStop=/opt/perforce/swarm/p4-bin/bin.linux26x86_64/redis-cli-swarm shutdown
Restart=always
RestartSec=10
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=redis
User=perforce
Group=perforce

[Install]
WantedBy=multi-user.target
```

3. Create the `SWARM_ROOT/data/config.php` file if it does not already exist. The `redis` block of the `SWARM_ROOT/data/config.php` file contains the password, namespace, host and port details of the P4 Code Review Redis server:

```
<?php
// this block should be a peer of 'p4'
'redis' => array(
    'options' => array(
        'password' => null, // Defaults to null
        'namespace' => 'Swarm',
        'server' => array(
            'host' => 'localhost', // Defaults to 'localhost' or enter your Redis server
hostname
            'port' => '7379', // Defaults to '7379' or enter your Redis server port
        ),
    ),
    'items_batch_size' => 100000,
    'check_integrity' => '03:00', // Defaults to '03:00' Use one of the following two
formats:
        // 1) The time of day that the integrity check starts each day. Set in
24 hour format with leading zeros and a : separator
        // 2) The number of seconds between each integrity check. Set as a
positive integer. Specify '0' to disable the integrity check.
    'population_lock_timeout' => 300, // Timeout for initial cache population. Defaults to
300 seconds.
),
```

Configurables:

- `password`: Redis server password. Defaults to null and should be left at default if using the P4 Code Review Redis server.
- `namespace`: the prefix used for key values in the Redis cache. Defaults to `Swarm` and should be left at default if using the P4 Code Review Redis server.

Note:

If you have multiple P4 Code Review instances running against a single Redis server, each P4 Code Review server must use a different Redis namespace. This enables the cache data for the individual P4 Code Review instances to be identified. The namespace is limited to ≤ 128 characters.

If one or more of your P4 Code Review instances is connected to multiple P4 Servers, the Redis namespace includes the server label and the character limit is reduced to ≤ 127 characters, see ["Multiple P4 Server instances" on page 638](#).

- `host`: Redis server hostname. Defaults to `localhost` and should be left at default if using the P4 Code Review Redis server.
- `port`: Redis server port number. Defaults to `7379`. P4 Code Review uses port `7379` as its default to avoid clashing with other Redis servers that might be on your network. It should be left at default if using the P4 Code Review Redis server. The default port for a non-P4 Code Review Redis server is `6379`.
- `items_batch_size`: Maximum number of key/value pairs allowed in an `mset` call to Redis. Sets exceeding this will be batched according to this maximum for efficiency. Defaults to `100000`.

Note:

The default value of `100000` was chosen to strike a balance between efficiency and project data complexity. This value should not normally need to be changed, contact support before making a change to this value.

- `check_integrity`: In some circumstances, such as when changes are made in the P4 Server when P4 Code Review is down or if errors occur during updates, the Redis cache can get out of sync with the P4 Server. P4 Code Review can run a regular integrity check to make sure that the Redis caches and P4 Server are in sync. If an integrity check finds an out of sync cache file, P4 Code Review automatically updates the data in that cache.

The `check_integrity` configurable specifies when the Redis cache integrity check is run. Set as a specific time of day (24 hour format with leading zeros) or a number of seconds (positive integer) between checks. Disable the integrity check with `'0'`. Defaults to `'03:00'`.

- `population_lock_timeout`: specifies the timeout, in seconds, for initial cache population. If you have a large P4 Code Review system, increase this time if the initial cache population times out. Defaults to `300` seconds.

Now that the P4 Code Review Redis service is configured for the P4 Code Review machine, start your P4 Code Review upgrade, see ["Upgrade P4 Code Review" on the facing page](#).

Upgrade from P4 Code Review 2019.2 and later

P4 Code Review requires Redis to manage its caches and by default P4 Code Review uses its own Redis server on the P4 Code Review machine. P4 Code Review caches data from the P4 Server to improve the performance of common searches in P4 Code Review and to reduce the load on the P4 Server. Redis is included in the P4 Code Review Tarball installation.

Note:

If you are using your own Redis server instead of the P4 Code Review Redis server, upgrade your Redis server to the version supplied with your P4 Code Review upgrade.

P4 Code Review has two Redis binaries in **SWARM_ROOT**/p4-bin/bin.linux26x86_64/:

- `redis-server-swarm`
- `redis-cli-swarm`

If the Redis binaries supplied with your new version of P4 Code Review have been updated, your Redis server is automatically upgraded to the new Redis server version. Your original Redis configuration files will not be changed. Do not change the default values of the Redis configuration files if you are using the P4 Code Review Redis server.

Restart the Redis server:

If the Redis binaries have been updated and you are [running Redis as a service](#) (recommended), you must restart the Redis server to use the new Redis server version.

1. Run the following command to restart the Redis server:

```
systemctl restart redis-server-swarm
```

2. You are now running the updated Redis server version.

Tip:

On P4 Code Review systems with a large number of users, groups, and projects, start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves, see the `redis-server.conf` file comments for detailed information.

3. Now that the P4 Code Review Redis service is configured for the P4 Code Review machine, start your P4 Code Review upgrade, see ["Upgrade P4 Code Review" below](#).

Upgrade P4 Code Review

The steps in this section describe how to upgrade P4 Code Review using the provided archive file. **SWARM_ROOT** refers to the current P4 Code Review installation.

1. Download the new TAR file from [Download Helix Swarm](#).
2. Expand the new `swarm.tgz`:

```
tar -xzf swarm.tgz
```

The contents of `swarm.tgz` are expanded into a top-level folder named `swarm-version`, where `version` corresponds to the version downloaded. This directory is identified as `SWARM_NEW` below.

3. Move `SWARM_NEW` to be a peer of `SWARM_ROOT`:

```
mv SWARM_NEW SWARM_ROOT/..
```
4. Copy the `SWARM_ROOT/data/config.php` file from `SWARM_ROOT` to `SWARM_NEW`:

```
cp -p SWARM_ROOT/data/config.php SWARM_NEW/data/
```
5. Create the queue token directory:

```
mkdir SWARM_NEW/data/queue
```
6. Copy the existing trigger token(s):

```
sudo cp -pR SWARM_ROOT/data/queue/tokens SWARM_NEW/data/queue/
```
7. Configure the ImageMagick `policy.xml` file to allow P4 Code Review to create thumbnails if it is not already configured:
 - a. Open the ImageMagick `policy.xml` file for editing:

```
/etc/ImageMagick-6/policy.xml
```
 - b. Disable lines with the pattern below for the following `<filetype>s`: PS, PS2, EPS, PDF, or XPS:

```
<policy domain="coder" rights="none" pattern="<filetype>" />
```
 - c. Save the `policy.xml` file.
8. Assign correct ownership to the new P4 Code Review's data directory:

```
sudo chown -R www-data SWARM_NEW/data
```

Note:

The `www-data` user above is an example of what the web server user name might be, and can vary based on distribution or customization. For example, the user is typically `apache` for Red Hat and `www-data` for Debian/Ubuntu.

9. Update your triggers, see ["Update your triggers " below](#).

Update your triggers

Important:

Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.

1. Copy the new P4 Code Review trigger script to your P4 Server machine. The trigger script is `SWARM_ROOT/p4-bin/scripts/swarm-trigger.pl`, and requires installation of Perl 5.08+ (use the latest available) on the P4 Server machine. If P4 Code Review is using SSL, then the triggers also require the `IO::Socket::SSL` Perl module.

Warning:

Do not overwrite any existing trigger script at this time. Give the script a new name, for example: `swarm-trigger-new.pl`.

2. Configure the P4 Code Review trigger script by creating, in the same directory on the P4 Server machine, `swarm-trigger.conf`. It should contain:

Note:

If you already have a `swarm-trigger.conf` file, no additional configuration is required.

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or "https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional) Do not use if the Workflow feature is enabled (default)
# For enforcing reviewed changes, optionally specify the normal Perforce user
# with admin privileges (to read keys); if not set, will use whatever Perforce
# user is set in environment.
ADMIN_USER=

# ADMIN_TICKET_FILE (optional) Do not use if the Workflow feature is enabled
(default)
# For enforcing reviewed changes, optionally specify the location of the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very long
# timeout; then, manually login as this user from the Perforce server machine to
# set the ticket.
ADMIN_TICKET_FILE=

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls whether
# the SSL certificate is validated or not. By default this is set to 1, which
# means any SSL certificates must be valid. If the web server is using a self
# signed certificate, then this must be set to 0.
# set the ticket.
VERIFY_SSL=1
```

Fill in the required `SWARM_HOST` and `SWARM_TOKEN` variables with the configuration from any previous P4 Code Review trigger script, typically `swarm-trigger.pl`.

Tip:

The ADMIN_USER and ADMIN_TICKET variables were used by the 'enforce triggers' in P4 Code Review 2019.1 and earlier. They can be removed unless you are explicitly [disabling workflow](#) and using the deprecated 'enforce triggers'.

Note:

P4 Code Review 2015.4 and earlier: P4 Code Review trigger script files were available as shell scripts in these earlier P4 Code Review versions, typically swarm-trigger.sh.

P4 Code Review must now use a Perl trigger script file, typically swarm-trigger.pl.

3. **On Linux:** ensure that the script is executable:

```
sudo chmod +x swarm-trigger-new.pl
```

4. Rename the new trigger script:

On Linux:

```
mv swarm-trigger-new.pl swarm-trigger.pl
```

On Windows:

```
ren swarm-trigger-new.pl swarm-trigger.pl
```

5. Update the triggers in your P4 Server.

Warning:

- The swarm.shelvedel shelfe-delete trigger line was added to P4 Code Review in version 2018.1 and updated in version 2020.1.
 - **Upgrading from P4 Code Review 2017.4 and earlier:** add the swarm.shelvedel shelfe-delete trigger line to the P4 Server trigger table if it is not already present, see ["Installing triggers" on page 187](#)
 - **Upgrading from P4 Code Review 2018.x and 2019.x:** replace the existing swarm.shelvedel shelfe-delete trigger line in the P4 Server trigger table with the one supplied in the P4 Code Review version you are upgrading to.
- **Workflow feature:**

The [Workflow feature](#) is enabled by default in P4 Code Review 2019.2 and later. The trigger lines required when workflow is enabled are different to those required when workflow is disabled:

 - **Workflow feature [enabled](#) (default):**
 - Comment out the swarm.enforce.1, swarm.enforce.2, swarm.strict.1, and swarm.strict.2 trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)
 - Add the swarm.enforce change-submit, swarm.strict change-content, and swarm.shelvesub shelfe-submit

trigger lines to the P4 Server trigger table if they are not already present, see ["Installing triggers" on page 187](#)

■ **Workflow feature disabled:**

Comment out the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)

- a. Run the P4 Code Review trigger script to capture (using **Ctrl+C** on Windows and Linux) the trigger lines that should be included in the Perforce trigger table:

On Linux:

```
./swarm-trigger.pl -o
```

On Windows:

```
path/to/perl swarm-trigger.pl -o
```

- b. As a Perforce user with *super* privileges, update the Perforce trigger table by running `p4 triggers` command and replacing any `swarm.*` lines with the previously captured trigger line output (using **Ctrl+V** on Windows and Linux).

Important:

If you previously customized the P4 Code Review trigger lines, perhaps to apply various ["Trigger options" on page 723](#), be sure to repeat those customizations within the updated trigger lines.

Replace your old P4 Code Review instance with your new P4 Code Review instance

Important:

Downtime occurs in this step.

Replace your old P4 Code Review instance with your new P4 Code Review instance. Run the following command:

```
sudo apache2ctl stop; mv SWARM_ROOT SWARM.old; mv SWARM_NEW SWARM_ROOT; sudo apache2ctl start
```

Note:

If you see the following error message when you start P4 Code Review, P4 Code Review is using the wrong version of P4 API for PHP. The latest version of P4 API for PHP is included in the P4 Code Review tarball but you must configure P4 Code Review to use that version of P4 API for PHP. For instructions about how to configure P4 Code Review to use the new version of P4 API for PHP, see ["PHP configuration" on page 174](#).

Swarm has detected a configuration error

Problem detected:

- The Perforce PHP extension (P4PHP) requires upgrading. Found 2016.2, only 2018.2 or later is supported.

The php.ini file loaded is /etc/php5/apache2/php.ini.

Other scanned php.ini files (in /etc/php5/apache2/conf.d) include:

- /etc/php5/apache2/conf.d/imagick.ini
- /etc/php5/apache2/conf.d/pdo.ini
- /etc/php5/apache2/conf.d/perforce.ini

Please investigate the below PHP error:

The Perforce PHP extension (P4PHP) requires upgrading.
Found 2016.2, only 2018.2 or later is supported.

/var/www/deployments/1703508/swarm/public/index.php on line 104

For more information, please see the [Setting Up](#) documentation; in particular:

- [Initial Installation](#)
- [Runtime dependencies](#)
- [PHP configuration](#)
- [Swarm configuration](#)

Please ensure you restart your web server after making any PHP changes.

Reload the P4 Code Review config cache

As an *admin* user, delete and reload the P4 Code Review configuration cache from the **Cache Info** tab of the **System Information** page. Navigate to the **User id** dropdown menu, select **System Information**, select the **Cache Info** tab, and select **Reload Configuration**. For more information, see "Cache Info" on page 719.

Validate your upgrade

Tip:

When P4 Code Review starts it verifies the Redis cache, during this time you cannot log in to P4 Code Review. The time taken to verify the Redis cache depends on the number of users, groups, and projects P4 Code Review has. Start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves in the redis-server.conf file, see "[Redis server configuration file](#)" on page 665.

Check that your new P4 Code Review instance is working correctly by doing the following:

1. Create a new changelist that:
 - a. Contains at least one modified file
 - b. Contains the #review [keyword](#) in the changelist description
2. Right click on the new changelist in P4V and click **Shelve Files...**

Important:

Do not select **Request New Review...** because this method uses the API and will not fully test the P4 Server extension.

This is also true if you are using P4 Code Review triggers instead of the P4 Server extension.

3. Check that a new review is created for the changelist.
 - If a review is created, the P4 Server extension is working. If you are using P4 Code Review triggers instead of the P4 Server extension and the review is created, the triggers are working.
 - If a review is not created, see "[Troubleshooting: Review not created](#)" on page 228.

P4 Code Review index upgrade

If you are upgrading from P4 Code Review 2017.2 or earlier you should run the index upgrade, this ensures that the review activity history is displayed in the correct order on the [Dashboard](#), and [Reviews list](#) pages.

Note:

If you are upgrading from P4 Code Review version 2017.3 or later, the index upgrade step is not required.

The index upgrade process can be configured to suit your P4 Code Review system specifications. See [Upgrade index](#) for details.

Run the upgrade as an *Admin* user by visiting the following URL:

`http://SWARM-HOST/upgrade`

Note:

After the P4 Code Review upgrade, on the first visit to some P4 Code Review pages, users might see a message to perform a browser hard refresh. This happens because we are updating the UI and content of some P4 Code Review pages, so the user's page cache is no longer valid and requires a hard refresh to load the upgraded page. For example, for Chrome on Windows/Linux **[CTRL]+[F5]**.

Secure your P4 Code Review installation

To make your P4 Code Review installation more secure apply the following recommendations for HTTP and P4 Code Review implementation through security groups.

HTTP

Here is a list of best practices to use when port 80 is exposed for HTTP traffic:

- **Redirect to HTTPS:** If Port 80 needs to be open to support legacy systems or specific use cases, ensure that all HTTP traffic is redirected to HTTPS to encrypt data in transit.
- **Use HSTS (HTTP Strict Transport Security) headers:** Implement HSTS headers to force browsers only to use secure HTTPS connections when interacting with your server.
- **Close port 80:** If there is no requirement to use HTTP, Port 80 must be closed entirely to prevent any unencrypted data transmission.
- **Implement SSL/TLS (secure sockets layer and transport layer security) certificates:** Ensure that your server is configured with a valid SSL/TLS certificate to enable secure HTTPS connections.
- **Firewall configuration:** Configure firewalls to block or filter access to Port 80, particularly from untrusted networks.
- **Continuous monitoring and auditing:** Regularly monitor network traffic and audit server configurations to ensure that unnecessary ports are not exposed and that data is transmitted securely.

When you implement HTTPS, you must make the following changes:

Modify your cron job for the P4 Code Review workers.

Edit the cron configuration file to point to your HTTPS URL, for example, `https://HOSTNAME/`. For more information about how to edit the cron configuration file, see ["Set up a recurring task to spawn workers" on page 208](#).

To verify if the cron configuration file points to your HTTPS URL, run the following curl statement:

```
curl https://myswarm.host/queue/worker
```

- 1.
2. Modify the P4 Code Review Extension or Trigger configuration.

If you are using the P4 Code Review extension run the following command and change ExtConfig's P4 Code Review URL to be your new HTTPS URL:

```
p4 extension --configure Perforce:helix-swarm
```

If you are using triggers, edit `swarm-trigger.pl` configuration file and set your `SWARM_HOST` to be `https`.

3. Edit the `external_url` in the `SWARM_ROOT/data/config.php` file's environment block to point to your HTTPS URL. This URL is used in emails, Jira links, and P4 Code Review test's pass-and-fail outgoing URL parameters.

Tip: If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

4. Modify the P4 Server's P4.Swarm.URL property. For more information about P4 Server integration, see ["Client integration" on page 573](#).

If your Apache server is listening on both HTTPS and HTTP in `performce-swarm-site.conf` file, you must set the `auto_register_urlconfigurable` in the `p4` block to `false` and correctly configure the P4.Swarm.URL property .

If your Apache server is listening only on HTTPS and if the `auto_register_urlconfigurable` in the `p4` block is set to `true` (default value), an Apache restart will correct the property.

To get all your current values for P4.Swarm.URL property, run:

```
p4 -Ztag property -A -l -n P4.Swarm.URL
```

Ensure that the P4.Swarm.URL property points to your HTTPS URL.

5. Modify the URL of all applications. Any other applications that reference the URL should be switched to using the HTTPS URL.

P4 Code Review implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the X-Forwarded-For header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.
- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.
- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

All done!

Moving your P4 Code Review instance

This section describes how to move your P4 Code Review instance to another server. This can be useful if you want to move P4 Code Review to a different server to improve performance or if you want to run P4 Code Review on a newer operating system. P4 Code Review data is held on the P4 Server and not in P4 Code Review so there is no need to migrate any data to your new P4 Code Review instance.

If you are installing a newer version of P4 Code Review than your original P4 Code Review instance, you might need to configure options if they are not already present in your original version of P4 Code Review.

For example, P4 Code Review workflow, updated trigger scripts, the introduction of Extensions, the introduction of a Redis cache, and changes to custom modules. For information on important changes to P4 Code Review, see the ["What's new in P4 Code Review" on page 21](#) section of the P4 Code Review ["What's new in P4 Code Review" on page 21](#) page.

Warning:

There is a risk of P4 Code Review missing data updates if P4 Code Review has been shut down while P4 Server is running and users are executing P4 Server commands.

When P4 Code Review is offline, workflow triggers will fail which could cause P4 Server to stop working.

Summary

Moving your P4 Code Review installation to a new server results in downtime, but the following process minimizes it. The process for moving your P4 Code Review installation to another server is summarized below:

1. Install P4 Code Review on a new server.
2. Copy your original P4 Code Review config.php file and authentication token to your new P4 Code Review instance.
3. **Only required if you are moving to a newer version of P4 Code Review:** Configure your new P4 Code Review instance to receive event notifications from the P4 Server.
4. Shutdown your original P4 Code Review instance and start your new P4 Code Review instance .

Caution: Downtime occurs at this step.

5. Validate that your new P4 Code Review instance is working correctly.

For step by step instructions for moving your P4 Code Review instance, start by ["Installing P4 Code Review on your new server" below](#).

Installing P4 Code Review on your new server

Choose one of the following installation methods (we recommend the package installation method whenever possible):

- ["Install and configure P4 Code Review from a package \(recommended\)" on page 106](#). Do not follow the **Post-installation configuration** steps. Copy your original P4 Code Review config.php file to your new P4 Code Review instance, see ["Copying your existing P4 Code Review configuration" below](#).
- ["Install and configure P4 Code Review manually from a Tarball" on page 166](#), configure [Redis](#), configure [Apache](#), and configure [PHP](#). Copy your original P4 Code Review config.php file to your new P4 Code Review instance, see ["Copying your existing P4 Code Review configuration" below](#).

Copying your existing P4 Code Review configuration

1. Copy your original `SWARM_ROOT/data/config.php` file to your new P4 Code Review instance.
2. Create the queue token directory on your new P4 Code Review instance:
`mkdir SWARM_ROOT/data/queue`
3. Copy your original trigger token(s) from the `SWARM_ROOT/data/queue/tokens` to your new P4 Code Review instance.
4. Assign correct ownership to the new P4 Code Review instance data directory:

```
sudo chown -R www-data SWARM_ROOT/data
```

The `www-data` user above is an example of what the web server user name might be, and can vary based on distribution or customization. For example, the user is typically `apache` for Red Hat and `www-data` for Debian/Ubuntu.

5. Do one of the following:
 - **If moving to an instance with the same P4 Code Review version:** go to ["Replacing your original P4 Code Review instance with your new instance" on page 283](#).
 - **If moving to an instance with a newer version of P4 Code Review:** go to ["Configuring P4 Server event notification" below](#).

Configuring P4 Server event notification

Important:

The steps in this section are only required if you are moving your P4 Code Review instance to a newer version of P4 Code Review. If the new and original P4 Code Review instances are the same P4 Code Review version, go to ["Replacing your original P4 Code Review instance with your new instance" on page 283](#).

P4 Code Review needs to know about a number of P4 Server events to operate correctly, this can be done by using P4 Server Extensions or P4 Server Triggers.

You must configure your new P4 Code Review instance to receive event notifications from the P4 Server to complete the P4 Code Review move.

Do one of the following so that P4 Code Review is notified about events on the P4 Server:

- **If your old P4 Code Review instance is using P4 Server Extensions:** update the P4 Server extension configuration, see ["Updating your P4 Server extension" below](#)
- **If your old P4 Code Review instance is using P4 Server Triggers:** update your P4 Code Review Triggers, see ["Updating your triggers" below](#)

Updating your P4 Server extension

Important:

You must be a user with *super* user permissions to configure P4 Server Extensions.

If you are moving a P4 Code Review instance that is already using P4 Server Extensions, there is no need to reinstall the extension because it is already installed on P4 Server.

To update your P4 Serverextension:

1. Open the extension configuration file with:

```
p4 extension --configure Perforce::helix-swarm
```

The spec file opens in your text editor
2. Edit the configuration for your new P4 Code Review instance as required.
3. **Swarm-URL:** do one of the following:
 - **If the hostname of your P4 Code Review server has changed:** set the **Swarm-URL** to the URL of your new P4 Code Review server instance.
 - **If the P4 Code Review server hostname has not changed:** you do not need to change the Swarm-URL in the configuration file.
4. Save the configuration changes.
5. Continue with the next task, see ["Replacing your original P4 Code Review instance with your new instance" on page 283](#).

Updating your triggers

Important:

Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.

1. Copy the new P4 Code Review trigger script to your P4 Server machine. The trigger script is `SWARM_ROOT/p4-bin/scripts/swarm-trigger.pl`, and requires installation of Perl 5.08+ (use the latest available) on the P4 Server machine. If P4 Code Review is using SSL, then the triggers also require the `IO::Socket::SSL` Perl module.

Warning:

Do not overwrite any existing trigger script at this time. Give the script a new name, for example: `swarm-trigger-new.pl`.

2. Configure the P4 Code Review trigger script by creating, in the same directory on the P4 Server machine, `swarm-trigger.conf`. It should contain:

Note:

If you already have a `swarm-trigger.conf` file, no additional configuration is required.

```
# SWARM_HOST (required)
# Hostname of your Swarm instance, with leading "http://" or "https://".
SWARM_HOST="http://my-swarm-host"

# SWARM_TOKEN (required)
# The token used when talking to Swarm to offer some security. To obtain the
# value, log in to Swarm as a super user and select 'About Swarm' to see the
# token value.
SWARM_TOKEN="MY-UUID-STYLE-TOKEN"

# ADMIN_USER (optional) Do not use if the Workflow feature is enabled (default)
# For enforcing reviewed changes, optionally specify the normal Perforce user
# with admin privileges (to read keys); if not set, will use whatever Perforce
# user is set in environment.
ADMIN_USER=

# ADMIN_TICKET_FILE (optional) Do not use if the Workflow feature is enabled
(default)
# For enforcing reviewed changes, optionally specify the location of the
# p4tickets file if different from the default ($HOME/.p4tickets).
# Ensure this user is a member of a group with an 'unlimited' or very long
# timeout; then, manually login as this user from the Perforce server machine to
# set the ticket.
ADMIN_TICKET_FILE=

# VERIFY_SSL (optional)
# If HTTPS is being used on the Swarm web server, then this controls whether
# the SSL certificate is validated or not. By default this is set to 1, which
# means any SSL certificates must be valid. If the web server is using a self
# signed certificate, then this must be set to 0.
# set the ticket.
VERIFY_SSL=1
```

Fill in the required `SWARM_HOST` and `SWARM_TOKEN` variables with the configuration from any previous P4 Code Review trigger script, typically `swarm-trigger.pl`.

Tip:

The ADMIN_USER and ADMIN_TICKET variables were used by the 'enforce triggers' in P4 Code Review 2019.1 and earlier. They can be removed unless you are explicitly [disabling workflow](#) and using the deprecated 'enforce triggers'.

Note:

P4 Code Review 2015.4 and earlier: P4 Code Review trigger script files were available as shell scripts in these earlier P4 Code Review versions, typically swarm-trigger.sh.

P4 Code Review must now use a Perl trigger script file, typically swarm-trigger.pl.

3. **On Linux:** ensure that the script is executable:

```
sudo chmod +x swarm-trigger-new.pl
```

4. Rename the new trigger script:

On Linux:

```
mv swarm-trigger-new.pl swarm-trigger.pl
```

On Windows:

```
ren swarm-trigger-new.pl swarm-trigger.pl
```

5. Update the triggers in your P4 Server.

Warning:

- The swarm.shelvedel shelfe-delete trigger line was added to P4 Code Review in version 2018.1 and updated in version 2020.1.
 - **Upgrading from P4 Code Review 2017.4 and earlier:** add the swarm.shelvedel shelfe-delete trigger line to the P4 Server trigger table if it is not already present, see ["Installing triggers" on page 187](#)
 - **Upgrading from P4 Code Review 2018.x and 2019.x:** replace the existing swarm.shelvedel shelfe-delete trigger line in the P4 Server trigger table with the one supplied in the P4 Code Review version you are upgrading to.
- **Workflow feature:**

The [Workflow feature](#) is enabled by default in P4 Code Review 2019.2 and later. The trigger lines required when workflow is enabled are different to those required when workflow is disabled:

 - **Workflow feature [enabled](#) (default):**
 - Comment out the swarm.enforce.1, swarm.enforce.2, swarm.strict.1, and swarm.strict.2 trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)
 - Add the swarm.enforce change-submit, swarm.strict change-content, and swarm.shelvesub shelfe-submit

trigger lines to the P4 Server trigger table if they are not already present, see ["Installing triggers" on page 187](#)

■ **Workflow feature disabled:**

Comment out the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` trigger lines in the P4 Server trigger table if they are present, see ["Installing triggers" on page 187](#)

- a. Run the P4 Code Review trigger script to capture (using **Ctrl+C** on Windows and Linux) the trigger lines that should be included in the Perforce trigger table:

On Linux:

```
./swarm-trigger.pl -o
```

On Windows:

```
path/to/perl swarm-trigger.pl -o
```

- b. As a Perforce user with *super* privileges, update the Perforce trigger table by running `p4 triggers` command and replacing any `swarm.*` lines with the previously captured trigger line output (using **Ctrl+V** on Windows and Linux).

Important:

If you previously customized the P4 Code Review trigger lines, perhaps to apply various ["Trigger options" on page 723](#), be sure to repeat those customizations within the updated trigger lines.

6. Continue with the next task, see ["Replacing your original P4 Code Review instance with your new instance" below](#).

Replacing your original P4 Code Review instance with your new instance

Warning:

Downtime occurs in this step.

Replace your original P4 Code Review instance with your new P4 Code Review instance.

1. Stop your original P4 Code Review instance:

```
sudo apache2ctl stop
```

2. Start your new P4 Code Review instance:

```
sudo apache2ctl start
```

Note:

If you see the following error message when you start P4 Code Review, P4 Code Review is using the wrong version of P4 API for PHP. The latest version of P4 API for PHP is included with P4 Code Review.

If you are installing P4 Code Review from a tarball, you must configure P4 Code Review to use that version of P4 API for PHP. For instructions about how to configure P4 Code Review to use the new version of P4 API for PHP, see ["PHP configuration" on page 174](#).

Swarm has detected a configuration error

Problem detected:

- The Perforce PHP extension (P4PHP) requires upgrading. Found 2016.2, only 2018.2 or later is supported.

The php.ini file loaded is /etc/php5/apache2/php.ini.

Other scanned php.ini files (in /etc/php5/apache2/conf.d) include:

- /etc/php5/apache2/conf.d/imagick.ini
- /etc/php5/apache2/conf.d/pdo.ini
- /etc/php5/apache2/conf.d/perforce.ini

Please investigate the below PHP error:

The Perforce PHP extension (P4PHP) requires upgrading.
Found 2016.2, only 2018.2 or later is supported.

/var/www/deployments/1703508/swarm/public/index.php on line 104

For more information, please see the [Setting Up](#) documentation; in particular:

- [Initial Installation](#)
- [Runtime dependencies](#)
- [PHP configuration](#)
- [Swarm configuration](#)

Please ensure you restart your web server after making any PHP changes.

3. Continue with the next task, see ["Validating your new P4 Code Review instance" below](#).

Validating your new P4 Code Review instance

Tip:

When P4 Code Review starts it verifies the Redis cache, during this time you cannot log in to P4 Code Review. The time taken to verify the Redis cache depends on the number of users, groups, and projects P4 Code Review has. Start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves in the redis-server.conf file, see ["Redis server configuration file" on page 665](#).

Check that your upgraded P4 Code Review instance is working correctly by doing the following:

1. Create a new changelist that:
 - a. Contains at least one modified file
 - b. Contains the `#review` keyword in the changelist description
2. Right click on the new changelist in P4V and click **Shelve Files...**

Important:

Do not select **Request New Review...** because this method uses the API and will not fully test the P4 Server extension.

This is also true if you are using P4 Code Review triggers instead of the P4 Server extension.

3. Check that a new review is created for the changelist.
 - If a review is created, the P4 Server extension is working. If you are using P4 Code Review triggers instead of the P4 Server extension and the review is created, the triggers are working.
 - If a review is not created, see "[Troubleshooting: Review not created](#)" on page 228.
4. Do one of the following:
 - **If your new P4 Code Review instance is working correctly:** Retire your original P4 Code Review instance.
 - **If your new P4 Code Review instance is not working correctly:** Shutdown the new P4 Code Review instance, remove any updates you made to the triggers or extensions on the P4 Server, and switch back to your original P4 Code Review instance. Check the configuration and connectivity of your new P4 Code Review instance. If you can't find the issue, you can submit a [Support Request](#).

Additional security measures

This section outlines additional security measures to enhance the security of your Swarm installation.

Hide your Apache version and Linux OS

Verbose user interfaces and banner-grabbing applications show system information to users looking for version-specific vulnerabilities in your server environment. The system information can be used for various purposes, such as marketing and competitor analysis. Hiding your OS and Apache version adds a degree of difficulty for potential cyber attackers.

View server HTTP headers

There are many ways to view a server's HTTP headers. The easiest option is to use an online tool such as [Security Headers](#) or [Mozilla Observatory](#).

If you're on a Linux system, you can use the `curl` or `wget` terminal commands:

```
curl --head yourdomain.com
```

```
wget --server-response --spider yourdomain.com
```

If you are logged into the Linux system that you want to modify, use localhost in the following commands:

```
curl --head localhost
```

```
wget --server-response --spider localhost
```

Within the header information you will see a line that states the web server software and version you are using alongside your server OS.

For example: `Server: Apache/2.4.10 (Debian)`

Hide Apache Version and OS

To remove your Apache version and OS from HTTP headers and server-generated pages, do the following:

1. Log into SSH (Secure Shell protocol) as root.
2. Edit your Apache server configuration file using a text editor.

CentOS/AlmaLinux:

```
nano /etc/httpd/conf/httpd.conf
```

Debian/Ubuntu:

```
nano /etc/apache2/conf-enabled/security.conf
```

3. Scroll down to the “ServerTokens” section where you will probably see multiple lines commented out (beginning with “#”) stating “ServerTokens” and different options. Change the uncommented line, likely “ServerTokens OS”, or comment out the line and create a new line to hide the Apache version and OS from HTTP headers: `ServerTokens Prod`

If you do not see the “ServerTokens” and “ServerSignature” sections, add the necessary lines to the bottom of your configuration file.

4. The next section down should be the “ServerSignature” section. Turning this off hides the information from server-generated pages. For example, Internal Server Error.

```
ServerSignature Off
```

5. Save the changes and close the file.
6. Restart Apache server as follows:

CentOS/AlmaLinux:

```
systemctl restart httpd
```

Debian /Ubuntu:

```
systemctl restart apache2
```

7. Recheck your server HTTP headers:

```
curl --head localhost
```

Disable Apache directory listings

For security best practices, Apache directory listings should be disabled. This prevents unauthorized users from browsing the contents of your server's directories. The `Options -Indexes` directive achieves this in the Apache configuration or a `.htaccess` file. Specific directories can be configured to allow listings if required.

Security risks of using a self-signed certificate

Using a self-signed certificate or untrusted Certificate Authorities exposes your system to vulnerabilities and security breaches. The risks of using a self-signed certificate or untrusted Certificate Authorities are as follows:

- **Not trusted by web browsers:** Self-signed certificates or untrusted Certificate Authorities are not trusted by web browsers because they can make it easy for malicious attackers to intercept users' data shared through the server. These certificates contain both private and public keys within the same entity, which cannot be revoked, making it difficult to detect security compromises.
- **Exposure to vulnerabilities:** Compromised private keys can be a threat to an organization's infrastructure. Certificate authorities can identify compromised certificates and revoke them. However, organizations cannot revoke self-signed certificates and may not be able to keep track of them, leading to compromised certificates being overlooked or going unnoticed. These compromised certificates can provide opportunities for malicious actors to access the network and carry out advanced malware attacks, man-in-the-middle (MITM) attacks, phishing attacks, and botnets.
- **Not meeting security requirements:** Digital certificates issued by trusted certificate authorities adhere to strong cyber security standards, utilizing the latest ciphers and hashing technologies. On the other hand, self-signed certificates are created internally and may not always meet the latest security standards, potentially using weaker ciphers.

4 | Basics

This chapter covers the basic operations provided by P4 Code Review. These include:

In this section:























My Dashboard

The purpose of the dashboard is to allow you to focus on reviews that need to be done, so that other users are not blocked. The dashboard lists the most recently modified reviews first and shows your role in the review. The dashboard displays a maximum of 25 reviews. Perform an action on the current list of reviews to see if any more reviews are available.

Your dashboard is available by clicking **My Dashboard** in the menu or by clicking the P4 Code Review icon above the main menu.



Tip: My Dashboard is only populated if you are logged in.

My Dashboard						
Project <input type="text"/> Role <input type="text"/> Author <input type="text"/>			There are new changes to your review list. Clear all		<input type="text" value="Search for a review"/>	
Description	Your Role	Last Activity	State	Tests	Complexity	Votes
 Updated README to have a bit more information. Jack Boone requested a  pre-commit review 365 for mercury:dev	Reviewer	44 minutes ago			 137	 0  0
 Updated notes for the new year. Jack Boone requested a  post-commit review 422 for jplugin:main	Required reviewer	2 months ago			 18	 0  0
 Add comment to method. Steve Russell requested a  post-commit review 328 for jplugin:main	Required reviewer	3 months ago			 6	 2  0
 All available reviews are displayed						

A review is displayed on your dashboard if any of the following criteria are met:

- You are a reviewer or required reviewer, the review status is **Needs Review** and you have not already voted on it.
- You are a member of a reviewer group or a required reviewer group, the review status is **Needs Review** and you have not already voted on it. The review will remain on your dashboard even if the group has met its criteria if you have not already voted on it.
- You are the review author and the review status is **Needs Revision**, or **Approved** (only if the review is approved but not committed).
- You are a moderator or a member of a moderator group, the review status is **Needs Review**, the **Minimum up votes** requirement for the branch is satisfied, and one of the following is true:

- There are no required reviewers.
- All of the required reviewers have up-voted the review.
- You are the last remaining required reviewer for the review.

Tip:

If moderator behavior is configured to require approval from one moderator per branch and the review spans multiple moderated branches:

- You will see the review in your dashboard if the review has not been approved by another moderator from your branch.
- You will be able to **Approve and Commit** the review if it has been approved by moderators from all of the other branches.

Filtering



The filters are used to filter your dashboard reviews:

- **Project:** filter by the project the review is part of, limiting results to **My Projects** (projects you are an owner of or a member of) or specific projects. To select multiple projects, click the projects you want to filter by. The **Project** filter will only show projects for which there are reviews in your dashboard.
- **Role:** filter by your specific role in a review, limiting results to reviews for which you are the author, a reviewer, a required reviewer, or a moderator. The **Role** filter will only show roles for which there are reviews in your dashboard.
- **Author:** filter the reviews to only those that have been authored by a certain user. Select the author from the drop down list of users. The **Author** filter will only show authors for which there are reviews in your dashboard.
- **Clear all:** click to reset all dashboard filters back to their defaults.
- **Search:** filter the reviews by searching the review descriptions.

Refresh button








Refresh button (only displayed when new content is available for your dashboard): click **Refresh** to update your dashboard with the new content.


Note:

By default, when your dashboard is open, P4 Code Review checks for new content every five minutes.

Review summaries




The dashboard displays a summary for each review.

Description	Your Role	Last Activity	State	Tests	Complexity	Votes
 Updated README to have a bit more information. Jack Boone requested a  pre-commit review 365 for mercury:dev	Reviewer	about an hour ago			 137	 2  0

- **Avatar** displays the avatar of the review author, click to view the profile of the author, see ["Viewing another user's profile" on page 350](#).
- **Description** contains:
 - first line of the review description, click to open the review. If the review description is too long it is truncated, click on the ellipsis  to expand it in the list page.
 - review author, click to view the profile of the author.
 - review type, either a [pre-commit review](#) or a [post-commit review](#).
 - review number, click to open the review, see ["Review display" on page 411](#).
 - review project branches, click to open the project.
- **Your role:** displays your role in the review and is the reason the review is in your dashboard. This can be Author, Reviewer, Required Reviewer, or Moderator.
- **Last activity:** displays the last time that any changes were made to the review, including votes, comments, commits, and file changes.
- **State:** a review can be in one of the following states:
 - **Needs review:** The review has started and the changes need to be reviewed.
 - **Needs revision:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
 - **Approved:** The review has been approved. The changes may need to be committed.

Note:

The Approved state only applies to review authors, it is only shown for a review that has been approved but has not been committed.

- **Tests:** shows the test suite state for the review, either tests in progress , tests passed , or tests failed .
- **Complexity:** a traffic light icon and number shows the relative complexity of the review and the total number of lines changed in the review. [Complexity can be configured](#) by your P4 Code Review administrator but, by default:
 - **Red:** ≥ 300 changes
 - **Amber:** < 300 and > 30 changes
 - **Green:** ≤ 30 changes

Tip:

- Review complexity is only calculated for a review when the review is updated and the file content has changed.
- Review complexity is only stored for the current version of a review.

Hover over the complexity icon to display more detailed information:



- **Votes:** displays the number of up votes and down votes for the review.

Global Dashboard

Note:

Global Dashboard is only available if your P4 Code Review administrator has configured P4 Code Review to connect to more than one P4 Server instance. For instructions on how to configure P4 Code Review to connect to multiple P4 Server instances, see ["Multiple P4 Server instances" on page 638](#).

Your Global Dashboard displays a list of reviews that you may need to act on for each of the P4 Server instances connected to P4 Code Review.

Open your global dashboard:

1. Enter the basic P4 Code Review URL without a P4 Server instance name, for example:
`https://swarm.company.com`
2. If the log in dialog is displayed:

Tip:

Single Sign-On (SSO)

If P4 AS is configured for your P4 Server and P4 Code Review, [SSO can be configured](#) by your P4 Code Review admin as either mandatory or optional:

- If SSO is mandatory, you will be directed to the sign-in process used by your Identity Provider (IdP).
- If SSO is optional, log in to P4 Code Review using one of the following methods:
 - Click **Log in with SSO**, you will be directed to the sign-in process used by your Identity Provider (IdP).
 - Type in your username and password, and click **Log in**.

a. **Server:**

- If your P4 Server instances do not share log in credentials, select an individual P4 Server instance from the dropdown list.
- If all of your P4 Server instances share the same log in credentials, leave set to **All available servers**.

Note:

If P4 AS is configured for your P4 Server and P4 Code Review, you must log into the servers individually.

b. Enter your username and password.

Select the **Remember Me** check box if you prefer to stay logged in between browser restarts.

Note:

P4 Servers can enforce maximum login times. You may become logged out even if **Remember Me** is selected. P4 Code Review administrators can change the maximum login time, see ["Sessions" on page 687](#) for details.

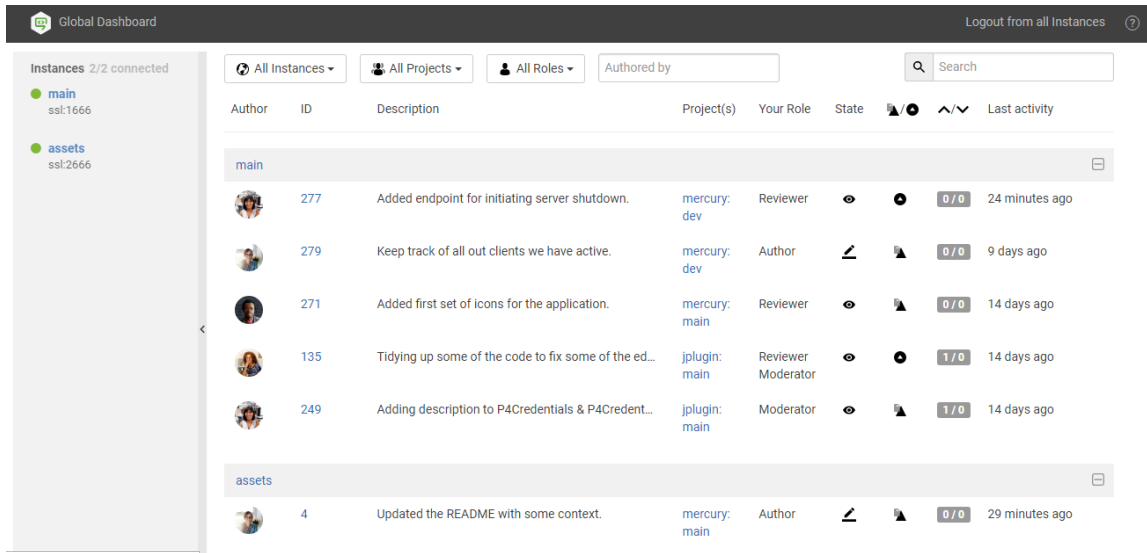
c. Click **Connect**.

P4 Code Review will populate the global dashboard from the P4 Server instances it connects to.

The global dashboard is displayed:

Tip:

- Since it is tied to the logged in user, the global dashboard is only populated for the P4 Server instances you are logged in to in P4 Code Review.
- If you are already logged in to a P4 Server instance in P4 Code Review, the dashboard for that server will be automatically populated when you open or refresh the global dashboard.



The global dashboard is made up of three main areas:

- **"Toolbar" below:** log in to P4 Server instances, log out of all P4 Server instances, and open the P4 Code Review help from the global dashboard toolbar.
- **"Sidebar" on page 296:** Log in/logout of individual P4 Server instances and view you profile for any instance that you are logged in to.
- **"P4 Server dashboards" on page 298:** view, filter and search the dashboards of the P4 Server instances you are logged in to. Jump directly to a P4 Server, review, or project by clicking on a link.

Toolbar

Log in to one or more P4 Server instances

Log In is only displayed in the global dashboard toolbar if you are not logged in to any P4 Server instances. If **Log In** is not available, see ["Log in to a P4 Server instance" on page 296](#).

1. In the global dashboard toolbar, click **Log In**.

Tip:
Single Sign-On (SSO)

If P4 AS is configured for your P4 Server and P4 Code Review, [SSO can be configured](#) by your P4 Code Review admin as either mandatory or optional:

- If SSO is mandatory, you will be directed to the sign-in process used by your Identity Provider (IdP).
- If SSO is optional, log in to P4 Code Review using one of the following methods:
 - Click **Log in with SSO**, you will be directed to the sign-in process used by your Identity Provider (IdP).
 - Type in your username and password, and click **Log in**.



Welcome to Global Dashboard

You must be connected to at least one Instance in order to access your Dashboard across all of your Instances. [Need more help?](#)

Server

 x ▼

User Name

Password

☐ Remember Me

 **Connect**

2. **Server:**

- If all of your P4 Server instances share the same log in credentials, leave set to **All available servers**.
- If your P4 Server instances do not share log in credentials, select an individual P4 Server instance from the dropdown list.

Note:

If P4 AS is configured for your P4 Server and P4 Code Review, you must log into the servers individually.

Tip:

If you have a number of P4 Server instances that use the same login credentials, select them one at a time from the dropdown list.

3. Enter your username and password.
Select the **Remember Me** check box if you prefer to stay logged in between browser restarts.

Note:

P4 Servers can enforce maximum login times. You may become logged out even if **Remember Me** is selected. P4 Code Review administrators can change the maximum login time, see ["Sessions" on page 687](#) for details.

4. Click **Connect**.

P4 Code Review will populate the global dashboard from the P4 Server instances it connects to.

Logout of all P4 Server instances

Logout from all instances is only displayed in the global dashboard toolbar if you are logged in to at least one P4 Server instance.

1. In the global dashboard toolbar, click **Logout from all instances**.
2. When prompted, click **Yes** to confirm that you want to logout of all of the P4 Server instances.

Tip:

- If P4 AS is configured for your P4 Server, logging out of P4 Code Review will not invalidate your Identity Provider (IdP) login status. If you try to log back in to P4 Code Review while your IdP status is still valid, you will not be prompted to complete the log in steps.

If `require_login` is also enabled, P4 Code Review will return you to the login and your IdP will automatically log you back in. In this case log out from your Identity Provider page before logging out from P4 Code Review.

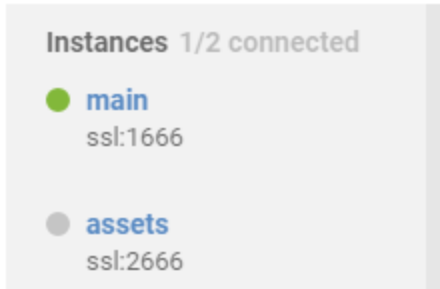
- If a custom redirect has been configured by your P4 Code Review administrator, you are logged out of all of the P4 Server instances by P4 Code Review and then redirected to the URL specified by the administrator.

The custom redirect can be set to any internal or external URL, for example:

- Company intranet, extranet, internet, FTP, or Web-mail page
- Industry news website
- Identity Provider page to invalidate your IdP log in status

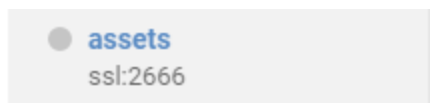
Sidebar

The P4 Servers that P4 Code Review can connect to are listed in the collapsible sidebar on the left of the page. From the server list you can log in/logout from individual P4 Servers and view your user profile.

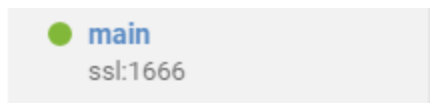


Log in status


- Not logged in to the P4 Server instance:



- Logged in to the P4 Server instance:



Log in to a P4 Server instance

1. In the sidebar, hover over the P4 Server you want to connect to.
2. Click the **Log in** button .

Tip:

- **Single Sign-On (SSO)**

If P4 AS is configured for your P4 Server and P4 Code Review, [SSO can be configured](#) by your P4 Code Review admin as either mandatory or optional:

- If SSO is mandatory, you will be directed to the sign-in process used by your Identity Provider (IdP).
- If SSO is optional, log in to P4 Code Review using one of the following methods:
 - Click **Log in with SSO**, you will be directed to the sign-in process used by your Identity Provider (IdP).
 - Type in your username and password, and click **Log in**.
- If you are already logged in to another P4 Server that is configured for P4 AS, your IdP status is valid and you will not be prompted to complete the log in steps.

Log In ✕

assets

ssl:2666

User Name

Password


☒ Try to log in to all available servers with these credentials.

☐ Remember Me



Connect

3. Type in your username and password for the P4 Server.

- Select the **Try to log in to all available servers with these credentials** checkbox if you use these credentials for more than one of the P4 Server instances. P4 Code Review will not try to log in to any P4 Server instances that are configured for P4 AS, log in to them individually using the instance **Log in** button  in the sidebar.

- Select the **Remember Me** check box if you prefer to stay logged in between browser restarts.

Note:


P4 Servers can enforce maximum login times. You may become logged out even if **Remember Me** is selected. P4 Code Review administrators can change the maximum login time, see ["Sessions" on page 687](#) for details.

4. Click **Connect**.

P4 Code Review will populate the global dashboard from the P4 Server.

If you selected the **Try to log in to all available servers with these credentials** checkbox, P4 Code Review will populate the global dashboard for the other servers in the list that it successfully connects to.

Logout of a single P4 Server instance

1. In the sidebar, hover over the P4 Server you want to log out from.
2. Click the **Gear** button  to open the dropdown menu.
3. Click **Logout** in the dropdown menu.

Tip:

- If P4 AS is configured for your P4 Server, logging out of P4 Code Review will not invalidate your Identity Provider (IdP) login status. If you try to log back in to P4 Code Review while your IdP status is still valid, you will not be prompted to complete the log in steps.

If `require_login` is also enabled, P4 Code Review will return you to the login and your IdP will automatically log you back in. In this case log out from your Identity Provider page before logging out from P4 Code Review.

- If a custom redirect has been configured by your P4 Code Review administrator, you are logged out of the P4 Server instance by P4 Code Review. You are only redirected to the URL specified by the administrator if you are not logged in to any other P4 Server instances on the global dashboard.

The custom redirect can be set to any internal or external URL, for example:

- Company intranet, extranet, internet, FTP, or Web-mail page
- Industry news website
- Identity Provider page to invalidate your IdP log in status

P4 Server dashboards

The purpose of the global dashboard is to allow you to focus on reviews that need to be done, so that other users are not blocked. The global dashboard lists reviews by P4 Server according to the most recently modified first, and shows your role in the review.

A review is displayed on your global dashboard if any of the following criteria are met:

- You are a reviewer or required reviewer, the review status is **Needs Review** and you have not already voted on it.
- You are a member of a reviewer group or a required reviewer group, the review status is **Needs Review** and you have not already voted on it. The review will remain on your dashboard even if the group has met its criteria if you have not already voted on it.
- You are the review author and the review status is **Needs Revision**, or **Approved** (only if the review is approved but not committed).
- You are a moderator or a member of a moderator group, the review status is **Needs Review**, the **Minimum up votes** requirement for the branch is satisfied, and one of the following is true:
 - There are no required reviewers.
 - All of the required reviewers have up-voted the review.
 - You are the last remaining required reviewer for the review.

Tip:

If moderator behavior is configured to require approval from one moderator per branch and the review spans multiple moderated branches:

- You will see the review in your dashboard if the review has not been approved by another moderator from your branch.
- You will be able to **Approve and Commit** the review if it has been approved by moderators from all of the other branches.

Example global dashboard showing a number of P4 Server dashboards:

<div> <div>All Instances ▾</div> <div>All Projects ▾</div> <div>All Roles ▾</div> <div>Authored by</div> <div> <input type="text" value="Search"/> </div> </div>									
Author	ID	Description	Project(s)	Your Role	State				Last activity
main									
	277	Added endpoint for initiating server shutdown.	mercury: dev	Reviewer				0 / 0	24 minutes ago
	279	Keep track of all out clients we have active.	mercury: dev	Author				0 / 0	9 days ago
	271	Added first set of icons for the application.	mercury: main	Reviewer				0 / 0	14 days ago
	135	Tidying up some of the code to fix some of the ed...	jplugin: main	Reviewer Moderator				1 / 0	14 days ago
	249	Adding description to P4Credentials & P4Credent...	jplugin: main	Moderator				1 / 0	14 days ago
assets									
	4	Updated the README with some context.	mercury: main	Author				0 / 0	29 minutes ago

P4 Server header bars

The dashboard for each P4 Server instance is displayed under the header bar for that instance. The header bars are collapsible allowing you to temporarily hide P4 Server instances you are not currently interested in.

Tip:

- Click on the P4 Server name in the header to open P4 Code Review for that instance in a new tab.
- Click on the header to the right of the P4 Server to collapse the dashboard for that instance. Click again to expand the dashboard for the instance .

Filtering

The global dashboard can be filtered to display only reviews from a particular P4 Server instance, project, authored by a particular user, or matching a role. The filter buttons filter all of the instances on the global dashboard. The filter buttons are always in view as you scroll up and down the page so that you can quickly modify the filters and see just the reviews you want to see. Click the **Reset** button to reset these filters.

Filtering options are:

- **Instances**

You can filter by the P4 Server instance, limiting results to **All Instances** or to an individual instance.

- **Projects**

You can filter by the project the review is part of, limiting results to **All Projects** or to an individual project. The **Project** filter will only show projects for which there are reviews in your dashboard.

- **Roles**

You can filter by your specific role in a review, limiting results to reviews for which you are the author, a reviewer, a required reviewer, or a moderator. The **Role** filter will only show roles for which there are reviews in your dashboard.

- **Authored by**

You can filter the reviews to only those that have been authored by a certain user. Type in this field to get a drop down list of users to filter by.

- **Reset** (only displayed if one or more filters are set):

Clicking the **Reset** button resets all dashboard filters back to their defaults.

- **Search**

Typing in the search field filters the reviews by description and review ID.

Review fields

The dashboard for each P4 Server shows a summary of the information for each review.

Author	ID	Description	Project(s)	Your Role	State		Last activity
main							
	277	Added endpoint for initiating server shutdown.	mercury: dev	Reviewer			0 / 0 1 hour ago

Reviews that appear here are those which are waiting for action from you. The information presented should help you prioritize what to work on next.

- **Author**

The author of this review.

- **ID**

The ID of this review. Click on this to go to the review page.

- **Description**

The review description. It may be truncated if it is too long, in which case click on the description to expand it.

- **Project(s)**

List of project branches this review covers. A review may span multiple branches and projects. Click on one of them to navigate to the project page for that branch.

- **Your role**

The reason this review is in your dashboard. This can be Author, Reviewer, Required Reviewer, or Moderator.

- **State**

The current status of the review. This can be Needs Review, Needs Revision, or Approved.

Note:

The Approved state only applies to review authors, it is only shown for a review that has been approved but has not been committed.

- **Type**

The type of review. This can be Pre-commit or Post-commit.

- **Votes**

The double column of votes displays the number of up votes and down votes for the review.

- **Last activity**

The last time that any changes were made to the review, including votes, comments, commits, and file changes.

Navigating directly to a specific P4 Server instance in P4 Code Review

If you want to quickly visit a specific P4 Server instance in P4 Code Review without going via the global dashboard, include the server name in the URL, for example: `https://swarm.company.com/serverA`.

Once you are viewing the P4 Server in P4 Code Review, P4 Code Review works as a standard single P4 Server-P4 Code Review system.

Tip:

- To browse jobs on **serverA**, navigate to: <https://swarm.company.com/serverA/jobs>
- To browse reviews on **serverB**, navigate to:
<https://swarm.company.com/serverB/reviews>
- To view the dashboard for **serverB**, navigate to
<https://swarm.company.com/serverB/#actionable-reviews>

Activity

Tip:

Activity streams are also shown on the [User](#), [Group](#), and [Project](#) pages. These activity pages show the activity for the user, project, or group you are viewing.


An activity stream displays a list of events that have occurred recently in the associated P4 Server.


For example, an event is added to the activity stream whenever:

- Changelists are checked in
- Jobs are created
- Code reviews are updated
- Comments posted
- Projects are created or edited
- Workflows are created, edited, or deleted
- Tests are created, edited, deleted, pass, or fail
- Deleted users are removed from workflows, projects, groups, test definition, and project followers

To view the main activity stream:


1. Log in to P4 Code Review.
2. Click **Activity** in the P4 Code Review menu.


All Activity ▾ Reviews Commits Comments Jobs 




paula.boyle updated description of [review 135 \(revision 2\)](#) for [main](#)
Tidying up some of the code to fix some of the edge conditions. 2 minutes ago


Also putting some support for future features in place. #review [@@trriage](#)

 [Add a comment](#)




claire.brevia voted up [review 185 \(revision 1\)](#) for [main](#)
These browsers are no longer supported by the product and the code
should be removed. 4 minutes ago


 [Add a comment](#)



alex.randolph cleared their vote on [review 135 \(revision 2\)](#) for [main](#)
Tidying up some of the code to fix some edge conditions. 6 minutes ago


Also putting some support for future features in place. #review [@@trriage](#)

 [Add a comment](#)



alex.randolph updated description of [review 135](#) for [main](#)
Tidying up some of the code to fix some edge conditions. 13 days ago

Also putting some support for future features in place. #review [@@trriage](#)

 [Add a comment](#)

Activity dropdown menu (main activity stream only):

- **All Activity:** Select to view all activity on the P4 Server
- **Followed Activity:** Select to view just the activity of the projects and users that they are following

Activity streams can be filtered to only display events related to **Reviews**, **Commits** (changes), **Comments**, or **Jobs**. Click a filter label to enable that filter. Click the label again to disable the filter, or click a different filter to change filters. When active, each filter label displays a distinct background color that matches the color stripe on the right side of each event in the activity stream, to help quickly identify each event's type.



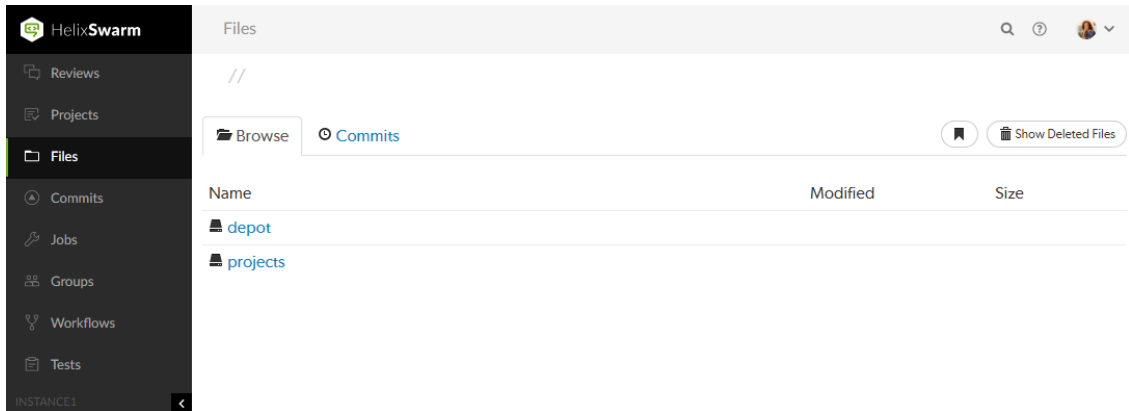
(RSS) button: Click to subscribe to an RSS (Really Simply Syndication) feed for a particular activity stream so that events can be monitored in your favorite feed reader.

Events within an activity stream contain links to their resources. Click the links to visit users, changelists, projects, comments, and more.

Each activity stream starts with as many as 50 events. As you scroll down the page, P4 Code Review fetches additional events in batches of 50 until the stream is exhausted. For a long-running server, or a server with significant activity, it could take quite a while to receive all of the events.




Files

P4 Server's primary task is to version files, so P4 Code Review makes it easy to browse the depot. Start browsing by clicking **Files** in the menu.



- P4 Code Review displays a list of breadcrumb links to help you quickly navigate to higher level directories quickly.

[// depot](#) / [Jam](#) / [MAIN](#) / [src](#) / [Jam.html #2](#)

- Links with folder icons represent directories of files within the depot. Click a directory link  to display the contents of that directory.
- Click the .. link with the up-arrow icon  [..](#), when it appears, to navigate to the current directory's parent.
- Links with dog-eared page icons  represent individual files within the depot. See ["File display" on the next page](#) for more information.
- Click the **Commits** tab to display the list of changes made to files in the current directory, or any directories it contains. See ["Commits" on page 313](#) for more information:



P4 Code Review creates links for files and directories wherever they appear in the P4 Code Review UI.

Note:

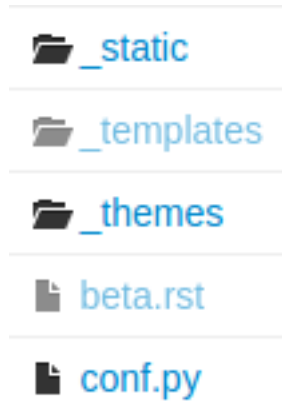
Directories that start with a period, for example `.foo`, are sorted to appear at the end of the list of directories. The `..` link, when it appears, always appears first.

Similarly, files that start with a period, for example `.htaccess`, are sorted to appear at the end of the list of files.

Browsing deleted files and folders

When the **Show Deleted Files** button is clicked, P4 Code Review toggles the inclusion of deleted folders and files in the file display.

Deleted folders and files are presented slightly muted compared to non-deleted entries:



File display

When P4 Code Review is asked to display a file, if the file is a type that P4 Code Review can display, P4 Code Review presents the file's contents.

Buttons: the actual buttons displayed depends on the file type you are viewing.

- Click the **Shorten URL** button to display a short-link to the file.
- Click the **Blame** button to add a column to the display that identifies the userid responsible for each line of the file.
- Click the **Open** button to display the file content with no surrounding page markup.
- Click the **Edit** button to edit the file, see ["File edit" on page 309](#).
- Click the **Download .zip** button to download a compressed version of the file, see ["Download files as a ZIP archive" on page 310](#).
- Click the **Download** button to download the file.

When a file is opened it will by default be truncated to 1 MB in size. This limit can be increased (or removed) via the P4 Code Review configurables. See [Files configuration](#) for details. This limit does not apply to downloaded files.

Along with the file name, P4 Code Review displays the version number for the currently displayed file in the breadcrumb. For example, the following breadcrumb indicates that version 2 of `Jam.html` is being displayed:

[// depot](#) / [Jam](#) / [MAIN](#) / [src](#) / [Jam.html #2](#)

If the version of a file being previewed has been deleted, the version number is displayed in red. If you rename a file, edit it, and request a review, only the edited content is highlighted in the diff panel.

Every version of a file is available on the [Commits](#) tab.

Text files

P4 Code Review displays the contents of text files (including the P4 Server filetypes unicode and UTF16) with line numbers. When possible, syntax highlighting is applied to make identification of various elements within the file easier.

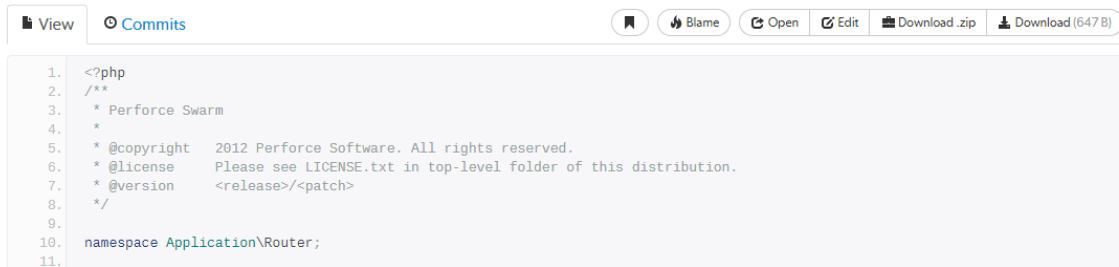
For more information on the supported extensions for syntax highlighting in the Review page, see ["Supported syntax highlighting in the Review page" on page 311](#).

For more information on P4 Server filetypes, see [File Types](#) in [P4 CLI Reference](#).

Tip:

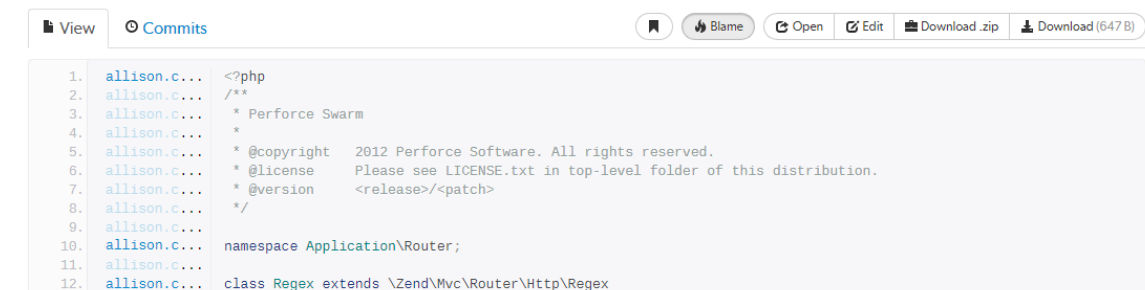
P4 Code Review can be configured to display some non-UTF8 characters, see ["Non-UTF8 character display" on page 624](#) for details.

// helix-swarm / development / module / Application / src / Application / Router / Regex.php #2



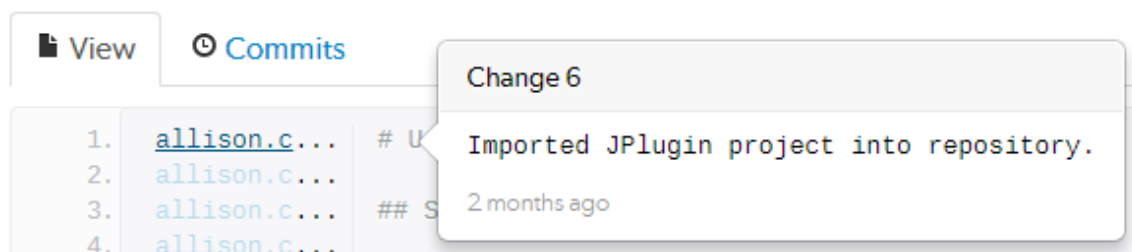
Click **Blame** to add a column to the display that identifies the userid responsible for each line of the file:

// helix-swarm / development / module / Application / src / Application / Router / Regex.php #2



Each userid presented is a link that, when clicked, displays the changelist that provided the associated text. Muted userids indicate that the associated text is from the same changelist as the line above. For example, the userid *allison.c* is responsible for lines 1, 10, and 12 in the screenshot above.

When you hover your pointer over a userid in the blame column, a tooltip appears displaying the associated changelist description:



When there are no lines displayed, for example when you are viewing empty or shelved files, the **Blame** button is disabled.

Markdown files

P4 Code Review displays Markdown files in two tabs:

- **Markdown:** by default, Markdown content is displayed but it is limited to prevent the execution of raw HTML and JavaScript content. This can be configured by the P4 Code Review administrator, see ["Markdown" on page 631](#). This tab is only displayed for Markdown files.
- **View:** displays the contents of Markdown files as plain text in the same way as ["Text files" on page 305](#) are displayed.

Tip:

Valid Markdown file extensions are: md, markdown, mdown, mkdn, mkd, mdwn, mdtxt, mdtext.

Markdown tab

Click the **Markdown** tab to view the file content in Markdown:

// depot / main / file.md #2



I'm a markdown file

View tab

Click the **View** tab to view the Markdown file in plain text:

// depot / main / file.md #2



```
1. # I'm a markdown file
```

Images

P4 Code Review displays web-safe images:

// helix-swarm / development / public / swarm / img / logo-lg.png #2



The checkerboard background in this example is not part of the logo; it helps identify where transparency exists.

Many browsers can display SVG images with no additional plugins, so P4 Code Review attempts to display SVG images rather than displaying the image's definition. When you use a browser that cannot natively display SVG images, you see the broken image icon.

When imagick (an optional module that integrates ImageMagick into PHP) is installed, P4 Code Review can also display the following image formats: BMP, EPS, PSD, TGA, TIFF.

3D models

Important:

You can view the 3D model file types only in the classic view of the review page. Use [P4 DAM](#), a digital asset management software solution by Perforce to view 3D model files. For more information about how to use P4 DAM, see [Helix DAM User Guide](#).

P4 Code Review includes support for displaying select 3D model file types in the browser:



Supported file types include:

- **DAE** - including any referenced web-safe texture images.
- **STL** - both binary and ASCII versions of the format.
- **OBJ** - including any referenced MTL files, and web-safe texture images.

When P4 Code Review can display a 3D model, it renders a generic grid *stage* and places the model in the center, scaled to make viewing straightforward. A toggle control appears in the top right: when enabled, you can control the view with the mouse, and when not enabled, permits auto-rotation to occur (when possible).

1. Click and hold the **left mouse button** to begin rotating the view. Drag while holding the left mouse button to rotate the view.
2. Click and hold the **right mouse button** to begin panning the view. Drag while holding the right mouse button to pan the view.
3. Roll the **mouse wheel** up or down to adjust the magnification of the view.

When possible, a second control appears allowing you to toggle between showing the model with surfaces, or just showing the model's wireframe.

Note:

For systems with hardware acceleration, if your browser supports WebGL and hardware acceleration is enabled, P4 Code Review renders the model and enables auto-rotation.

For systems without hardware acceleration or WebGL, but your browser supports HTML5 canvas elements and JavaScript TypedArrays, P4 Code Review renders the model but auto-rotation is disabled. Rendering is likely to be slow and rendering quality is likely to be low.

For browsers without HTML5 canvas elements and JavaScript TypedArrays, no rendering is attempted; instead, users see a message indicating that the browser is not supported.

Other file types

It is possible to view other file types in P4 Code Review, through the addition of additional modules, or by installing "[LibreOffice](#)" on page 543 on the P4 Code Review host.

When the file is a type that P4 Code Review cannot display, P4 Code Review presents the file's [history](#), along with the **Download** button:

// helix-swarm / development / collateral / build-utils / ant-contrib.jar #1

Commits

Download .zip

Download (219 KB)

#	Change	User	Description	Committed
#1	541195	steve.russell	Initial set of build infrastructure: * Ant build.xml to drive it all * Ant Contrib for s...	6 years ago

File edit

Important:

File edit is a Technology Preview feature.

Features offered in Technology Preview are experimental and not guaranteed to always work as expected. If you have feedback and functionality suggestions, email techpreview@perforce.com.

Note:

Requirements:

- You must have permissions to edit the file.
- If the file version changes while you are editing it or if you are not editing the Head revision of the file, committing your file changes will overwrite the current head revision of the file.

If you are unsure, shelving your change is a safer option because it will not overwrite the head revision of the file and you can do a manual resolve when you commit the shelf.

- File edit from the **Files** browser page is enabled by default but can be disabled by your P4 Code Review administrator, see "[allow_edits](#)" on page 615.

You can edit a file from the P4 Code Review **Files** browser page and shelve or commit your file changes.

To edit a file from the P4 Code Review Files page:

1. Navigate to the file on the **Files** page.
2. Click the **Edit** button above the file to open the file editor.

Edit file

projects / acme / main / README.md#1

1

ACME

2

3

This is the ACME project. It is a nice project.

4

5

Other Things

6

7

Nothing much here.

8

9

A nice new line has been added here.

Changelist description

Added new line of text at line 9.

#review

Discard changes

Commit

Shelve

3. Edit the file and add a **Changelist description**.

Tip:

Optional: you can create a review for your change or add it to an existing review by including a review keyword in the **Change description**. For instruction on creating a review and adding a changelist to a review using review keywords, see ["Create a review" on page 669](#) and ["Add a changelist to a review" on page 670](#).

4. Click **Commit** to commit the file or **Shelve** to shelve the file.

To discard your changes, click **Discard changes** and confirm that you want to discard your changes when requested.

Download files as a ZIP archive

The **Download .zip** button is used to download the selected files or folders in the P4 Server as a ZIP archive. This makes it easy to get a copy of files without having to setup a client.

Note:

The **Download zip** option is not displayed if the zip command-line tool is not installed on the P4 Code Review server. For information about installing, and configuring the zip command-line tool, see [Zip archive](#).

When you select the **Download zip** option, P4 Code Review performs the following steps:

1. Scans the files/folders:
 - Checks that you have permission to access their contents, according to the P4 Server protections.
 - Checks that the total file size is small enough to be processed by P4 Code Review.
2. Syncs the file contents to the P4 Code Review server from the P4 Server.
3. Creates the ZIP archive by compressing the file content.
4. Starts a download of the generated ZIP archive.

Note:

- You might not see all of the above steps; P4 Code Review caches the resulting ZIP archives so that repeated requests to the same files/folders can skip the sync and compress steps whenever possible.
- If an error occurs while scanning, syncing, or compressing, P4 Code Review indicates the error.

Supported syntax highlighting in the Review page

P4 Code Review displays the code in the default syntax colors for that coding language. However, this feature can be disabled to display the code without color. For more information, see ["Disable syntax highlighter" on page 434](#).

This section outlines a list of all the supported extensions for syntax highlighting in the Review page.

File type	File extension
actionscript for flash	fla, as
batch	bat
c	c, h
c++	cc, cpp, hh, hpp, cxx

File type	File extension
csharp	cs, dotnet
docker	dockerfile
fortran	f, f90, for
golang	go
haxe	hx
java	java
javascript	js, javascript
jsx	jsx
lua	lua
markup	xml, xaml, htm, html, phtml, md, markdown
matlab	mat
perl	pl
php	php
powershell	ps1, psm1, ps, psm
python	py

File type	File extension
ruby	rb
rust	rust, rs, rlib
shell	sh
sql	sql
tcl	tcl
typescript	ts
unreadscript	uc
vhd1	vhd1
visualBasic	vb
yaml	yaml, yml

Commits

Whenever a new version of a file is checked into the P4 Server, a commit record is created. Begin browsing the history of commits by clicking **Commits** in the menu.

When you are viewing a particular file or directory, clicking the **Commits** tab displays the commits for that location in the depot.

HelixSwarm

Reviews
Projects
Files
Commits
Jobs
Groups
Workflows
Tests

Commits

// projects / mercury / dev

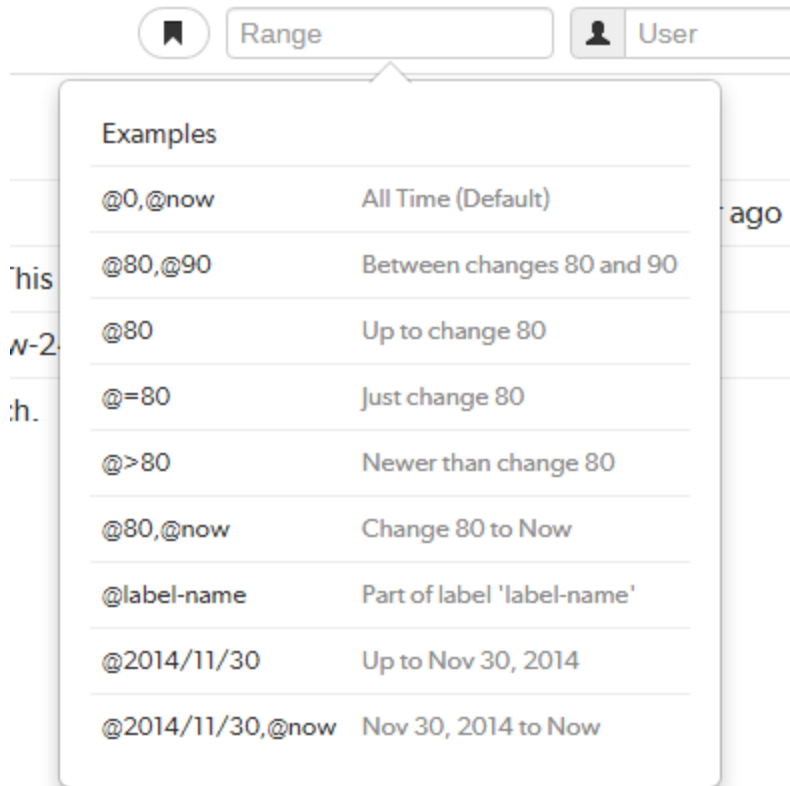
Browse Commits

Range User

Change	User	Description	Committed
276	Paula Boyle	Added endpoint for initiating server shutdown. #review-277	2 years ago
263	Steve Russell	Add functionality to report on the current status of the server. This is so the client ca ...	2 years ago
239	Steve Russell	Adding in basic client api for getting client information. #review-240	3 years ago

Range filter

The **Range** field lets you filter the list of changes for the depot path being viewed. When you click the **Range** field, a dropdown syntax guide appears providing sample commit filtering expressions.



The expressions that can be used within the **Range** field include:

- @0,@now (the default): displays all commits for the current depot path.
- @80,@90: displays all changes between 80 and 90.
- @80: displays all changes up to change 80.
- @=80: displays only change 80. Change 80 might not involve the current depot path, so there may be no commits to display.
- @>80: displays all changes newer than change 80.
- @80,@now: displays all changes from change 80 to now.
- @label-name: any changes represented by label *label-name*.
- @2014/11/30: displays all changes up to November 30, 2014.
- @2014/11/30,@now: displays all changes from November 30, 2014 to now.

File Commits

A file's commit history presents each version of a file that the P4 Server knows about, including the change number, userid, change description, time ago, along with **Open** and **Download** buttons.









// projects / mercury / candidate / README.md #1

View


Commits

Open

Download (57 B)

#	Change	User	Description	Committed	
#1	238	steve.russell	Branched the main line to both a new candidate and dev branch.	5 months ago	 
▼ //projects/mercury/main/README.md					
#1	5	jack.boone	Moved project to projects depot.	7 months ago	 
▼ //depot/projects/mercury/main/README.md					
#2	3	jack.boone	Updated format of the project README.	7 months ago	 
#1	2	jack.boone	Added README to the Mercury project main branch.	7 months ago	 

P4 Code Review also displays *contributing commits* when available, such as when a file has been renamed, or integrated from another location in the P4 Server.

If a commit represents a deleted revision, the **Open** and **Download** links are replaced with a trashcan icon  to indicate that this version is no longer available.

Remote depot commits

When your P4 Server has a remote depot configured, you can browse the contents of the remote depot, but remote depots do not share their commit history. If you attempt to view the commits of a file provided by a remote depot, P4 Code Review displays:

// builds

Browse Commits Range User

Remote depot (change details are not available).

Jobs

Jobs are a component of P4 Server's defect tracking system and a record of bugs found or improvement requests. Jobs can be associated with [changelists](#) to create *fix* records, indicating the work that solved the problem or provided the requested feature. Begin browsing jobs by clicking **Jobs** in the menu.

You can search available jobs by entering a job filter in the search box. Words, phrases, and `field=value` pairs can be entered. For example, entering `reportedby=steve.russell swarm` displays jobs that user *Steve Russell* has reported that also contain the word *swarm* in the description.

HelixSwarm Jobs

reportedby=steve.russell swarm Search

Job	Status	Reported By	Description	Modified Date
job093521	Inprogress	steve.russell	Remove the bash and vbs versions of the swarm-trigger. With this change we need to be careful not t...	about 3 hours ago
job093520	Open	steve.russell	The post format dropdown on project settings can appear in the wrong spot (to the right of post bod...	about 4 hours ago

The fields you can search for depend on the jobspec defined in your P4 Server.

Edit the Job columns


Configure the columns that are displayed:



1. Click the **Select Columns** button beside the search field to display the dropdown menu. The dropdown menu shows all of the available jobspec fields.
2. Select the columns to display using the checkboxes in the dropdown menu.
3. Drag the column label up or down the list to change the order the columns are displayed in.
4. Click the **Select Columns** button again, or a blank portion of the page, to hide the menu.

Change the column order from the **Jobs** page:

1. Click a column heading and drag it to the left or right to move the column to a new position.
2. The column display updates as columns are dragged.

Q reportedby=steve.russell swarm Search 

Job	Status	Reported By	Owned By	Description	Modified Date
job093521	Inprogress	steve.russell	jack.boone	Remove the bash and vbs versions of the swarm-trigger. With this change we need to be careful not t...	about 4 hours ago
job093520	Open	steve.russell	jack.boone	The post format dropdown on project settings can appear in the wrong spot (to the right of post bod...	about 4 hours ago

For more information on customizing job specifications, see [p4 jobspec](#) in [P4 Server Administration Documentation](#).

Job display

Jobs are typically identified with the word job followed by six digits. For example, `job000123`.

View a specific job by clicking on a linked job identifier, or by visiting the URL:

<https://myswarm.url/jobs/jobid>.

When P4 Code Review displays a job, the presentation is similar to:

job0000001



jack.boone created this job, last modified 7 months ago Closed

The details about edits to reviewers are hard to differentiate from the description of the review (in email notifications).

Notifications such as 'Added alex.randolph as a required reviewer.' should be made to stand out more prominently.

👤 245 Adding comment

✍️ 247 Fixing default reviewers code.

Details

Comments 0

Status Closed

User jack.boone

Date 7 months ago

The upper portion of the job presentation includes:

- The avatar and userid of the user that created the job
- The job's [creation time](#)
- If changes have been made to the job, the modifying userid and [time](#)
- A status indicator
- The job's description
- If any changelists have been submitted that *fix* the job, a list of those changelists and their descriptions. Each associated changelist includes an icon to represent their type: 🧑 (review), 🖋️ (pending), 📦 (committed)

The lower portion of the job presentation lists all of the keys configured in your P4 Server's jobspec. P4 Code Review inspects the jobspec and enhances the presentation of fields it recognizes. For example, date fields display as *time ago*, and links are created for *userids*.

Click the ["Comments" on page 328](#) tab to view any comments added to the job, or to add a comment.

Details

Comments 1



steve.russell Verified on version 2014.2/841040

Reply · Edit · ❤️

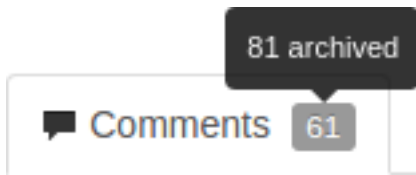


Add a comment

Emoji codes are supported

Post

Adding a comment sends a [notification](#). The **Comments** tab displays the number of open comments associated with the job. If you hover your mouse over the comment count, a tooltip is displayed showing how many comments are archived:



For more information on customizing job specifications, see [p4 jobspec](#) in [P4 Server Administration Documentation](#)..


Note:

The default P4 Server job specification contains very few fields. Adding fields to record additional information, such as the modification time and userid, reporting time and userid, can assist P4 Code Review use appropriate terminology when describing the current disposition of a job.

Add jobs

P4 Code Review does not provide the ability to create new jobs in the P4 Server, but jobs can be added to changelists or reviews.

Add a job to a changelist

1. Navigate to a changelist.
2. Click the **Add Job** link  [Add Job](#).
3. Scroll through the available jobs, or enter job search criteria to search available jobs.
For more information on job search criteria, see [Jobs](#) in [P4 CLI Documentation](#).
4. If you find the job you want to add, click its row to highlight it and then click **Select**. Or, double-click the desired job to add it.
5. If you do not find the appropriate job, click **Cancel**.

Add a job to a review

1. Click **Jobs (n)** (where **n** is the number of jobs that are already part of the review).
2. Click **Add a job**.
3. Scroll through the available jobs, or enter job search criteria to search available jobs.
For more information on job search criteria, see [Jobs](#) in [P4 CLI Documentation](#).
4. If you find the job you want to add, click it to highlight it and then click **Select** to add it.

Note:

If you attempt to add a job to a review that affects a single project, P4 Code Review applies the project's job view filter to display only jobs that affect the project. It is not currently possible to expand the filter to include jobs outside of the project.

Unlinking jobs

P4 Code Review does not provide the ability to delete jobs from the P4 Server, but jobs can be unlinked from changelists or reviews.

Unlink a job from a changelist

1. Navigate to a changelist that has an associated job.
2. Click the **X** button beside the job.
3. When prompted for confirmation, click **Unlink** to unlink the job.

Unlink a job from a review

1. Click **Jobs (n)** (where **n** is the number of jobs that are already part of the review).
2. Click **X** next to the job you want to unlink.

The job is immediately unlinked from the review.

Changelists

Changelists are the basic unit of versioning work in P4 Server. A changelist is a list of files, their revision numbers, and the changes made to those files. Commits is a shorter synonym used throughout P4 Code Review.

More information is available here:

- [Browsing changelists](#)
- [Browsing a user's shelved changelists](#)
- [P4 Code Review's internal use of changelists to facilitate code reviews](#)

Changelist display

View a specific *changelist* by clicking on a linked changelist number, or by visiting the URL: <https://myswarm.url/changes/changelist number>.

When P4 Code Review displays a change, the presentation is similar to:

Change 238



steve.russell committed this change 21 days ago into [//projects/mercury](#)
[Request Review](#)

Branched the main line to both a new candidate and dev branch.

[Add a Comment](#)

[Add Job](#)

Files **8**

Comments **0**

0 edited
 8 added
 0 deleted

[Icons: Comment, List, Filter, Search, etc.]

> + candidate/README.md#1	[Icon]
> + candidate/src/api/jamfile#1	[Icon]
> + candidate/src/api/jamfile.api#1	[Icon]
> + candidate/src/api/p4api.cc#1	[Icon]
> + dev/README.md#1	[Icon]
> + dev/src/api/jamfile#1	[Icon]
> + dev/src/api/jamfile.api#1	[Icon]
> + dev/src/api/p4api.cc#1	[Icon]

Tip: Use **[n]** and **[p]** to cycle through the changes.

P4 Code Review supports stream specs in your workspace using the [Private editing of streams](#) feature. If a changelist or review contains a stream spec, it will be displayed first in **Files** with the prefix **stream: //**, for example: `stream://MyStreamDepotName/MyStreamSpecLocationName`. A changelist/review can only contain one stream spec.

The changelist display includes:

- The avatar and userid of the user who made the change
- The time the change was made
- The common depot location containing all the files included in the change
- The **Download .zip** button, used to download a ZIP archive that contains all of the files in the changelist:

Note:

The **Download zip** option is not displayed if the zip command-line tool is not installed on the P4 Code Review server. For information about installing, and configuring the zip command-line tool, see [Zip archive](#).

When you select the **Download zip** option, P4 Code Review performs the following steps:

1. Scans the files/folders:
 - Checks that you have permission to access their contents, according to the P4 Server protections.
 - Checks that the total file size is small enough to be processed by P4 Code Review.

2. Syncs the file contents to the P4 Code Review server from the P4 Server.
3. Creates the ZIP archive by compressing the file content.
4. Starts a download of the generated ZIP archive.

Note:

- You might not see all of the above steps; P4 Code Review caches the resulting ZIP archives so that repeated requests to the same files/folders can skip the sync and compress steps whenever possible.
- If an error occurs while scanning, syncing, or compressing, P4 Code Review indicates the error.

- If the change was involved in a code review, a link to the review along with a **View Review** button.
- The description of the change
- A list of jobs that this change *fixes*, if any. You can [add jobs](#) and [unlink jobs](#) here.
- The list of files included in the change, including any folders between the common depot location and the file, and the file's version number.
- A tab to review any comments made regarding the change, or any of its files.

Each file is presented in a diff display, showing you whether the file was added, modified, or deleted. For text-based and image files, P4 Code Review can display any changes made within the file. For changes with only a single file, the diff display is the default; otherwise each file is listed. Click the filename to see the diff display. See ["Diffs" on the facing page](#) for more information.


The **Request Review** button indicates the current state of this change; no Review record has been created. Clicking **Request Review** starts a code review for this change. For more information, see ["Start a review" on page 443](#).

Important:

If your P4 Server is configured as a commit-edge deployment, and your normal connection is to an edge server, P4 Code Review refuses to start reviews for shelved changes that have not been promoted to the commit server.

Within P4 Code Review, this means that the **Request Review** button does not appear for unpromoted shelved changes. Outside of P4 Code Review, attempts to start reviews for unpromoted shelved changelists appear to do nothing. Ask your P4 Server administrator for assistance if you cannot start a review.

An administrator of the P4 Server can automatically promote shelved changes to the commit server by setting the configurable `dm.shelve.promote` to 1.

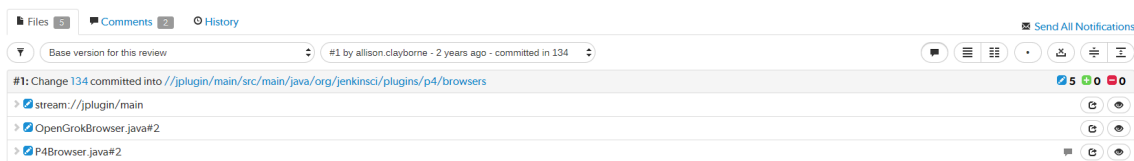
When a file in a changelist has one or more associated comments, an icon  appears near the far right of the file's entry.

Diffs

Tip: Code review page

The information on this page only applies to the classic view of the P4 Code Review Code review page. For information about diffs on the default review page, see ["Files tab" on page 430](#). To switch between the default review page and the classic review page, use the **Classic view** toggle switch at the top of the review page.

When you view a changelist or code review, the associated files are presented as *diffs*, short for differences, showing you how they have changed.



P4 Code Review supports stream specs in your workspace using the [Private editing of streams](#) feature. If a changelist or review contains a stream spec, it will be displayed first in **Files** with the prefix **stream: //**, for example: `stream://MyStreamDepotName/MyStreamSpecLocationName`. A changelist/review can only contain one stream spec.

Note:

If you append a changelist with a stream spec to a review that already contains a stream spec, the spec in the changelist replaces the original one in the review. If it is a different spec from the original spec in the review, P4 Code Review cannot display the diff between them and displays **File content unchanged**.

The review version selectors are used to specify which versions of a review you want to *diff*, they are located in the **Files** tab above the list of files.

- **Left dropdown selector:** the base version for the review is selected by default, base is the revision of the file that was checked out of the depot before it was changed for this review. The current review version in the depot, sometimes called Head can be selected, if there are multiple versions of the review a specific version can be selected.
- **Right dropdown selector:** the latest version of the review is selected by default. If there are multiple versions of the review, a specific version of the review can be selected.

Tip:

If a review consists of one or more P4 Code Review-managed changelists. When comparing versions of a review, P4 Code Review is showing any differences between the selected versions, not the review author's personal changelist. See ["Internal representation" on page 402](#) for details.

The first row of buttons above the files allow you to (left to right):






- **Show Comments** button: toggles the display of comments to inline in files or only in the **Comments** tab.
- **Show Diffs In-Line** button: displays all diffs as inline.
- **Show Diffs Side-by-Side** button: displays all diffs as side-by-side.
- **Toggle Show Whitespace** button: toggles the display of whitespace characters (such as space, tab, and newline) for all files.
- **Toggle Ignore Whitespace** button: toggles the highlighting of whitespace changes in all of the file diffs.
 - **Highlight whitespace changes:** makes it easier to identify changes in file types where whitespace is important. This is the default value.
 - **Ignore whitespace changes:** whitespace changes are not highlighted, this makes it easier to see the important changes in file types where whitespace changes are not important.
- **Collapse All** button: collapses all files
- **Expand All** button: expands all files. By default this button is disabled if there are more than 10 files in the review. For details, see [Expand All Limit](#).

Tip:

The default states for the **Show Comments**, **Show Diffs Side-by-Side**, **Toggle Show Whitespace**, and **Toggle Ignore Whitespace** buttons are set in your user settings, see [User Settings](#).

Each file is presented with an icon indicating whether the file was:

-  Added/Branched/Imported
-  Edited/Integrated
-  Deleted

The file's presentation can be controlled with (left to right):

Tip:

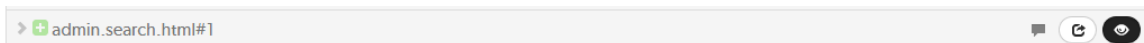
Only the **Show Full Context** button is available for stream specs.



- **Comments in File** icon (only displayed if the file contains comments): indicates that the file contains comments.
- **Show In-line** button: highlights line additions, modifications, and removals in a single pane.
- **Show Side-by-Side** button: highlights additions, modifications, and removals in two panes, with the older revisions of the file on the left, and the newer revision on the right.

- **Show Whitespace** button: makes whitespace characters more visible; spaces show up as dots, tabs show up as arrows that point to a bar, and line endings show up as down-pointing arrows.
- **Ignore Whitespace** button: toggles the highlighting of whitespace changes in a file, making it easier to identify non-formatting changes. Normally, whitespace is not ignored.
- **Toggle Ignore Whitespace** button: toggles the highlighting of whitespace changes in the file diff.
 - **Highlight whitespace changes:** makes it easier to identify changes in file types where whitespace is important. This is the default value.
 - **Ignore whitespace changes:** whitespace changes are not highlighted, this makes it easier to see the important changes in file types where whitespace changes are not important.
- **Show all diffs** button (edited and integrated files only): toggles between displaying all the diffs for the file and only the first few. The limit of what are shown by default is configurable by the administrator (see [Max Diffs](#)) and defaults to 1500. If there are fewer than that then this button is hidden.
- **Show Full Context** button (edited and integrated files only): toggles between displaying only the portions of the file that have changed and the full file.
- **Open File** button (edited or integrated files only): opens a new browser tab/window display the full file (where possible), provide access to its history, and a button to download the file.
- **Mark file as read** button (displayed for code reviews only): helps you (and others) keep track of which files have been reviewed. This is particularly useful when a code review consists of many files.

When clicked, the button color inverts and the associated file is visually muted, to make it easy to distinguish read files from unread files:



If a file has been marked as read, click the button a second time to reset the status to unread.

Viewing a diff

When you view a diff, the changes are highlighted:

- Red indicates lines that have been removed.
- Blue indicates lines that have been modified.
- Green indicates lines that have been added.

151	# Initialize variables	151	# Initialize variables
152	#	152	#
153	# "default =" - set only if unset	153	# "default =" - set only if unset
154		154	
155		155	
156	OSFULL = \$(OS)\$(OSPLAT)\$(OSVER) \$(OS)\$(OSPLAT) \$(OS)	156	if \$(NT) { OS = NT ; }
157		157	
158	#	158	OSFULL = \$(OS)\$(OSPLAT)\$(OSVER) \$(OS)\$(OSPLAT) \$(OS)
159	# OS specific variable settings	159	#
160	#	160	# OS specific variable settings
161		161	#
162	switch \$(OS)	162	
163	{	163	
164	case AIX : LINKLIBS default = -lbsd ;	164	switch \$(OS)
165	case DGUX : RANLIB default = "" ; RELOCATE =	165	{
166	case HPUX : RANLIB default = "" ;	166	case AIX : LINKLIBS default = -lbsd ;
167	INSTALL default = "" ;	167	case DGUX : RANLIB default = "" ; RELOCATE =
168	case IRIX : RANLIB default = "" ;	168	case IRIX : RANLIB default = "" ;
169	INSTALL default = "" ;	169	case HPUX : RANLIB default = "" ;
170	case MVS : RANLIB default = "" ; RELOCATE =	170	INSTALL default = "" ;
171	case NEXT : AR default = libtool -o ;	171	case MVS : RANLIB default = "" ; RELOCATE =
172	RANLIB default = "" ;	172	
173	case NCR : RANLIB default = "" ;	173	case NCR : RANLIB default = "" ;
174	INSTALL default = "" ;	174	INSTALL default = "" ;
175	case PTX : RANLIB default = "" ;	175	case PTX : RANLIB default = "" ;
176	case QNX : INSTALL default = "" ;	176	case QNX : INSTALL default = "" ;
177	case SCO : RANLIB default = "" ;	177	case SCO : RANLIB default = "" ;
210	CCFLAGS default = -nosyspath	209	CCFLAGS default = -nosyspath
211	C++ default = \$(CC) ;	210	C++ default = \$(CC) ;
212	C++FLAGS default = -nosyspath ;	211	C++FLAGS default = -nosyspath ;
213	FORTTRAN default = "" ;	212	FORTTRAN default = "" ;
214	LIBDIR default = /boot/devel	213	LIBDIR default = /boot/devel
215	LINK default = mwld ;	214	LINK default = \$(CC) ;
216	LINKFLAGS default = "" ;	215	LEX default = "" ;
217	LEX default = "" ;	216	MANDIR default = /boot/docum
218	MANDIR default = /boot/docum	217	NOARSCAN default = true ;
219	NOARSCAN default = true ;	218	RANLIB default = "" ;
220	RANLIB default = "" ;	219	STDHDRS default = /boot/deve
221	STDHDRS default = /boot/deve		
1884	actions quietly updated piecemeal together RmTe		
1885	{		
1886	{		
1887	\$(RM) \$(>)		
1888	}		
1889			
1890	actions Shell		
1891	{		
1892	copy \$(>) \$(<)		
1893	}		
1894	}		
1895		1769	
1896		1770	
1897	#	1771	#
1898	# Backwards compatibility with jam 1, where rules w	1772	# Backwards compatibility with jam 1, where rules w
1899	#	1773	#

Show more context buttons








The diff presentation displays a concise view of where changes were made within a file, showing the changed lines and only a few lines before and after the change. Sometimes, this concise view needs to be expanded to fully understand the context of the change, use the **Show More** and **Show All** buttons to display extra lines around the change:

Note:

By default, the number of extra lines displayed when you click the **Show More Lines Above** and **Show More Lines Below** buttons is 10. This setting can be changed by your P4 Code Review administrator, see ["More context lines" on page 680](#).

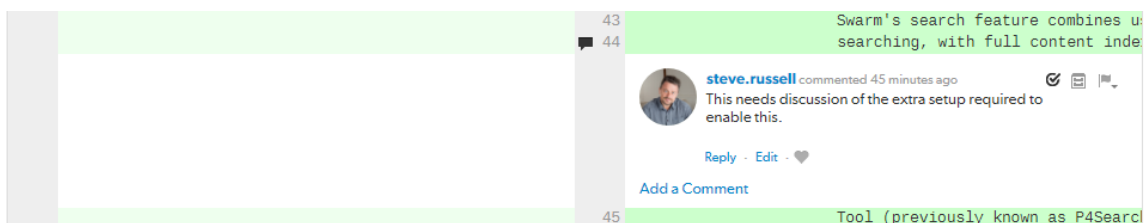
Tip:

The following buttons are not available for stream specs.


- **Show the file list panel**  button (only displayed on the vertical bar next to the file list panel): click to view the file list panel.
- **Show the information panel**  button (only displayed at the bottom of the information panel): click to view the detailed information panel.
- **Show All Lines to Start of File**  button (only displayed for the first change in the file): click to show all of the lines up to the start of the file.
- **Show More Lines for the Code Below**  button: click to show 10 more lines above the change, the extra lines are displayed in the pane below the button.
- **Show Entire Section**  button: click to show all of the lines between the changes that are above and below the button, the two changes and the lines between them are displayed in a single pane.
- **Show More Lines for the Code Above**  button: click to show 10 more lines below the change, the extra lines are displayed in the pane above the button.
- **Show All Lines to End of File**  button (only displayed for the last change in the file): click to show all of the lines down to the end of the file.

Comments

Comments can be displayed within the body of a file and appear immediately below the line the commenter targeted for comment. See ["Comments" on page 328](#) for more details.

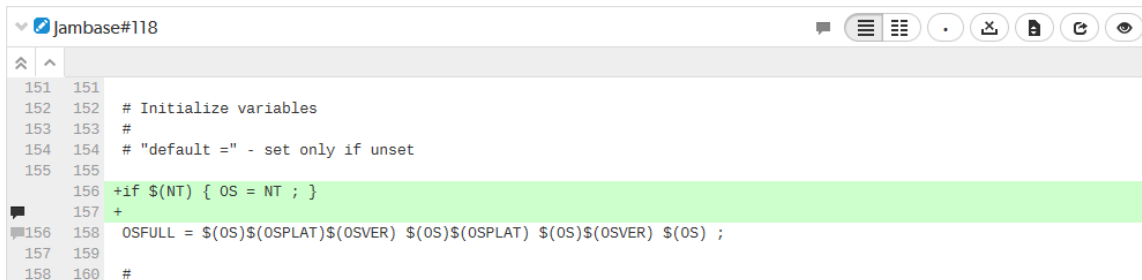


Note:

The *comment here* icon  appears in the line number column whenever there is a comment. Click the icon to display the comment and click the icon again to collapse the comment. This is useful when the comment display is toggled off.

Inline diff view

When viewing a diff in-line, the line numbers for the old version are first, the line numbers for the new version are second, and they are followed by the file content. Some users find it easier to use this view to locate an area that has changed, but they then switch to the side-by-side view to understand the change better. P4 Code Review maintains the scroll position, you do not lose your place in the file after toggling the diff view.



```

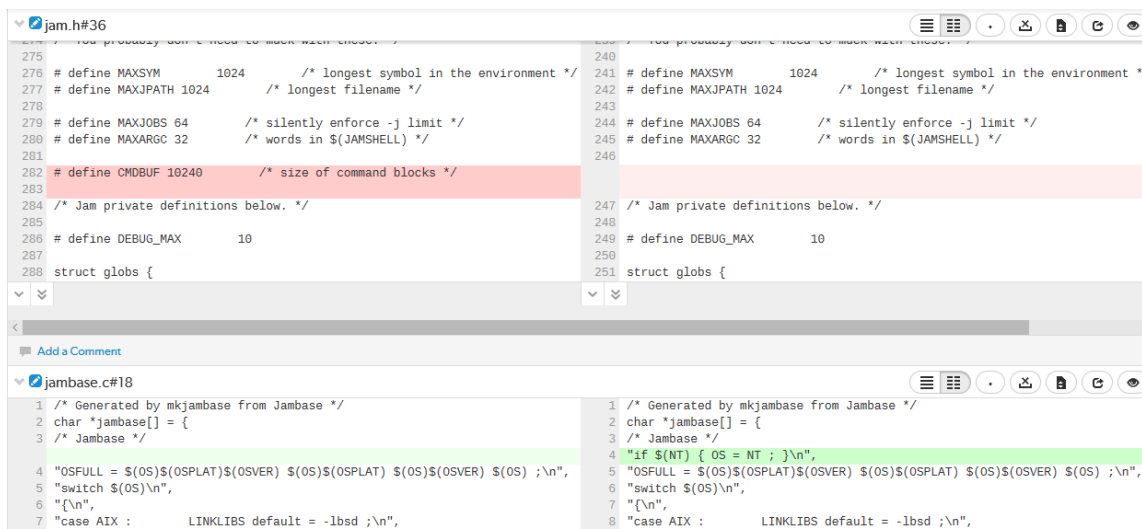
151 151
152 152 # Initialize variables
153 153 #
154 154 # "default =" - set only if unset
155 155
156 +if $(NT) { OS = NT ; }
157 +
158 OSFULL = $(OS)$(OSPLAT)$(OSVER) $(OS)$(OSPLAT) $(OS)$(OSVER) $(OS) ;
159
160 #
  
```

Note:

Press **n** on your keyboard to scroll to the next changed area within a file. Press **p** to scroll to the previous change.

Per-file toolbar

When a diff contains multiple files, the changes can be taller than your browser window. P4 Code Review keeps the *per-file toolbar* in view for each file as you scroll, this means that you can identify the file and control its presentation with the buttons on the right of the toolbar.



```

275 275
276 # define MAXSYM 1024 /* longest symbol in the environment */
277 # define MAXJPATH 1024 /* longest filename */
278
279 # define MAXJOBS 64 /* silently enforce -j limit */
280 # define MAXARGC 32 /* words in $(JAMSHELL) */
281
282 # define CMBUF 10240 /* size of command blocks */
283
284 /* Jam private definitions below. */
285
286 # define DEBUG_MAX 10
287
288 struct globs {
  
```

Comments

Comments are the primary feedback mechanism provided by P4 Code Review. Review comments can be flagged as *tasks* for a lightweight workflow within a review that helps authors and reviewers prioritize review feedback, see ["Tasks" on page 332](#) for details. By default, comment notifications are delayed to allow you to add or edit comments as you progress through a review without sending a notification for each individual comment on the review. Comment notifications are rolled up into a single notification that you can either leave to be sent automatically, or you can send manually, see ["Comment notification delay" on page 336](#).

You can also like comments, add links, add attachments, and add Emojis, see ["Comment features" on page 336](#) for details.

Tip:

- Markdown content is displayed in review comments, but Markdown support is limited to prevent execution of raw HTML and JavaScript content. For information about Markdown, see ["Markdown in comments and review descriptions" on page 397](#)
- If you use Markdown styles in your review comment, P4 Code Review renders them when you post the comment.

You can add comments to:

- [A changelist, review, or job](#)
- [A review description](#)
- [A changelist description](#)
- [A line of a text file in a changelist, or review](#)
- [A file in a changelist, or review](#)

Access comments for a changelist, review, or job, by clicking the **Comments** tab. The number of open (non-archived comments) is displayed in the tab. Hover your mouse pointer over the comment count, the tooltip shows how many comments are archived. See ["Archiving comments" on page 343](#) for details.

Adding comments

This section describes how to add comments.

Related comment features:

- [Comment notification delay](#)
- [Reply to comment](#)
- [Emoji](#)
- [Links](#)
- [Attachments](#)
- [Markdown](#)

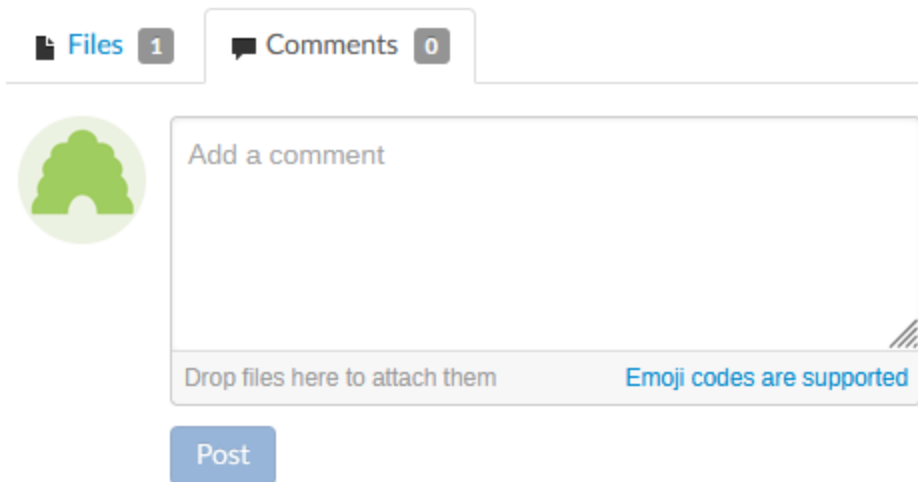
Tip:

You can use "[Links in descriptions and comments](#)" on [page 387](#) in comments. An @mention includes the specified user in the review, and they will receive a notification whenever there is an update to the review.

Commenting on a changelist, review or job

1. From the changelist, review, or job page: click **Comments** to view the **Comments** tab.
2. Add your comment in the text area.
3. Click **Post**.

Tip: Use the keyboard shortcut **Ctrl + Enter** to submit the comment or form.



The screenshot shows a user interface for adding comments. At the top, there are two tabs: 'Files' with a count of 1 and 'Comments' with a count of 0. Below the tabs is a circular profile picture of a green tree. To the right of the profile picture is a large text area with the placeholder text 'Add a comment'. Below the text area is a horizontal bar with the text 'Drop files here to attach them' on the left and 'Emoji codes are supported' on the right. At the bottom of the interface is a blue button labeled 'Post'.

Commenting on a review description

1. From the changelist or review page description area: click **Comments (n)** (where **n** is the number of comments that already exist).
2. Click **Add a comment**.
3. Add your comment in the text area.
4. Click **Post**.

Tip:

To hide the description comments, click **Comments n** (where **n** is the number of description comments that exist). To display the comments again, click **Comments n**.

Branched the main line to both a new candidate and dev branch. [Edit](#)

[Comments \(0\)](#) [Jobs \(0\)](#)

There are no review description comments yet.

All are failing to build on the different branches

Drag and drop files to attach them or [Choose your files](#)

Post
Cancel
☒ Flag as Task
[Post and notify \(0\)](#)

Commenting on a changelist description

1. From the changelist or review page description area: click **Add a Comment** or **n Comments** (where **n** is the number of comments that already exist).
2. Add your comment in the text area.
3. Click **Post**.

Change 704



Steve Russell committed this change 17 days ago into `//depot/tests/files` under Review 702

Changes to CI/CD Script for testing purposes

[1 comment](#)

[+ Add Job](#)

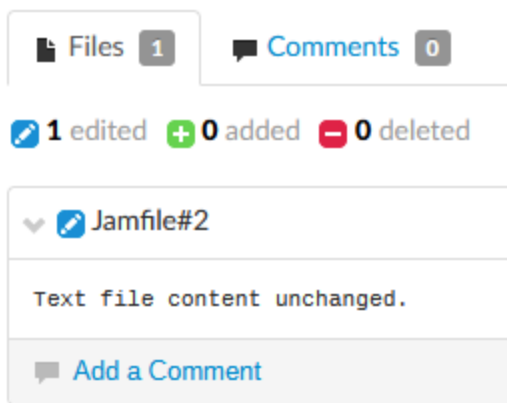
Commenting on a specific line in a file in a changelist or review

1. From the changelist or review page: click **Files** to view the **Files** tab.
2. Click on the line you want to comment on.
3. Add your comment in the text area.
4. **Optional** (Review page only): Select the **Flag as Task** checkbox to mark the comment as a task that needs to be addressed.
5. Click **Post**.

Commenting on a file in a changelist or review

1. From the changelist or review page: click **Files** to view the **Files** tab.
2. If there are multiple files, click the file you want to comment on to expand its view.
3. Click the **Add a comment** link in the footer of the file display.
4. Add your comment in the text area.
5. **Optional** (Review page only): Select the **Flag as Task** checkbox to mark the comment as a task that needs to be addressed.
6. Click **Post**.

To close an empty comment text box, click outside the comment text box or select **Esc** on your keyboard.



Editing comments


This section describes how to edit comments.

Related comment features:

- [Comment notification delay](#)
- [Reply to comment](#)
- [Emoji](#)
- [Links](#)
- [Attachments](#)
- [Markdown](#)

Note:

- You can only edit comments that you have created.
- If a comment is edited, all of the likes for that comment are removed.

1. Click the comment **Edit comment**  link.
2. Edit the comment text.

3. Add attachments or remove attachments. For instructions on adding and removing attachments, see ["Comment attachments" on page 339](#).

Tip:

If you remove the wrong file attachment but have not saved the comment yet, click **Cancel**, and the file will remain attached to the comment.

4. Click **Save** to save the edited comment.

P4 Code Review sends a notification to everyone involved in the review, including the review author and the reviewers, but not the editor of the comment.

The comment timestamp is marked as (*edited*) to show that the comment has been changed.

To close an empty comment text box, click outside the comment text box or select **Esc** on your keyboard.

Note:

P4 Code Review does not provide a mechanism to see older versions of edited comments.

Tasks


Flagging review comments as tasks is a lightweight workflow within a review that helps authors and reviewers prioritize review feedback. Any comment on a code review can be flagged as a task, indicating to the code review's author that the described issue needs to be addressed, and that the review is unlikely to be approved without a fix.

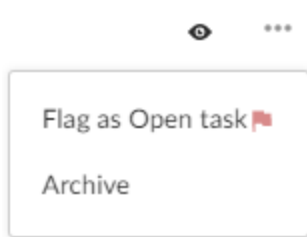
Note:

Changelist and Job comments cannot be flagged as tasks.

Flag a comment as a task

To flag a comment as a task, select the **Flag as Task** checkbox when posting a comment, or click the

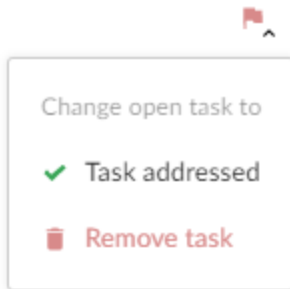
Comment actions  button in the upper right of an existing comment and select **Flag as Open task** in the drop-down menu.

**Note:**

If you do not have permission to archive comments, you do not have permission to flag comments as tasks. Anonymous users never have permission to archive comments, and can only view current task states.

Set a task to Task addressed or Not a task

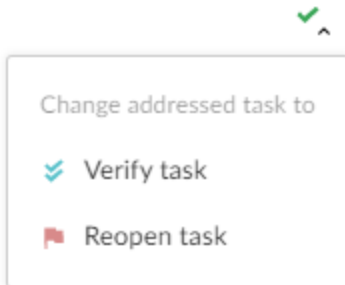
Once a comment is flagged as a task, it is considered to be an *open task*. Click the **Red flag** button to display a drop-down menu with the following options:



- **Task addressed:** usually used by the author of the review to indicate that the issue has been fixed.
- **Remove task:** used to correct comments that have been flagged as tasks by mistake.

Verify a task, verify and archive a task, or reopen a task

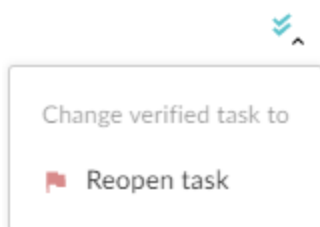
A comment with a green check indicates that the task has been addressed. Click the **Green check mark** button to display a drop-down menu with the following options:



- **Verify task:** usually used by the author of the comment, or another reviewer, after confirming that the issue is fixed.
- **Verify and Archive** (only supported in the classic review page): used to both indicate that the issue has been fixed, and to archive the comment so that it is hidden from view. Archived tasks, whether they are open, addressed, or verified, are not included in the task counts for the code review.
- **Reopen task:** used if the issue needs further work after it has been marked as addressed.

Reopen a task

A comment with a blue double-check indicates that the task has been verified. Click the **Blue double-check mark** button to display a drop-down menu with the following option:



- **Reopen task:** used if the issue needs further work post-verification, or if verification was made by mistake.

Task details

A summary of the number and status of comments flagged as tasks is displayed in the **Information panel** to the right of the review.

Note:

Archived comments that are flagged as tasks are not included in the summary or the **Tasks** dialog.



- **Red Flag:** Displays the number of open tasks on the review.
- **Green check mark:** Displays the number of addressed tasks on the review.
- **Blue double-check mark:** Displays the number of addressed and verified tasks on the review.

Show Task details  : Click to display a dialog listing all of the tasks associated with the review:

✕

Tasks		
Reporter ▼		<div> <div>🚩</div> <div>✓</div> <div>✓✓</div> </div>
Status	Reporter	Description
Addressed ✓▼	Allison Clayborne	I have passed this to QA.
Addressed ✓▼	Allison Clayborne	We will need to update this for the next revision.
Open 🚩▼	Claire Brevia	This description needs more detail. I feel the lack of detail makes it very hard to review this work. When will you get some time to change this? ⋮
Verified ✓✓▼	Claire Brevia	This makes no sense, please update

Within the **Tasks** dialog, you can filter the tasks by the **Reporter** (the userid of the user who created the task), and/or by task state using the buttons at the top of the dialog:

- Click the **Red flag** button to display only open tasks (comments that need to be addressed).
- Click the **Green check mark** button to display only addressed tasks (comments that have been addressed).
- Click the **Blue double-check mark** button to display only verified tasks (comments that have been addressed and verified).

To change the state of a task from the **Tasks** dialog, click the task state dropdown for the task and select the new task state.

To view the full comment text for a task, click the ellipses to the right of the task.

Approve a review with open tasks

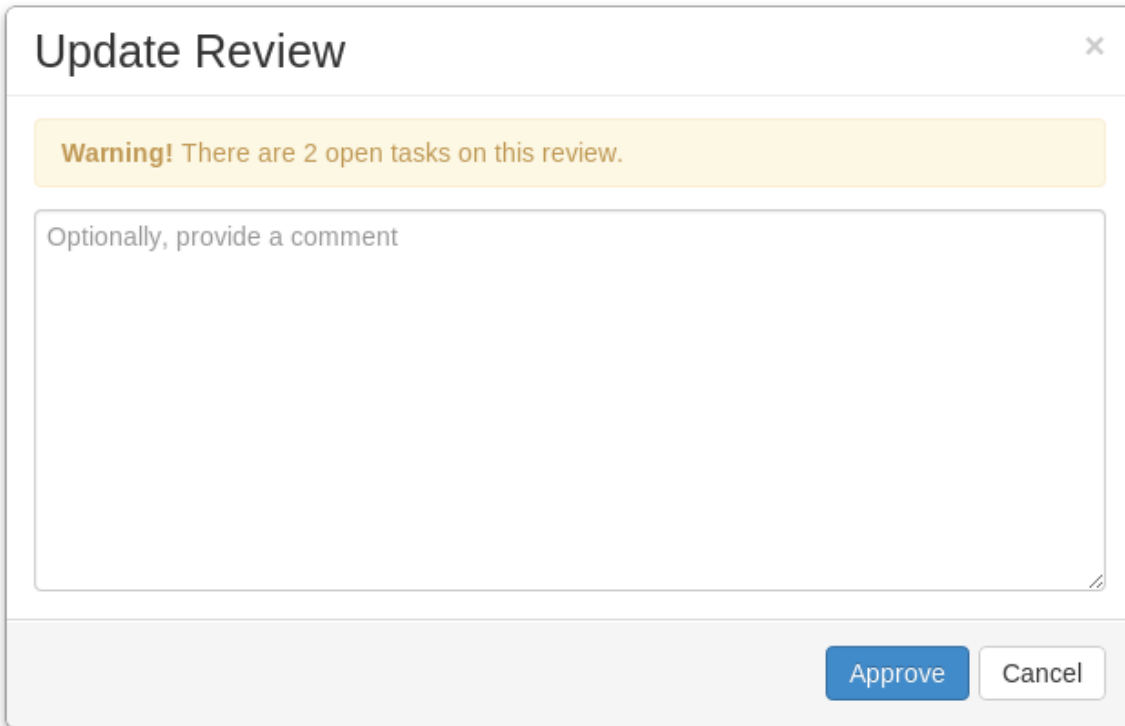
Flagging a comment as a task provides a visual indication that there is an identified issue that needs to be addressed before the review can be approved, see ["Set a task to Task addressed or Not a task" on page 333](#).

Note:

If P4 Code Review is configured to prevent approval of reviews with open tasks and a review has open tasks, the **Approve**, and **Approve and commit** options will not be available for the review. This option is configured by an administrator. See ["Disable approve for reviews with open tasks" on page 676](#).

To approve, or approve and commit a review with open tasks, you must address the tasks first and then set them to **Task Addressed**, or **Not a Task**. See ["Set a task to Task addressed or Not a task"](#) on page 333 for details.

If you select **Approve** or **Approve and Commit** for a review that has open tasks, a warning message is displayed in the **Update Review** dialog. The warning is only advisory, either click **Cancel** and address the open tasks or click **Approve** to approve the review. Archived open tasks will not trigger the warning message.

The image shows a dialog box titled "Update Review" with a close button (X) in the top right corner. Inside the dialog, there is a yellow warning banner that reads "Warning! There are 2 open tasks on this review." Below the banner is a large text area with the placeholder text "Optionally, provide a comment". At the bottom right of the dialog, there are two buttons: "Approve" (in blue) and "Cancel" (in white with a grey border).

Comment features

Comment notification delay

By default, comment notifications are delayed to allow reviewers to add or edit comments as they progress through a review without sending a notification for each individual comment on the review. Comment notifications are rolled up into a single notification and sent either manually by the reviewer, or automatically after the notification delay time has been exceeded.

The delay countdown is reset each time the reviewer adds or edits a comment on the review, by default the notification delay time is set to 30 minutes.

- If you are commenting on more than one review, each of the reviews that you are commenting on has its own notification delay countdown that only applies to the comments that *you* make on *that* review.

- If another reviewer is making comments on the same review as you, *that* reviewer has their own notification delay timer for *that* review.
- If you manually send a delayed comment notification, the notification will only contain the comments that *you* made on *that* review.

Tip:

The comment notification delay does not delay the posting of the comments, only the comment notification is delayed.

Note:

Comment notifications are only delayed for comments on reviews. Comments on commits or jobs produce notifications immediately.

Note:

The notification delay time is a global configuration setting configured by the P4 Code Review administrator, see [Comment notification delay](#).

Manually send the comment notification immediately:

1. Add or edit your comment as normal.
2. Click the **Post and notify (n)** button to the right of the comment box.
Where (n) is the number of delayed comment notifications in the queue waiting to be sent, this number does not include the current comment you are working on.

Tip:

- If you forget to click the **Post and notify (n)** button, click the **Send All Notifications (n)** button below the review description to send all of your notification for the review immediately.
- Only the comments that you have made on this review are rolled up into the notification that is sent when you click **Post and notify (n)** or **Send All Notifications (n)**.

Reply to comments

By default, you can reply to comments, replies are displayed in a thread below the parent comment. Comment thread depth is set to 4 by default, this means you can have up to 4 levels of replies for a parent comment. The **Reply** link is not displayed for replies at or above the maximum thread depth set for P4 Code Review. If the parent comment is archived, replies are archived with the parent, see ["Archiving comments" on page 343](#).


Note:

Comment replies can be disabled, and the thread depth can be increased or reduced by a P4 Code Review administrator, see ["Comment threading" on page 581](#).

Tip:


If the thread depth is reduced by a P4 Code Review administrator, earlier replies at a deeper level will continue to be displayed but you cannot reply to them.


To reply to a comment:

1. Click **Reply**  below the comment you are replying to.
2. Add your comment in the text area.
3. Click **Post**.

Files **6**
Comments **4**
Activity
Send All Notifications (0)

5 archived comments



Steve Russell commented on review 644 (version 1) //projects/mercury/dev/src/api/db.c, line 51 - 11 months ago
✓ read ✓


Allison Clayborne commented on review 644 (version 1) //projects/mercury/dev/src/api/db.c, line 105 - 10 months ago
👁️ ⋮

```
+ */
+int TalisAliquam(long quandoScelerisqueSem, float count, double litora) {
+ // Peristo velit parco.
+ parco = volutpat / 1;
+ printf("Aquam arcu vehicula faucibus umbra magna.");
```


@steve.russell what do you think of this?

👍


Steve Russell commented 4 months ago
👁️ ⋮

This looks good to me. I am happy for it to be checked in.

👍


Allison Clayborne commented 4 months ago
👁️ ⋮

Thanks for the confirmation.

👍

Add a comment

Drag and drop files to attach them or [Choose your files](#)

Post
☐ Flag as Task
☒ Post and notify (0)

Emoji

P4 Code Review comments support Emoji shorthand. So when you save a comment, emoticon text like `:smile:` is displayed as: 😊

Emoji emoticons are listed in the [Emoji Cheat Sheet](#).

Links in comments

Whenever you include a URL in a comment, it is automatically made into a link.

If the link points to an image, or a YouTube video, that resource is displayed at the end of the comment. For information about linking images and videos, see "[Common text styles](#)" on page 393.

338

Comment attachments


Note:

- P4 Code Review must be [configured to enable comment attachments](#). Once the configuration is complete, the comment area will include the following text **Drop files here to attach them**.
- By default, the maximum file size of a single comment attachment is limited to:
 - **Ubuntu:** 8Mb
 - **RHEL 8 and later:** 2Mb

Your P4 Code Review administrator can increase the maximum attachment size if required. See ["Increasing the maximum attachment file size" on page 577](#).

Files can be attached to comments. This is useful for sharing files in code reviews, such as screenshots of error conditions, reference code, and documents.

When a file is attached to a comment, you can:

- Hover over the attachment to see its filename and file size.
- Click the attachment to open the file in a new browser tab.
- Hover over the attachment and click the **Download**  button to download the file.

Adding attachments to a comment

Tip:


Multiple files can be attached to a comment, either one at a time or multiple files at the same time.

To attach a file to an open comment:

1. Do one of the following:
 - Drag the files from your file browser and drop them on the comment.
If you drag a folder onto a comment, all of the files in the folder and the files in any subfolders are added to the comment.
 - Click **Choose your files** and select the files from the browse dialog.
You cannot add folders to a comment using the file picker.
2. If you attach a file to the comment by mistake, you can remove it by clicking the **X** button on the attachment.
3. Click **Post**.

The file attachments are displayed below the comment text and image thumbnails are displayed for supported image formats.


Removing an attachment from a comment

1. Click the comment **Edit comment**  link.
2. Hover over the attachment you want to remove and click **X** to remove the file.
3. Confirm you want to remove the attachment when prompted.
4. Click **Save** to save the comment.

Tip:

If you remove the wrong file attachment but have not saved the comment yet, click **Cancel**, and the file will remain attached to the comment.

Reacting to comments

As an authenticated user, you can *like* a comment by clicking the thumbs up  icon beneath the comment.

When you like a comment, a [Notification](#) is sent to the author of the comment, and the thumbs up icon changes to yellow to indicate that you have liked the comment. The number of likes the comment has is displayed next to the thumbs up icon. If a comment is edited, all of the likes for that comment are removed.

Hover your mouse pointer over the number of likes to display the usernames of everyone that has liked the comment.

Click the thumbs up icon again to *unlike* a comment. You can like/unlike a comment that has been archived.

Comment context

When comments are added to files in a review, on lines that have been changed, P4 Code Review records several lines of context before the line receiving the comment. This helps makes sense of the comments should later changes remove those lines.

90

95

91

96

97

97

98

98

99

99

100

100

101

101

102

102

103

103

104

104


105

105

```

97 /**
98  * Herbam class tortor justo, augue justo congue. Volutpat a magnis
99  * cursus pupula. Aliquet diam peristo ullamcorper cultura. Suspendisse
100  * nostra.
101  */
102 int TalisAliquam(long quandoScelerisqueSem, float count, double litora) {
103     // Peristo velit parco.
104     parco = volutpat / 1;
105     printf("Aquam arcu vehicula faucibus umbra magna.");

```




Allison Clayborne commented on review 644 (version 1) - 10 months ago

👁️ ⋮

@steve.russell what do you think of this?

👍




Steve Russell commented 4 months ago

👁️ ⋮

This looks good to me. I am happy for it to be checked in.

👍



Allison Clayborne commented 4 months ago

👁️ ⋮

Thanks for the confirmation.

👍

Add a comment


Each comment associated with that line has a record of the context, but only the first comment displays that context.

Mark comments as read


When you mark a comment as read, the comment is rolled up into a single line to save space and make it easier for you to find comments you have not read. The read flag is remembered independently for each user.

Mark a single comment as read:

1. Click the **Mark comment as read**  button to the right of the comment.
2. The comment is rolled up into a single line to save space.



Vernon Renno commented on review 1805 (version 3) - about 3 hours ago

read 


Note:

Marking a parent comment as read will not mark the child comments as read.

Mark all of the comments on a review as read:

Tip:

Mark all comments read will mark all the comments as read including the description comments.

1. Click the **Review actions**  button.
2. Select **Mark all comments read** from the dropdown menu.
3. All of the comments in the review are rolled up into single lines to save space.

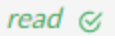
	Steve Russell commented on review 644 (version 1) //projects/mercury/dev/src/api/db.c, line 51 - 7 months ago	read 
	Allison Clayborne commented on review 644 (version 1) //projects/mercury/dev/src/api/db.c, line 105 - 7 months ago	read 
	Allison Clayborne commented on review 644 (version) - 10 days ago	read 
	Steve Russell commented on review 644 (version 2) //projects/mercury/dev/src/api/db.c, line 67 - 3 days ago (edited 3 days ago)	read 
	Jack Boone commented on review 644 (version 2) - about 15 hours ago	 read 

Mark comments as unread

Marking a comment as unread expands the comment so that you can view the comment content. If a comment is marked as read and the comment changes, P4 Code Review will automatically clear the read flag so that you can see that the comment has changed. Changes that automatically mark a comment as unread are:

- Comment text is edited
- Comment attachments are added or removed
- Comment is marked as a task
- Task is reopened
- Comment or task is unarchived

Mark a single comment as unread:

1. Click the  button to the right of the comment.
2. The comment content is expanded.

Note:

Marking a parent comment as unread will not mark the child comments as unread.

Mark all of the comments on a review as unread:

Tip:

Mark all comments unread will expand the content of all the comments including the description comments.

1. Click the **Review actions**  button.
2. Select **Mark all comments unread** from the dropdown menu.

3. The content of all of the comments in the review are expanded.

Archiving comments

Tip:

When you archive a comment, it is archived for all of the P4 Code Review users.

As a code review progresses, comments made on earlier versions of a file might become less useful. Archiving these comments tidies up the comment view and makes it easier to find the more important comments.

The **Archive** option is only available for top level comments, if a comment has replies they are archived with the parent comment.

To archive a comment:

1. Click the **Comment actions** *** button at the top right of the comment you want to archive.
2. Select **Archive** from the dropdown menu.

Archived comments are hidden from view, the number of archived comments that exist for the review is displayed in the **archived comments** button at the top of the **Comments** tab.

Files **1** Comments **7** Activity

2 archived comments

To display and hide archived comments:

Click on the **archived comments** button to toggle the archive comment display on and off.

Files **1** Comments **8** Activity

2 archived comments



Vernon Renno commented on review 1710 (version 1) - about a minute ago

Please run this past QA before release.



Vernon Renno commented on review 1710 (version 1) - less than a minute ago

Thanks



Restoring comments

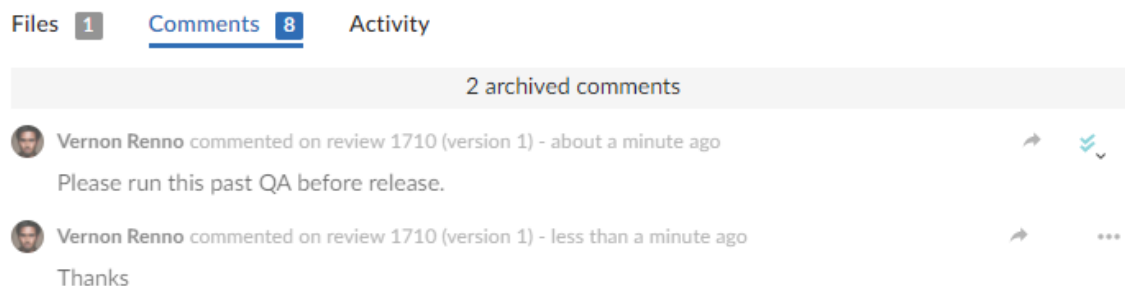
Tip:


When you restore a comment, it is restored for all of the P4 Code Review users.

Archived comments can be restored. If archived comments have replies, the replies are also restored.

To restore an archived comment:

1. Click the **archived comments** button at the top of the **Comments** tab to display all of the archived comments.



2. Find the comment you want to restore.
3. Click the **Restore** button  in the top right of the comment to restore it.

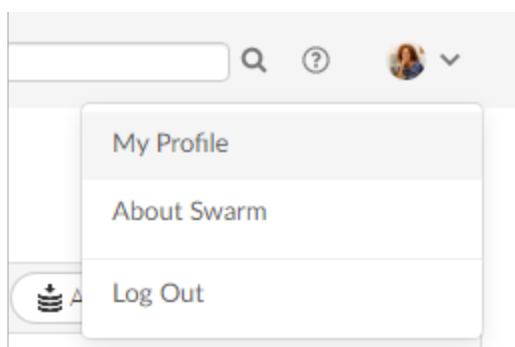
Users


P4 Code Review's users are based on the users configured in the P4 Server.

- ["Viewing your user profile" below](#)
- ["Viewing another user's profile" on page 350](#)
- ["Viewing another user's profile when you have admin or super user privileges" on page 351](#)

Viewing your user profile


Display your own user profile when you are logged in by clicking on your userid in the header and selecting **My Profile**.



The user profile page displays your [avatar](#), your full name, your email address, an **Unfollow all projects and users for your-username** button, a list of users following your activity, the users or [projects](#) that you are following, and the list of projects of which you are an owner or member, including an eye icon () beside any private projects you belong to.

Users /Allison.Clayborne

allison.clayborne



2 FOLLOWERS
1 FOLLOWING
6 PROJECTS

FULL NAME
Allison Clayborne
EMAIL ADDRESS
allison.clayborne@nowhere.xxjzzj.com
Unfollow all Projects and Users for allison.clayborne

FOLLOWERS
FOLLOWING
PROJECTS

- Blue Book
Owner, Member
- JPlugin
Owner, Member
- Maker
Owner, Member
- Test Data
Member
- Test project for workflow
Member
- Mercury
Following

Activity Shelves Settings Notifications Reviews Commits Comments Jobs

Allison Clayborne updated project (JPlugin)
A Java plugin for continuous integration. 4 days ago

Allison Clayborne updated project (JPlugin)
A Java plugin for continuous integration. 4 days ago

Allison Clayborne updated project (JPlugin)
A Java plugin for continuous integration. 4 days ago

Allison Clayborne commented on review 264 (revision 1) (client.cc, line 37) for mercury:dev
The strings should be declared as constants. 25 days ago
3 comments

Allison Clayborne rejected review 232 (revision 1) for test-data:data
Updating tests for test-18.txt, about a month ago
Add a comment

Allison Clayborne approved review 230 (revision 1) for test-data:data
Updating tests for test-17.txt, about a month ago
Add a comment

Allison Clayborne approved review 215 (revision 1) for test-data:data
Updating tests for test-09.txt, about a month ago
Add a comment

Allison Clayborne approved review 199 (revision 1) for test-data:data
Updating tests for test-01.txt, about a month ago
Add a comment

Allison Clayborne approved review 197 (revision 1) for test-data:data
Updating tests for test-00.txt, about a month ago
Add a comment

Allison Clayborne approved review 194 (revision 1) for test-data:data
Updating test data to the latest format. about a month ago
Add a comment

Allison Clayborne (on behalf of steve.russell) committed change 309 into test-data:data
Updating test data to the latest format. about a month ago
Add a comment

Allison Clayborne requested review 307 (revision 1) for jplugin:main
Record and return the number of ticks per thread. about a month ago
Add a comment

Allison Clayborne requested review 303 (revision 1) for jplugin:main
The user property should be based on the email address of the user. about a month ago

Unfollow all projects and users for yourself

When you are viewing your own user profile, you can unfollow all projects and users that you are following.

Note:

This action cannot be undone.

1. Click the **Unfollow all projects and users for *your-username*** button.
2. Click **OK** when the confirmation dialog is displayed to complete the unfollow action.

3. You will no longer be following any projects or users.

Activity tab

Click the **Activity** tab to display the [activity stream](#) for events you have created.

Activity

Shelves

Settings


Notifications

Reviews

Commits

Comments


Jobs



Allison Clayborne

updated project (JPlugin)
A Java plugin for continuous integration.


4 days ago



Allison Clayborne

updated project (JPlugin)
A Java plugin for continuous integration.


4 days ago



Allison Clayborne

updated project (JPlugin)
A Java plugin for continuous integration.

4 days ago




Allison Clayborne

commented on [review 264 \(revision 1\)](#) ([client.cc](#), [line 37](#)) for [mercury:dev](#)
The strings should be declared as constants.

25 days ago

3 comments




Allison Clayborne

rejected [review 232 \(revision 1\)](#) for [test-data:data](#)
Updating tests for test-18.txt,

about a month ago

Add a comment



Allison Clayborne

approved [review 230 \(revision 1\)](#) for [test-data:data](#)
Updating tests for test-17.txt,

about a month ago

Add a comment

Shelves tab

Click the **Shelves** tab to display a list of your shelved changelists.

<div> Activity Shelves Settings Notifications </div>			
Change	Description	Created	
1380317	#review-1380318 Update the coverage of My Reviews to cover the new look/behaviour. @ph...	4 days ago	<div> <div>View Review</div> </div>
1379450	Update the phone numbers listed in Swarm's contact page to match those (a...	5 days ago	<div>Request Review</div>
1378653	#review-1378654 Remove language indicating that Swarm is at an early stage. @pmclary s...	7 days ago	<div> <div>View Review</div> </div>
1378432	#review-1378433 Add coverage of the new p4 -> auto_create_url configuration item, whic...	7 days ago	<div> <div>View Review</div> </div>
1371898	#review-1371899 Update the admin configuration to capture the change in project settings...	26 days ago	<div> <div>View Review</div> </div>
1371895	#review-1371896 Remove the note fro the JIRA integration page about manual configuration...	26 days ago	<div> <div>View Review</div> </div>
1371884	#review-1371886 Update delayed notifications description to make it clear that the checkb...	26 days ago	<div> <div>View Review</div> </div>
1357393	#review-1357327 My tweaks to Jenny's review.	2 months ago	<div> <div>View Review</div> </div>

A *shelved* changelist is a pending changelist that has a copy of one or more files from within the changelist stored on the server. Shelved files are not versioned. If you update the shelved files, the update replaces any existing files on the changelist's shelf.

Note:

P4 Code Review can use multiple shelved changes to record the history of reviews. See ["Internal representation" on page 402](#) for details.

P4 Code Review uses shelved changelists as the basis of its code review feature. However, not all shelved changelists are reviews. Users may shelf files for other reasons, including ensuring that the P4 Server has a copy of work in progress, or as a way to move temporary work from one workspace to another.

Click **Request Review** to start a review for any shelved changelist that is not already involved in a review.

Click **View Review** to view the review associated with shelved changelists when a review has already started.

Important:

By default, when you delete files from a shelved changelist, the files are not removed from the associated review.

However, P4 Code Review can be configured to remove files from a review when they are deleted from an associated shelf, see ["Process shelf file delete when" on page 680](#). To delete a shelved changelist without removing all of the shelved files from the associated review, see ["Deleting shelves" on page 447](#).

Do not use the P4 Code Review user that is configured in the [P4 Code Review configuration file](#) when deleting shelves, or deleting files from shelves. The P4 Code Review logic processes the *shelf-delete* trigger event, if the event is invoked by the P4 Code Review user it is rejected. The delete operations will fail.

Settings tab

The **Settings** tab allows you to configure how [diffs](#) are initially displayed when you view them, and how time is displayed by P4 Code Review.

Note:

Defaults for these options are configured by the P4 Code Review administrator, see [Users](#). The defaults set by the P4 Code Review administrator are used until you change your settings.

The screenshot shows the 'Settings' tab in the P4 Code Review interface. At the top, there are four tabs: 'Activity', 'Shelves', 'Settings' (which is active), and 'Notifications'. Below the tabs is a 'Profile Settings' section with a 'Reset to default' button. The settings are organized into two main sections: 'Reviews' and 'Time Display'. The 'Reviews' section has four toggleable options: 'Show comments in files' (checked), 'View diffs side-by-side' (checked), 'Show space and newline characters' (unchecked), and 'Ignore whitespace when calculating differences' (unchecked). The 'Time Display' section has one dropdown menu labeled 'Display the time in' with 'Timestamp' selected. At the bottom right of the settings panel are 'Save' and 'Cancel' buttons.

Toggle the **Reviews** settings to control how diffs are initially displayed on changelists and reviews by default.

Choose the **Time Display** settings to configure how time is displayed by P4 Code Review. You can configure P4 Code Review to display timestamps based on your local machine browser time, or by how long ago events happened.

Tip:

If timestamp is selected, the date format used by P4 Code Review matches the date format configuration of the local machine browser.

Click the **Save** button to save the settings. Click **Reset to default** to reset the settings back to system defaults.

Notifications tab

The **Notifications** tab allows you to configure which notifications you receive when events occur within P4 Code Review. This allows you to limit the number of emails you receive to just those you are interested in. The settings apply across all projects.

Note:

Defaults for these options are configured by the P4 Code Review administrator. Your P4 Code Review administrator may force some of these options to on or off, see "[Global settings](#)" on page 654.

Activity
Shelves
Settings
Notifications

Adjust when notifications are sent to you about reviews that you're associated with (as an author, reviewer, project member or moderator).

Email me when:
Reset to default

	Check all <input type="checkbox"/>
I change the state of a review	<input type="checkbox"/>
I am the author, and	
a review is requested	<input checked="" type="checkbox"/>
files in the review are updated	<input checked="" type="checkbox"/>
tests on the review have finished	<input checked="" type="checkbox"/>
a vote is cast on a review	<input checked="" type="checkbox"/>
the state of the review changes	<input checked="" type="checkbox"/>
a review or change is committed	<input checked="" type="checkbox"/>
a comment is made on the review or change	<input checked="" type="checkbox"/>
a comment on the review or change is updated	<input checked="" type="checkbox"/>
Someone likes one of my comments	<input checked="" type="checkbox"/>
I am a member, and	
a review is requested	<input checked="" type="checkbox"/>
a review or change is committed	<input checked="" type="checkbox"/>
I am a reviewer, and	
files in the review are updated	<input checked="" type="checkbox"/>
tests on the review have finished	<input checked="" type="checkbox"/>
a vote is cast on a review	<input checked="" type="checkbox"/>
the state of the review changes	<input checked="" type="checkbox"/>
someone joins or leaves the review	<input checked="" type="checkbox"/>
a review or change is committed	<input checked="" type="checkbox"/>
a comment is made on the review	<input checked="" type="checkbox"/>
a comment on the review is updated	<input checked="" type="checkbox"/>
I am a moderator, and	
files in the review are updated	<input checked="" type="checkbox"/>
tests on the review have finished	<input checked="" type="checkbox"/>
a review or change is committed	<input checked="" type="checkbox"/>

Save
Cancel

Toggle notifications for each event on or off to control whether you receive an email when that event occurs.

Click the **Save** button to save the settings. Click **Reset to default** to reset the settings back to system defaults.

Viewing another user's profile

View the profile pages of other users displayed anywhere in P4 Code Review by clicking on their [avatar](#), [userid](#), or by visiting the URL: <https://myswarm.url/users/userid>.

Note:

When viewing another user's profile page you can see the user's **Settings** tab but you cannot make changes to it.

Note:

Currently, P4 Code Review does not provide an overall list of users.

Follow or unfollow the user

- **If you are not following the user:** click the **Follow** button below the user's avatar to start following the user.
- **If you are following the user:** click the **Unfollow** button below the user's avatar to stop following the user.

The screenshot shows the profile page for a user named Alex Randolph. The page has a header with the username 'alex.randolph' and a search bar. Below the header, there's a navigation bar with tabs: Activity, Shelves, Settings, Notifications, Reviews, Commits, Comments, Jobs, and a RSS icon. The main content area is divided into two columns. The left column contains the user's profile information: a circular avatar, a 'Follow' button, and statistics showing 0 followers, 1 following, and 4 projects. Below this, there's a 'FULL NAME' section with 'Alex Randolph', an 'EMAIL ADDRESS' section with 'alex.randolph@nowhere.xjzzj.com', and a 'FOLLOWING' section with a small circular icon. The right column contains a list of activity items, each with a small circular icon, the user's name, a description of the activity, and a timestamp. The activity items include: 'Alex Randolph committed change 295 into mercury:main', 'Alex Randolph approved review 177 (revision 1)', 'Alex Randolph approved review 165 (revision 1)', 'Alex Randolph approved review 131 (revision 1)', 'Alex Randolph approved review 129 (revision 1)', 'Alex Randolph approved review 179 (revision 1)', 'Alex Randolph approved review 175 (revision 1)', 'Alex Randolph voted down review 119 (revision 1)', and 'Alex Randolph cleared an issue on review 135 (revision 1)'. Each activity item also has a comment count and a link to 'Add a comment'.

Viewing another user's profile when you have admin or super user privileges

View the profile pages of other users displayed anywhere in P4 Code Review by clicking on their [avatar](#), [userid](#), or by visiting the URL: <https://myswarm.url/users/userid>.

Unfollow all projects and users for another user

When a user has been removed from the P4 Server but they are still following projects and users, it is useful to be able to remove all of their follows. This helps to keep the project and user follower lists up to date.

Warning:

This action cannot be undone.

1. Click the **Unfollow all projects and users for *username*** button located below the user's email address.
2. Click **OK** when the confirmation dialog is displayed to complete the unfollow action.
3. The user will no longer be following any projects or users.

Groups

Groups are a feature of P4 Server that makes it easier to manage permissions for users.

It is important to be aware of certain aspects of P4 Server groups:

- Groups can have both users and sub-groups. A user that is a member of a sub-group is also a member of the parent group.
- Groups have owners, but owners are not members of their groups. A group owner can be added as a member of a group.
- Groups can be created by users with *super* privileges in P4 Server (**p4d**). If p4d is at version 2012.1 or newer, users with *admin* privileges can also add groups.
- Groups can be edited by their owners, or by users with *super* privileges in P4 Server.

- Groups have a separate namespace from users; you can have a user named `fish` that is a member of a group named `fish`.
- Groups can be project moderators
- Groups can be reviewers, see [Review display](#) and [@@mention](#) for details.
- You can [@@mention](#) a group in a code review comment to ensure the mentioned group receives [notifications](#) of code review events. This also creates a link that displays the group's details when clicked. See [@@mention](#) for details.
- Groups cannot be project owners.

More information on adding, editing, and removing groups is included in the ["Groups" on page 471](#) chapter.

Listing groups

Begin browsing groups by clicking **Groups** in the menu.

Name	Description	Owners	Members
Marketing marketing	Marketing team		3
triage	Triage team.		5
docs	Docs team		3
swarm-group	Swarm group		1
swarm-support	Swarm support team		3

For each group, the group's name, description, list of owner avatars, and a membership count (which has a darker background when you are a member). Click on a group name to display details for that group.

The groups listing is sorted by multiple criteria:

- Groups that you own, or belong to, are listed first. A thicker row border indicates where your group ownership/membership ends; you are not an owner/member of any group below the thicker border.
- Groups are then sorted by name.

You can search available groups by entering some text in the **Search** field and clicking **Search**. Any group names or descriptions that match the entered text are displayed.

Name	Description	Owners	Members
swarm-group	Swarm group		1
swarm-support	Swarm support team		3

If you have *super* privileges in P4 Server (P4D), or have admin privileges in P4D version 2012.1 or newer, P4 Code Review displays the **+ Add Group** button. Click **+ Add Group** to add a new group.

Note:

If your P4 Server has a large number of groups, the time required to display the list of groups might be notable.

Viewing a group

View a specific group by clicking on a linked group name, or by visiting the URL:

<https://myswarm.url/groups/group-id>.

Activity page

The group **Activity** page shows the **activity stream** for events generated group members to the right of the group sidebar.

The screenshot displays the 'docs' group's activity page. On the left is the group sidebar, and on the right is the activity stream.

Group Sidebar (docs):

- Avatar:** A purple circular logo with three stylized bees.
- Group Name:** Docs team
- Owner Count:** 1 OWNER
- Member Count:** 3 MEMBERS
- Email Address:** docs.team@nowhere.xxjzzj.com
- OWNERS:** One owner profile picture is shown.
- MEMBERS:** Three member profile pictures are shown.

Activity Stream:

- Activity Tab:** Selected, showing a list of events.
- Event Headers:** Reviews, Commits, Comments, Jobs, and a notification bell icon.
- Events:** A list of six activity entries, all from user 'paula.boyle' and dated '7 days ago'. Each entry includes a profile picture, a description of the action (e.g., 'updated files in review 249 (revision 8) for jplugin:main'), and an 'Add a comment' link.

The following information is displayed in the group sidebar:

- Group name
- Group avatar, if it has one
- Group description, if it has one
- Count of the number of owners and members of the group
- Group email address, if it has one

- Avatars of the group owners
- Avatars of the group members

Reviews page

The group **Reviews** page shows a list of code reviews authored by any of the group members. It lists code reviews that are in progress and code reviews that are complete.

ID	Description	Project	Created	Status
368	Updated definitions (again)	mercury:dev	5 months ago	6 1/0
330	Added a missing comment.	jplugin:main	7 months ago	0 0/0
328	Add comment to method.	jplugin:main	7 months ago	0 1/0
264	Add functionality to report on the current status of the server. This is so the client ca...	mercury:dev	2 years ago	3 1/0
119	Adding admin.projects to documentation. #review @@!docs		3 years ago	9 0/1
105	Optimised some of the error checking to improve performance. #review @@!triage	mercury:main	3 years ago	2 1/0
77	Adding admin.files to documentation. #review @@!docs		3 years ago	2 0/1

The following information is displayed in the group sidebar:








- Group name
- Group avatar, if it has one
- Group description, if it has one
- Count of the number of owners and members of the group
- Group email address, if it has one
- Avatars of the group owners
- Avatars of the group members

Reviews list

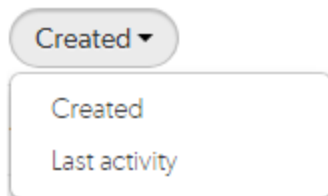
The **Opened** and **Closed** tabs display:

- **Opened tab:** displays a list of all code reviews that have started, are being reviewed, are awaiting revisions, or need to be committed.
- **Closed tab:** displays a list of all code reviews that have been approved and committed, rejected, or archived.

Each reviews list displays a summary for each review:

ID	Description	Project	Created						
368	 Updated definitions (again)	mercury:dev	5 months ago						

- The review id
- The avatar of the review author
- The review description
- The associated project name and branch
- **Created** or **Last activity** dropdown: click to change the order the reviews are displayed in, options are:
 - **Created**: Reviews sorted by when they were created
 - **Last activity**: Reviews sorted by when they were last updated



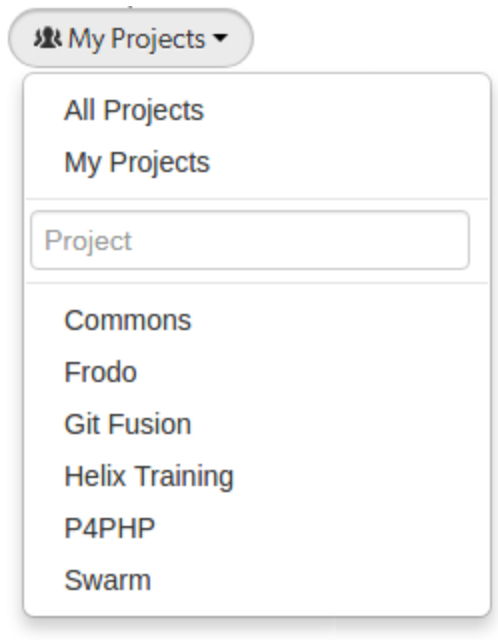
Note:

- If the **Result order** button is not displayed reviews are sorted by when they were created.
 - The **Result order** button display is a global setting controlled by the P4 Code Review administrator. See ["Reviews filter" on page 673](#) for details.
 - When review results are older than 24 hours they are displayed in numerical order within each day.
- An icon indicating the current review state
 - An icon indicating the review type. This can be Pre-commit or Post-commit.
 - An icon indicating the test suite state for the review, either tests in progress, tests passed, or tests failed.
 - A counter for the number of open (non-archived) comments that are associated with the review. Hover your mouse over the comment count to display a tooltip showing the number of archived comments associated with the review.
 - An indicator showing the number of up votes and down votes

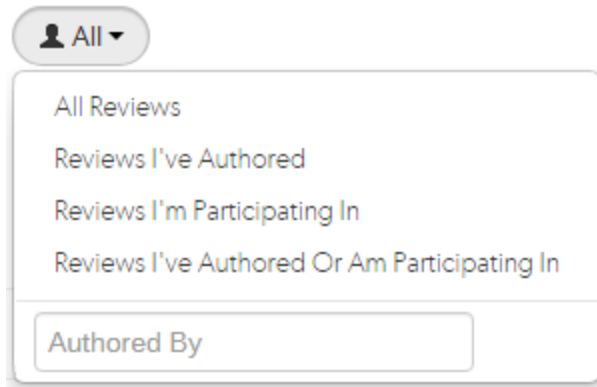
Filtering reviews

The following filtering options are available for code reviews:

- **Project:** a dropdown menu that lets you filter which reviews to display based on project:



- **All Projects:** all reviews are displayed for all projects.
- **My Projects:** all reviews for all of the projects you are participating in, as a member, owner, moderator, or follower.
- **Project Search:** an auto-complete search field that allows you to choose one of the projects defined in P4 Server. Once specified, only reviews for the selected project are displayed. Click the **X** button to remove a *projectid* after it has been specified.
- **Select Project:** selecting a project name displays only reviews for that project.
 When you select one of the available options, the list of options updates to match the currently selected filter, and the **Projects** dropdown indicates the current filter: **All Projects**, **My Projects**, or *projectid*.
- **Users:** a dropdown menu that lets you filter which reviews to display based on user involvement:



- **All Reviews:** displays all reviews.
- **Reviews I've Authored:** displays reviews that you have authored.
- **Reviews I'm Participating In:** displays reviews that you are a reviewer of, but not an author of.
- **Reviews I've Authored Or Am Participating In:** displays reviews that you have authored, or are a reviewer of.
- **Specific User:** An auto-complete search field that allows you to choose one of the user accounts defined in the P4 Server. Once specified, only reviews authored by the user are displayed. Click the **X** button to remove a *userid* after it has been specified.

When you select one of the available options, the list of options updates to match the currently selected filter, and the **Users** dropdown indicates the current filter: **All**, **Author**, **Participant**, or *userid*.

- **Reviewer presence (Opened tab only):**



- **Has Reviewers:** displays reviews that have one or more reviewers.
 - **No Reviewers:** displays reviews that have no reviewers.
- **Review state (Opened tab only):**



- **Needs review:** the review's changes need to be reviewed.
 - **Needs revision:** the review's changes have been reviewed, but further revisions are required before the review can be accepted.
 - **Approved:** the review's changes have been approved, and should be committed.
- **Review state (Closed tab only):**



- **Approved:** the review's changes have been approved and committed.
- **Rejected:** the review's changes have been rejected.
- **Archived:** the review's changes have been put aside.

▪ **Test status:**



- **Tests pass:** when automated tests are enabled for the associated project, and the test suite execution succeeds, P4 Code Review updates the review accordingly.
 - **Tests fail:** similar to the Tests pass state, except that the test suite execution has failed. Check with your test suite to determine why the tests failed.
- **Vote status:**



- **Voted up:** I have voted the review up.
- **Voted down:** I have voted the review down.
- **Not voted:** I am a participant but have not voted on the review.

Note:

Filters for voting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after commenting on it, then this review will not be included in the list.

▪ **Comment status:**



- **Has comments:** I have commented on the review.
- **Does not have comments:** I have not commented on the review.

Note:

Filters for commenting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after voting on it, then this review will not be included in the list.

- **Bookmark:** review filters can be remembered by bookmarking the page, and this icon acts as a reminder of that.



P4 Code Review updates the URL in your browser to reflect filtering options. This makes it easy to bookmark or share review list URLs. P4 Code Review maintains the current filtering if you click on a review link and then use your browser **Back** button to return to the review list.

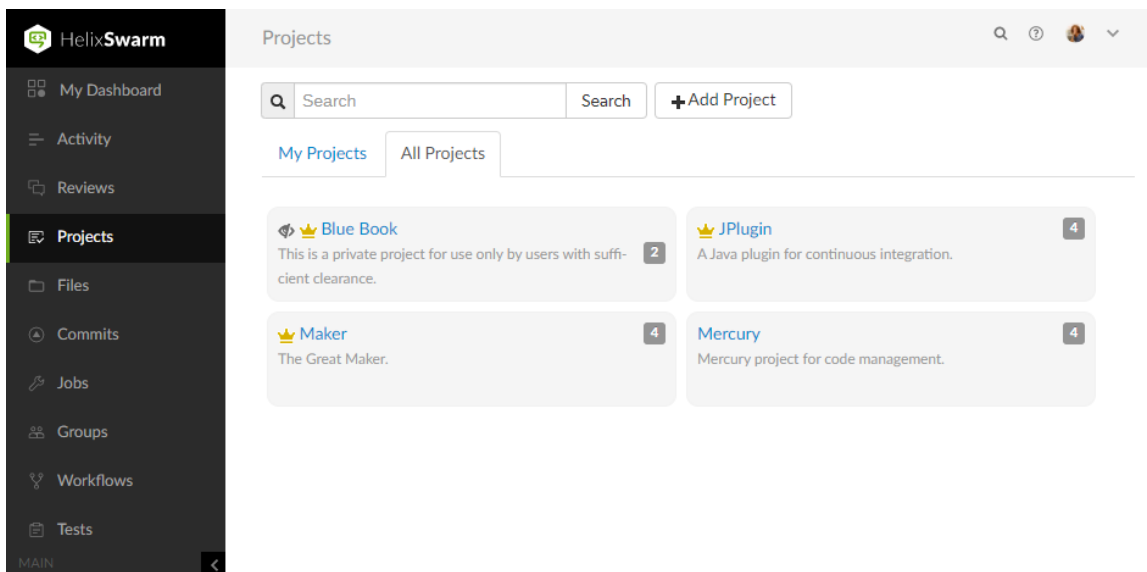
- **Search term:** where review descriptions match your search string.

Projects

A P4 Code Review project is made up of a group of P4 Server [users](#) who are working together on one or more codelines within the P4 Server. A project's definition includes one or more branches of code, and optionally a job filter, [automated test integration](#), and [automated deployment](#). This section provides an introduction to the interactions users have with projects. For details about managing projects, see "[Projects](#)" on page 477.

Listing projects


To view a list of projects, click **Projects** in the menu.



Logged-in users can choose which projects to display by clicking on the **My Projects** tab or the **All Projects** tab. Anonymous users only see the public projects, the **My Projects** tab is not available to anonymous users:

- The **My Projects** tab lists all of the projects that you are an owner of or a member of.
- The **All Projects** tab lists all of the available projects.

Tip:

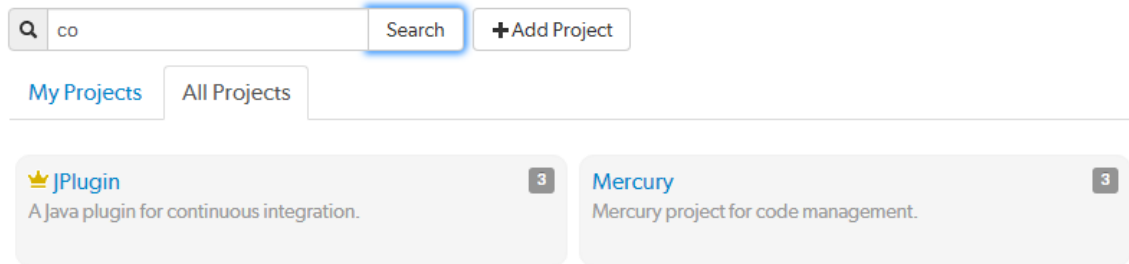
- The **Project Owner** icon  is displayed for projects you own.
- The **Private Project** icon  is displayed for private projects you are authorized to view. For details, see ["Private projects" on page 366](#).
- The number in the icon to the right of the project name displays the total number of members for the project.

Search project names and project descriptions

You can search for P4 Code Review project names and for the content of the project description from the projects page:

1. Enter text in the search box.
2. Click **Search**.

Any project names, and descriptions that match the search text are displayed in the project tab:



Tip:

Switch to the other project tab to display search results for that tab.

Viewing a project

View a project by doing one of the following:

- Click on the project name in the projects list on the P4 Code Review **Projects** page.
- Click on the project name or branch identifier in an [activity stream](#).
- Visit the URL: <https://myswarm.url/projects/project-name>.

Overview page

The project **Overview** page shows a description of the project.

Important:

The project's **Overview** page is only displayed if there is a README markdown file in the project's mainline.

The README file can contain Markdown text, allowing a formatted description of the project to be provided. For a description of the type of formatting that is supported, see ["Markdown in projects" on page 398](#). For details on how to configure the mainline of a project, see ["Mainline branch identification" on page 629](#).

The screenshot shows the Helix Swarm interface for a project named 'JPlugin'. On the left is a dark sidebar with a 'MAIN MENU' and options: 'JPlugin', 'Overview' (selected), 'Activity', 'Reviews', 'Files', 'Commits', and 'Settings'. The main content area has a header 'Projects/Jplugin' with a search bar and user icon. Below this is an 'About' section describing 'JPlugin' as a Java plugin for continuous integration, with statistics: 3 MEMBERS, 1 FOLLOWER, and 2 BRANCHES. It lists OWNERS, MODERATORS, MEMBERS, and FOLLOWERS with their avatars. At the bottom of the sidebar, 'MAIN' and 'CANDIDATE' branches are listed. The main content area features a 'P4 Plugin' title, a description 'Jenkins plugin for a Perforce Helix Versioning Engine (P4D)', and a 'Contents' section with links to 'Release notes', 'Setup guide', 'Notes page', and 'Jenkins page'. Below is a 'Requirements' section listing Jenkins 1.642.3 or greater, Helix Versioning Engine 2012.1 or greater, and Perforce Protection of 'open' for the Jenkins user. An 'Install' section provides a 4-step guide. A 'Building' section states that to build the plugin and run tests, the following should be used.

The following information is displayed in the project sidebar:

- A short description of the project
- A **Follow** button, click to follow the project. This button is not available if you are a member of the project.
- A list of project owners
- A list of project moderators
- A list of project members
- A list of project followers
- A list of the branches defined for the project

Tip:

Hover over a branch name to view the moderators on that branch.

Activity page

The project **Activity** page shows the [activity stream](#) for the project

The following information is displayed in the project sidebar:

- A short description of the project
- A **Follow** button, click to follow the project. This button is not available if you are a member of the project.
- A list of project owners
- A list of project moderators
- A list of project members
- A list of project followers
- A list of the branches defined for the project

Tip:

Hover over a branch name to view the moderators on that branch.

Reviews page

The project **Reviews** page shows a list of code reviews specific to the project.

Reviews

Opened 5 Closed 7


Search for a review

Branch Role Reviewers States Tests Comments Votes

Description	Created	State	Tests	Complexity	Comments	Votes
Added a missing comment. Steve Russell requested a pre-commit review 330, 6 months ago for <code>jplugin:main</code>	6 months ago	Open	1	6	0	0/0
Add comment to method. Steve Russell requested a post-commit review 328, 6 months ago for <code>jplugin:main</code>	6 months ago	Open	1	18	0	1/0
Updated some of the documentation. Allison Clayborne requested a pre-commit review 311, 6 months ago for <code>jplugin:main</code>	6 months ago	Open	1	137	1	3/0

For more details on browsing, filtering, and searching reviews, see ["Reviews list" on page 406](#).

Files page

The project **Files** page shows a list of files for the project, starting with a folder view representing each branch. Branches are designated with the *branch* icon .

Projects/Jplugin

Search

Branch Role Reviewers States Tests Comments Votes

Browse Commits

Show Deleted Files

Name	Modified	Size
main		
candidate		

The project's *main* branch, identified by using a name such as *main*, *mainline*, *master*, *trunk*, is sorted to the top of the list of branches and appears in bold. The list of names can be configured, see ["Mainline branch identification" on page 629](#) for details.

For more information on browsing files, see ["Files" on page 303](#).

Commits page

The project **Commits** page shows a list of changes made to the project.

Projects/Jplugin

Search

Branch Role Reviewers States Tests Comments Votes

Browse Commits

Range User

Change	User	Description	Committed
300	Claire Brevia	Update candidate from main. #review-301	about a month ago
286	Claire Brevia	Updated message text for the application. #review-287	about a year ago
241	Steve Russell	Created candidate branch for the plugin, so that we know what is ready to be released to...	2 years ago
134	Allison Clayborne	Tidying up some of the code to fix some edge conditions. Also putting some support for f...	2 years ago

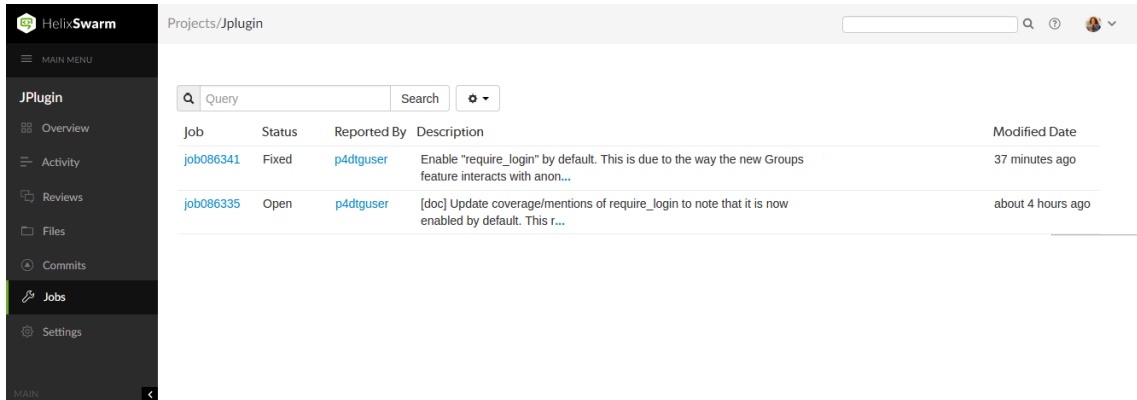
For more details on history browsing, see ["Commits" on page 313](#).

Jobs page

The project **Jobs** page shows a list of jobs associated with the project.

Important:

The **Jobs** page is only displayed when the project configuration includes a *job filter*. See ["Specify a Job filter." on page 496](#) for details.



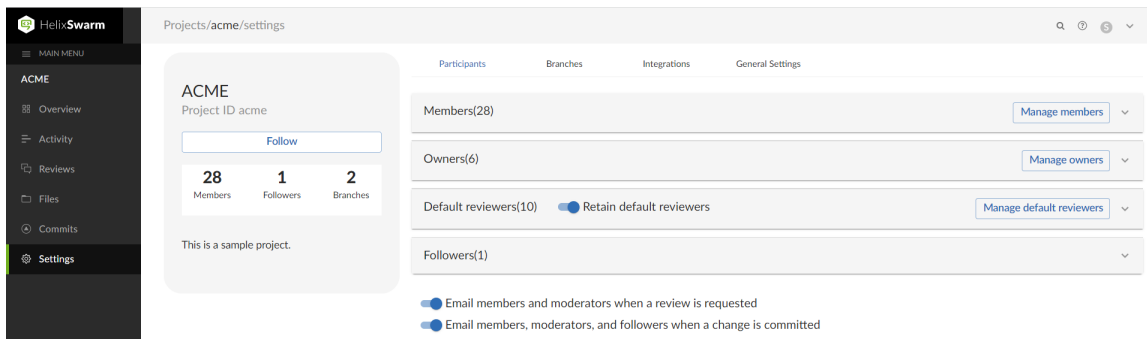
The screenshot shows the HelixSwarm interface for the 'Jplugin' project. The left sidebar contains a 'MAIN MENU' with options: Overview, Activity, Reviews, Files, Commits, **Jobs**, and Settings. The main content area is titled 'Projects/Jplugin' and features a search bar with the text 'Query' and a 'Search' button. Below the search bar is a table of jobs:

Job	Status	Reported By	Description	Modified Date
job086341	Fixed	p4dtguser	Enable "require_login" by default. This is due to the way the new Groups feature interacts with anon...	37 minutes ago
job086335	Open	p4dtguser	[doc] Update coverage/mentions of require_login to note that it is now enabled by default. This r...	about 4 hours ago

For more details on browsing and searching jobs, see ["Jobs" on page 315](#).

Settings page

The project **Settings** page shows the settings associated with the project.



The screenshot shows the HelixSwarm interface for the 'ACME' project settings. The left sidebar contains a 'MAIN MENU' with options: Overview, Activity, Reviews, Files, Commits, **Settings**. The main content area is titled 'Projects/acme/settings' and features tabs for Participants, Branches, Integrations, and General Settings. The 'Participants' tab is active, showing a summary for 'ACME' (Project ID acme) with 28 Members, 1 Followers, and 2 Branches. Below this is a 'Follow' button and a note: 'This is a sample project.' The right side of the page lists various participant groups with their counts and management links:







- Members(28) - Manage members
- Owners(6) - Manage owners
- Default reviewers(10) - Retain default reviewers - Manage default reviewers
- Followers(1)

At the bottom, there are two toggle switches:

- ☒ Email members and moderators when a review is requested
- ☒ Email members, moderators, and followers when a change is committed

Tip:

Open the **Branches** tab and select a branch to view the moderators on that branch.

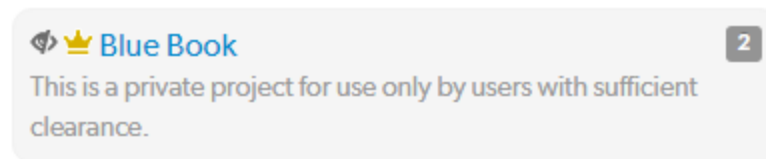
Branch moderators(3), only moderators can approve or reject reviews			Manage moderators
Name	Type		
 Amy Crover	User		
 Delbert Henkel	User		
 My Group	Group		
Rows per page: 25 ▾ 1-3 of 3 < >			

For more information about project settings, see ["Project settings" on page 478](#).

Private projects

Private projects, introduced in P4 Code Review 2016.2, provide a way to make specific projects and their activity less visible to P4 Code Review users. When a project is made private, only the projects owners, moderators, and members, plus users with *super* privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews.

If you are logged in as an owner, moderator, or member of a private P4 Code Review project, that project appears on the P4 Code Review project page with an eye icon to indicate that it is private and has limited visibility:



Similarly, the eye icon appears beside the project's title when viewing the project. When you hover your mouse over the eye icon, a tooltip appears indicating that this project is indeed private.

Caveats

The following are important caveats regarding private projects:

- While P4 Code Review can mask the existence of projects, their activity, and reviews, P4 Code Review honors each user's access to files within the P4 Server. This means that users that have access to the files in a private project's branches can browse to those branch paths within the depot and see the files and any committed changes.
- If you need to prevent access to important files, your P4 Server administrator is required to manage the protections table accordingly. For details, see [Access authorization](#) in the [P4 Server Administration Documentation](#).
- It is possible for a user to start a review that touches files that belong to a private project. If the user is not a member, owner, or moderator of the private project, or does not have super privileges in the P4 Server, they cannot participate in the review.

- If a user is not a member, owner, or moderator of a private project, or does not have super privileges in the P4 Server, and they are added as a review participant (via an ["Links in descriptions and comments" on page 387](#) or by editing a review's participants) to a review containing files within the private project's branches, that user cannot participate in the review: the user cannot see the review, its files, or comments. Due to limitations in how P4 Code Review sends email notifications, such users could still receive notifications for reviews they cannot participate in.
- Notifications usually include links and mentions of the associated projects. Notifications involving private projects are filtered to remove any references to those private projects.

Workflows

Important:

Workflow is enabled by default and can be disabled by your P4 Code Review administrator, see ["workflow " on page 743](#).

A workflow can be applied to a project or project branch to ensure that changelists and code reviews in the project/branch follow the rules specified in that workflow.

- P4 Code Review workflows can be created by any P4 Code Review user.
- Shared P4 Code Review workflows can be viewed by any P4 Code Review user.
- Shared P4 Code Review workflows can be applied to a project or project branch by any P4 Code Review user that is authorized to edit the project.
- The global workflow can be viewed by any P4 Code Review user but can only be edited by a P4 Code Review user with *super* or *admin* privileges, or users that have been made an owner.

Important:

The P4 Code Review administrator can enforce a minimum workflow rule setting (by setting the global rule to **Enforce**) for the entire P4 Server. If a project or project branch has an associated workflow, the global workflow rule is merged with the workflow rule and the most restrictive setting is used.

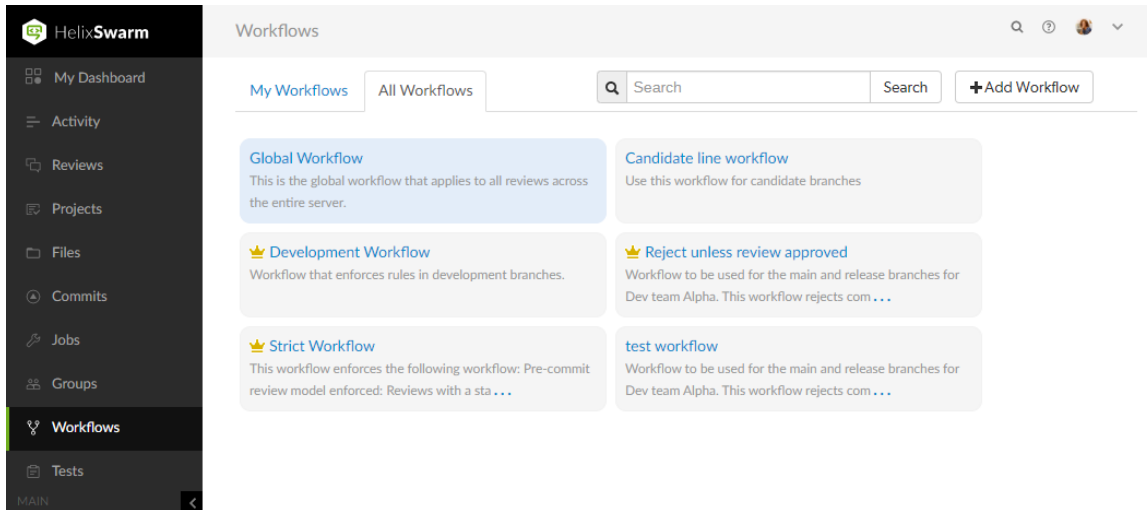
- For more information about how workflow rules are merged, see ["Merging multiple workflows" on page 515](#).
- For more information about setting Global Workflow rules, see ["Workflow global rules" on page 742](#).

This section provides an introduction to listing, searching, and viewing workflows.

- For an overview of workflows, see ["Workflow overview" on page 509](#).
- For instructions on how to add a workflow, see ["Add a workflow" on page 521](#).
- For instructions on how to delete a workflow, see ["Delete a workflow" on page 526](#).
- For instructions on how to add a workflow to a project, and a project branch, see ["Project settings" on page 478](#).

Listing workflows

To view a list of workflows, click **Workflows** in the menu.



Logged-in users can choose which workflows to display by clicking on the **My Workflows** tab or the **All Workflows** tab. Anonymous users only see the global and shared workflows, the **My Workflows** tab is not available to anonymous users:

- The **My Workflows**: tab lists all of the workflows that you are an owner of.
- The **All Workflows**: tab lists the global workflow, all of the shared workflows, and workflows that you are an owner of.

Tip:

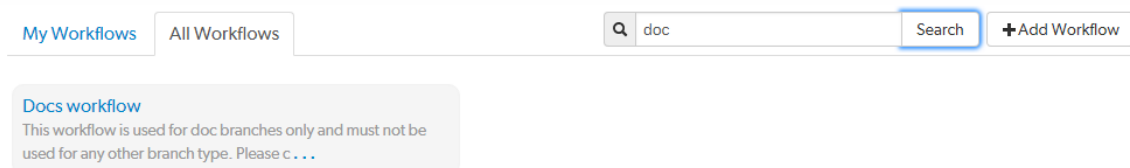
The **Workflow Owner** icon 👑 is displayed for workflows you own.

Search workflow names and workflow descriptions

You can search for P4 Code Review workflow names and for the content of the workflow description from the workflows page:

1. Enter text in the search box.
2. Click **Search**.

Any workflow names, and descriptions that match the search text are displayed in the workflows tab:



Tip:

Switch to the other workflow tab to display search results for that tab.

Viewing a workflow that you own

To view workflow details:

1. Click **Workflows** in the menu.
2. Click on the name of the workflow you want to view.

For information about editing the workflow, see ["Add a workflow" on page 521](#).

Note:

You can edit a workflow if you are the owner of the workflow, or if you have *super* user rights.

Candidate line workflow

Name

Candidate line workflow

Description

Use this workflow for candidate lines

Owners

Add an Owner

Individuals:

bruno

☒ Shared with others

Rules

On commit without a review

Reject

On commit with a review

Reject unless approved

On update of a review in an end state

Reject

Count votes up from

Members

Automatically approve reviews

Based on vote count

Tests

Tests	When	Blocks
Code sniff	On Update	Approve
Doc tests	On Submit	Nothing
Security check	On Demand	Approve

+Add Test

2 projects use this workflow

jPlugin

Branches

1. Candidate

Test Data

Branches

1. Candidate branch

Save

Cancel

Tip:

- The project count does not include project branches.
- If the workflow is associated with a [Private project](#), the private project name is only displayed if you are authorized to view it. Private projects are included in the project count.

Viewing a shared workflow that you do not own

Shared workflows can be viewed and used by all P4 Code Review users.

To view workflow details:

1. Click **Workflows** in the menu.
2. Click on the name of the workflow you want to view.

Main line workflow
x

Name
Main line workflow

Description
Use this workflow for main branches

Owners
Add an Owner

Individuals:
swarm

☒ Shared with others

Rules ⓘ
On commit without a review Allow
On commit with a review Allow
On update of a review in an end state Allow
Count votes up from Anyone
Automatically approve reviews Never

Tests ⓘ	When	Blocks
Code sniff	On Update	Nothing
Build tests	On Submit	Approve
Doc tests	On Demand	Nothing

2 projects use this workflow

jPlugin
Branches
1. Main

Close

Tip:

- The project count does not include project branches.
- If the workflow is associated with a [Private project](#), the private project name is only displayed if you are authorized to view it. Private projects are included in the project count.

Viewing the global workflow

The global workflow can be viewed by any P4 Code Review user but can only be edited by a P4 Code Review user with *super* or *admin* privileges, or users that have been made an owner. For information about editing the global workflow, see ["Workflow global rules" on page 742](#).

To view the global workflow details:

1. Click **Workflows** in the menu.
2. Select the **All Workflows** tab.

- Click on **Global Workflow** to view it.

Tip:

By default, the global workflow is named **Global Workflow**, but it can be changed by your P4 Code Review administrator. To help you identify the global workflow it is always shown as the first workflow in the **All Workflows** tab and it has a different background color to the other workflows.

Global Workflow x

Name

Description

Description

Owners

Add an Owner

Individuals:

swarm

Global Rules ⓘ

On commit without a review

Allow

Enforce ⓘ

On commit with a review

Reject unless approved

☒

On update of a review in an end state

Reject

☒

Count votes up from

Members

☐

Automatically approve reviews

Never

☐

Members or groups who can ignore ALL workflow rules

Member or group

Groups:

swarm-test

Tests ⓘ

Code sniff

On Update

Nothing

Build tests

On Submit

Nothing

Security check

On Update

Approve

Doc tests

On Submit

Nothing

Close

Work-in-progress tag

Add the work-in-progress tag (`#wip` by default) to a pending changelist description to tell P4 Code Review that your work is not ready to be reviewed yet. For example, when changes are requested for a review, and you want to address them in stages but get them all reviewed once your work is complete. When you are ready for your changes to go back into review, delete `#wip` from the changelist description and update your shelf to update the review.

Note:

- By default, the work-in-progress tag is `#wip`, but this can be changed by your P4 Code Review administrator if required. See ["Tag processor"](#) on page 720.

- The following instructions assume your organization uses the default #wip work-in-progress tag. If they are using a different work-in-progress tag, substitute that tag in this page.
- The work-in-progress tag only applies to pending changelists ([pre-commit](#) reviews).

Summary

The work-in-progress tag:

- Prevents a new review from being created for the changelist, even if you have the `#review keyword` in the changelist description.
- Prevents an existing review from being updated when you add, edit, rename, or delete files from the changelist, even if you have the `#review-1234 keyword` in the changelist description.
- Prevents an existing review from being attached to the changelist, even if you have `keywords` in the changelist description. For example, `#append-1234`, `#replace-1234`, or `#review-1234`.
- Prevents the review description from being synchronized if ["Synchronize review description" on page 681](#) is enabled. When the `#wip` tag is deleted from the changelist description, P4 Code Review will synchronize the review description.

When the `#wip` tag is deleted from the changelist, P4 Code Review will update the review when you next update the changelist (add, edit, rename, or delete a file). Simply deleting the `#wip` tag will not update the review.

The work-in-progress tag will not let you break the workflow rules, so it is not a method for getting around the rules set by your organization.

The work-in-progress tag does not prevent a commit from updating its associated review.

Using the work-in-progress tag

Adding a work-in-progress tag to a pending changelist

1. Add `#wip` to your pending changelist description.
2. Update the files in your changelist as often as you need to.

P4 Code Review will not create a new review or update an existing review.

Deleting a work-in-progress tag and updating the review

1. When you are ready for your changes to be reviewed, delete `#wip` from your changelist description.
 2. Update your changelist shelf by making a file change (add, edit, rename, or delete a file).
- P4 Code Review will create a review or update the existing review with your changes so the reviewers can review them.






























Tests

A test in P4 Code Review is an HTTP URL that can be called at different stages of the review process.

You can use tests to automate triggering test runs as well as triggering other URLs when for example a review is created. The requests support URL, JSON and XML encodings and can include various arguments.

You can define multiple tests, and each review can be covered by a set of different tests. Tests are closely coupled with workflows. These workflows control which tests are run when and if they should be launched manually or automatically.

Test results sent back from the test systems are displayed in P4 Code Review.

State	Tests	Complexity	Comments	Votes
		 43	 0	 0  0
		 24	 0	 0  0
		 12	 0	 0  0
		 30	 0	 0  0
		 49	 1	 0  0

Important: The tests page is only available if Workflow is enabled (Workflow is enabled by default).

Associate a test with a [workflow](#) to ensure that the test is run when a review associated with that workflow is started or updated. Associate a test with the [global workflow](#) to ensure that the test is run whenever a review is started or updated. This ensures that the global tests are enforced for all changes even if they are not part of a project.

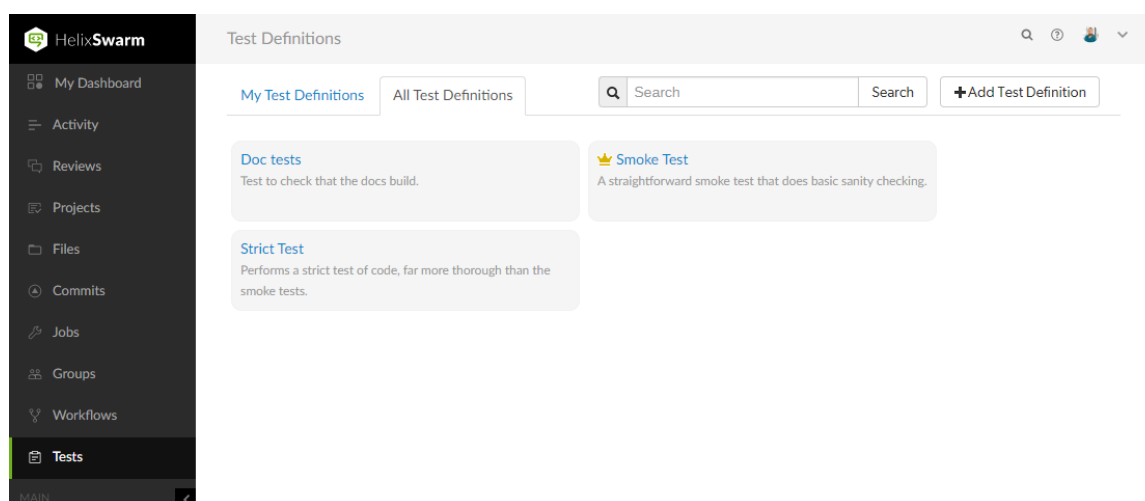
- P4 Code Review tests can be created by any P4 Code Review user.
- Shared P4 Code Review tests can be viewed by any P4 Code Review user.
- Shared P4 Code Review tests can be added to a workflow by any P4 Code Review user that is authorized to edit the workflow.
- Shared P4 Code Review tests can be added to the global workflow by any P4 Code Review user that is authorized to edit the global workflow.

This section provides an introduction to listing, searching, and viewing tests.

- For instructions on how to add a test, see ["Add a test" on page 528](#).
- For instructions on how to edit a test, see ["Edit a test" on page 534](#).
- For instructions on how to delete a test, see ["Delete a test" on page 534](#).
- For instructions on how to add a test to a workflow, see ["Add a workflow" on page 521](#).
- For instructions on how to add a test to the global workflow, see ["Global workflow" on page 744](#).

Listing tests

To view a list of tests, click **Tests** in the menu.



Logged-in users can choose which tests to display by clicking on the **My Tests** tab or the **All Tests** tab. Anonymous users only see the shared tests, the **My Tests** tab is not available to anonymous users:

- The **My Tests**: tab lists all of the tests that you are an owner of.
- The **All Tests**: tab lists the all of the shared tests, and tests that you are an owner of.

Tip:

The **Test Owner** icon  is displayed for tests you own.

Search test names and test descriptions

You can search for P4 Code Review test names and for the content of the test description from the tests page:

1. Enter text in the search box.
2. Click **Search**.

Any tests names and descriptions that match the search text are displayed in the tab:

My Test Definitions

All Test Definitions

Q

proj

Search

+ Add Test Definition

👑 Blue Book tests

Tests for the Blue Book project

👑 JPlugin tests

Tests for the JPlugin project

Jam test

Tests for the Jam project

Tip:
Switch to the other test tab to display search results for that tab.

Viewing a test that you own

To view test details:

1. Click **Tests** in the menu.
2. Click on the name of the test you want to view.

For information about editing the test, see ["Add a test" on page 528](#).

Note:
You can edit a test if you are the owner of the test, or if you have *super* user rights.

Doc tests

Name

Doc tests

Description

Test to check that the docs build.

Owners

+ Add an Owner

Individuals:

allison.clayt

Shared with others

☒

URL

http://jenkins_host:8080/job/[branches]-review-test/review/build

Timeout(seconds)

20

Body

status={status}&review={review}&change={change}&update={update}

☒ URL Encoded

☐ JSON Encoded

☐ XML Encoded

☒ Iterate tests for affected projects and branches

Header

Value

BasicAuth

YWxpY2U6QUJDREYzMzQ1Njc4Cg==

+ Add header

Standard arguments (URL and Body fields only)

(change) the change number

(status) the status of the shelved change, shelved or committed

(review) the review identifier

(version) the version of the review

(reviewStatus) the Swarm status of the review: needsReview, needsRevision, archived, rejected, approved, approved:commit

(description) the change description of the change (request body only)

(test) the name of the test

(testRunId) the test run id

(projects) the project identifiers of projects that are part of the review

(projectNames) the project names of projects that are part of the review

(branches) the branch identifiers of branches that are part of the review

(update) the update callback URL

(pass) the tests pass callback URL

(fail) the tests fail callback URL

1 workflow uses this test definition

Strict Workflow

Save

Cancel

Delete

Viewing a shared test that you do not own

Shared tests can be viewed and used by all P4 Code Review users.

To view test details:

1. Click **Test** in the menu.
2. Click on the name of the test you want to view.

To avoid leaking sensitive information to users that do not own the test, configuration details are hidden.

Doc tests x


Name

Doc tests

Description

Test to check that the docs build.

Owners

 Add an Owner

Individuals:

allison.clayb

☒ Shared with others

Some details of this test definition are private and cannot be displayed. Only the test definition owners have access to these details.

▼ 1 workflow uses this test definition

Strict Workflow

Close

Notifications

Provided you have entered a working email address for your userid in P4 Server, P4 Code Review sends email notifications to you when various events take place. The table below shows the notifications sent if all of the project, group, and user notifications are enabled. This is the default behavior.

Note:

The system-wide default notifications can be changed by the system owner. See ["Global settings"](#) on page 654 for details.

Tip:

Many of the notifications can be disabled, this allows project, group, and user notifications to be customized to limit the number and type of notifications sent.

- Project notifications are configured on the [project settings tab](#).
- Group notifications are configured on the [group Notifications tab](#).
- User notifications are configured on the [user Notifications tab](#).

Legend:

A = Author	PM = Project member or project member group	Yes = role user/group receives notification
R = Reviewer or reviewer group	PF = Project follower	No = role user/group does not receive notification
M = Moderator or moderator group	AF = Author follower	NA = role not included for this event
GM = Group member	CA = Comment author	

Event	Notification sent to one of these roles...								
	Event by you?	A	R	M	GM	PM	PF	AF	CA
A review is committed in P4 Code Review	No	Yes	Yes	Yes See (3)	Yes See (4)	Yes See (1)	Yes See (1)	No	NA
A review is committed outside of P4 Code Review	No	No	Yes	Yes See (3)	Yes See (4)	Yes See (1)	Yes See (1)	No	NA

Event	Notification sent to one of these roles...								
	Event by you?	A	R	M	GM	PM	PF	AF	CA
A change is committed outside of P4 Code Review See "(1) Committed change notifications" on page 381	No	No	NA	Yes See (3)	Yes See (4)	Yes See (1)	Yes See (1)	Yes See (1)	NA
A review is started See "(2) Review start notifications" on page 382	Yes	Yes	Yes	Yes See (3)	Yes See (4)	Yes See (2)	No	No	NA
A review is created automatically by the On commit without a review workflow rule	No	No	Yes	No	No	No	No	No	NA
A reviewer casts a vote	Yes	Yes	Yes See (5)	No	Yes See (4)	No	No	No	NA
A review's state changes	No	Yes	Yes See (5)	No	Yes See (4)	No	No	No	NA

Event	Notification sent to one of these roles...								
	Event by you?	A	R	M	GM	PM	PF	AF	CA
A review's reviewers are changed	No	Yes	Yes See (5)	No	Yes See (4)	No	No	No	NA
A review's files are updated	Yes	Yes	Yes See (5)	No	Yes See (4)	No	No	No	NA
A review's automated tests have finished See "(7) Tests have finished" on page 383	No	Yes	Yes	Yes	Yes See (4)	No	No	No	NA
A review's description is changed	No	No	No	No	No	No	No	No	NA
A review comment is created See "Comment notification delay" on page 336	No	Yes	Yes See (5)	No	Yes See (4)	No	No	No	No See (6)
A review comment is edited See "Comment notification delay" on page 336	No	Yes	Yes See (5)	No	Yes See (4)	No	No	No	No See (6)

Event	Notification sent to one of these roles...								
	Event by you?	A	R	M	GM	PM	PF	AF	CA
A committed change comment is created	No	Yes	NA	No	Yes See (4)	No	No	No	No See (6)
A committed change comment is edited	No	Yes	NA	No	Yes See (4)	No	No	No	No See (6)
Someone joins or leaves a review	No	Yes	Yes See (5)	No	Yes See (4)	No	No	No	NA
Someone likes a comment you wrote	No	No	No	No	No	No	No	No	Yes See (5)
You like a comment you wrote	Yes	No	No	No	No	No	No	No	No

Any custom modules added to P4 Code Review may also send notifications.

Important:

Email delivery for events related to restricted changes is disabled by default. See ["Restricted Changes"](#) on page 695 for details on how to enable restricted change notifications.

@mention notifications

Users: @mention

Using an [@mention](#) in a review, changelist, or comment causes the referenced userid to receive a notification and be included in any future notifications regarding the associated file or review.

When a comment is added to a job, Swarm sends a notification to users listed in user fields in the job, users [@mentioned](#) in the job description, and the authors of any associated changes.

Note:

Typically, you would not receive a notification when you @mention yourself. @mentions themselves do not trigger notifications; they inform who receives notifications. See "[notify_self](#)" on [page 598](#) for details on how to enable self notification.

Groups: @@mention

Using an [@@mention](#) in a review, changelist, or comment causes the referenced groupid members to receive a notification and be included in any future notifications regarding the associated file or review.

When a comment is added to a job, Swarm sends a notification to groups listed in group fields in the job, groups @@mentioned in the job description, and the authors of any associated changes.

Note:

- **Group mailing list enabled:** notifications are sent to the group email address.
When a group mailing list is enabled it overrides the group member's preference set for the `notify_self` configurable in the [SWARM_ROOT/data/config.php](#) file. So even when the `notify_self` configurable is set to false the group member will receive an email notification.
There is a caveat that if a user is a group member and the same user is added individually to the review and has set the `notify_self` configurable to true, the user receives two email notifications.
- **Group mailing list disabled:** notifications are sent to the group members individual email addresses.

(1) Committed change notifications

Notifications for committed changes are sent by default, but can be disabled or require users to opt-in. Please see the [notification configuration](#) and [project settings overview](#) for more details.

When committed change notifications are configured for opt-in, you need to copy the configuration's special depot path to the **Reviews:** field in your user spec within P4 Server. Once you have done so, P4 Code Review can send you committed change notifications for any change that matches a depot path specified in the **Reviews:** field.

Update Reviews with p4

1. Begin editing your user spec:

\$ p4 user

2. Edit the **Reviews:** field to include the special depot path, plus any other paths you want to receive notifications for.
3. Save the spec.

Update Reviews with P4V

1. Select **Connection > Edit Current User**.
2. Edit the **Reviews:** field to include the special depot path, plus any other paths you want to receive notifications for.
3. Click **OK**.

(2) Review start notifications

By default, notifications are sent when a review is started, but can be disabled for a project, group, or user.

Note:

If a user or group is added as a reviewer when a new review is created, the reviewer will be notified that the review has been started. This initial notification cannot be muted.

(3) Moderator notifications

Moderators and moderator groups are associated with specific project branches. Moderators receive notifications for reviews and commits against the branches they are associated with.

(4) Group member notifications

Groups can be members of a project, moderators of a project branch, and reviewers of a review. The notifications that group members receive will depend on, the role the group has and the group notification settings that are configured for that group. Notifications received by group members also depend on whether the group has a group mailing list enabled, in this case additional notification settings are available. See *Group mailing list enabled* below for details.

Group mailing list disabled

Group members can be notified when a member of the group starts a review. Group members can also be notified when a change is committed by, or on behalf of, a changelist owner who is also a member of the group. These two notifications can be individually selected as required. See ["Add a group" on page 471](#) for details. Notifications are sent to the group email address.

Notifications are sent to the group members individual email addresses.

Group mailing list enabled

Group members can be notified when a member of the group starts a review. Group members can also be notified when a change is committed by, or on behalf of, a changelist owner who is also a member of the group. These two notifications can be individually selected as required.

When a group mailing list is enabled it overrides the group member's preference set for the `notify_self` configurable in the `SWARM_ROOT/data/config.php` file. So even when the `notify_self` configurable is set to false the group member will receive an email notification.

There is a caveat that if a user is a group member and the same user is added individually to the review and has set the `notify_self` configurable to true, the user receives two email notifications.

For more information about `notify_self` configurable, see ["notify_self" on page 598](#).

Additional notifications are available if the group mailing list address is enabled. See ["Add a group" on page 471](#) for details.

(5) Disable notifications

By default, notifications for events in a review are sent to all reviewers. Reviewers can disable notifications during a review to avoid further email notifications. Once notifications are disabled for a reviewer, they can be re-enabled when they are specifically [@mentioned](#); reviewers can disable notifications again after they have been re-enabled. See ["Disable notifications" on page 420](#) for details.

(6) Comment author notifications

By default, notifications are not sent to comment authors, but P4 Code Review can be configured to send notifications of comments to comment authors. See ["notify_self" on page 598](#) for details.

(7) Tests have finished

By default, notifications are sent when:

- Automated tests have failed for a review.
- The first time automated tests pass for a review after a test failure for that review.

Tip:

The test notifications email contains the complete URL to the tests.

Log in/Log out

When you are not logged into P4 Code Review, certain features are unavailable to you, such as providing comments to changes or reviews, adding projects, and more.

Log in with a password

1. Click **Log in** on the right of the P4 Code Review header.
2. Type in your username and password, appropriate for the P4 Server that P4 Code Review is configured to use.
3. Select the **Remember me** check box if you prefer to stay logged in between browser restarts.

Note:

The P4 Server can enforce maximum log in times. You may become logged out even if **Remember me** is selected. P4 Code Review administrators can change the maximum log in time, see ["Sessions" on page 687](#) for details.

4. Click **Log in with credentials**.

Log in with SSO

1. If your organization has single sign-on (SSO) configured for P4 Server, you can use it to log in through your organization's identity provider.
2. Open P4 Code Review.
3. Enter your **Email** or **Username**.
4. Click **Log in with SSO**.
You are prompted to log in to the identity provider.
5. Enter your identity provider credentials.

If your login is successful, the P4 Code Review home page opens. See ["Quickstart" on page 23](#) to get started with P4 Code Review.

To log in manually to P4 Code Review using your username and password instead, click **Log in with credentials**. If you switch to log in with credentials and then decide to use single sign-on, click **Log in with SSO**.

Log out of P4 Code Review

1. Click your userid, on the right of the P4 Code Review header.
2. Select **Log Out** from the drop-down menu.

Tip:

- If P4 AS is configured for your P4 Server, logging out of P4 Code Review will not invalidate your Identity Provider (IdP) login status. If you try to log back in to P4 Code Review while your IdP status is still valid, you will not be prompted to complete the log in steps.

If `require_login` is also enabled, P4 Code Review will return you to the login and your IdP will automatically log you back in. In this case log out from your Identity Provider page before logging out from P4 Code Review.

- If a custom redirect has been configured by your P4 Code Review administrator, you are logged out of P4 Code Review and then redirected to the URL specified by the administrator.

The custom redirect can be set to any internal or external URL, for example:

- Company intranet, extranet, internet, FTP, or Web-mail page
- Industry news website
- Identity Provider page to invalidate your IdP log in status

require_login

By default, P4 Code Review requires users to log in, which prevents anonymous users from accessing a P4 Server via P4 Code Review. Users who have not logged in see a log in page immediately when visiting P4 Code Review:

The steps to log in are identical to using the Log in dialog.

P4 Code Review administrators can disable [require_login](#) to allow anonymous users to see commits, reviews, etc.

Note:

service and *operator* users are not permitted to login. See [Types of users](#) in the [P4 Server Administration Documentation](#).

Notable minor features

Quick URLs

P4 Code Review handles URLs intelligently to reduce the amount of information you need to enter to quickly locate what you are looking for:

1. Enter a URL in the following format:
`https://myswarm.url/<identifier>`
2. P4 Code Review checks the `<identifier>` and redirects you to the first match it finds. P4 Code Review checks for the identifier in following order:
 - [Reviews](#)
 - [Changelists](#)
 - [Depot paths](#)
 - [Projects](#)
 - [Jobs](#)
 - [Users](#)
 - [Groups](#)
 - Depot names

Note:

- Changelist and review identifiers must be numeric.
- If you enter an identifier that does not exist for any type of resource, P4 Code Review displays a Page Not Found error.

Example usage

Display the latest version of a review

For example, to display review 124: enter the URL `https://myswarm.url/124`, P4 Code Review redirects to `https://myswarm.url/reviews/124`.

Tip:

- To display a specific version of a review, enter the full URL including reviews and append `/v<n>/` to the URL. For example, to display version 2 of review 124: enter `https://myswarm.url/reviews/124/v2/`
- To display a diff between two versions of a review, enter the full URL including reviews and append `/v<n,n>/` the URL. For example, to display the diff between version 2 and 4 of review 124: enter `https://myswarm.url/reviews/124/v2,4/`
- To open a review with a specific tab open, append `#<tab name>` to the end of the URL (`#<tab name>` must be the last item in the URL):
 - **Files tab:** `#files` (default if `#<tab name>` is not specified)
 - **Comments tab:** `#comments`
 - **Activity tab:** `#activity`

Display a changelist

For example, to display changelist 123: enter the URL `https://myswarm.url/123`, P4 Code Review redirects to `https://myswarm.url/changes/123`.

Tip:

To open a changelist with a specific tab open, append `#<tab name>` to the end of the URL:

- **Files tab:** `#files` (default if `#<tab name>` is not specified)
- **Comments tab:** `#comments`

Display the latest version of a file

Enter the URL `https://myswarm.url/depot/alpha/readme.txt`, P4 Code Review redirects to `https://myswarm.url/files/depot/alpha/readme.txt`.

Tip:

- To display a specific version of a file, enter the full URL including files and append `?v=<n>` to the URL. For example, to display version 2 of the `depot/alpha/readme.txt` file: enter `https://myswarm.url/files/depot/alpha/readme.txt?v=2`
- To open a file with a specific tab open, append `#<tab name>` to the end of the URL (`#<tab name>` must be the last item in the URL):
 - **View tab:** `#view` (default if `#<tab name>` is not specified)
 - **Commits tab:** `#commits`

Display the user profile page for jsmith

Enter the URL `https://myswarm.url/jsmith`, P4 Code Review redirects to `https://myswarm.url/users/jsmith`.

Links in descriptions and comments

When you write a job description, changelist description, review description, or comment, you can link to users, groups, changelists, reviews, and jobs. P4 Code Review will attempt to automatically create links to your references to make it easy to navigate to the specified resource.

In this section:

- ["@mentioning users and groups" below](#)
- ["Linking to a review " below](#)
- ["Linking to a review, changelist, or numeric userid" on the facing page](#)
- ["Linking to a job" on the facing page](#)

@mentioning users and groups

To link to users or groups, use `@mention` to refer to users and `@@mention` to refer to groups when you write a job description, changelist description, review description, or comment. P4 Code Review automatically creates a link for each `@mention` and `@@mention`.

When you start a code review, including a user `@mention` or a group `@@mention` in the changelist description automatically makes them reviewers for that code review. During a code review, including a user `@mention` or group `@@mention` in a comment causes the mentioned user or group to receive [notifications](#) of code review events, even if they are not a member of your project or following you or your project.

Additional option for a user `@mention`:

- `@*mention`: include an asterisk (*) before the `userid` to make the user a required reviewer. See ["Required reviewers" on page 457](#) for details.

Additional options for a group `@@mention`:

- `@@*mention`: include an asterisk * before the `groupid` to make the group a required reviewer with **All votes required**. See ["Required reviewers" on page 457](#) for details.
- `@@!mention`: include an exclamation mark ! before the `groupid` to make the group a required reviewer with **One vote required**. See ["Required reviewers" on page 457](#) for details.

Note:

- **Group mailing list enabled:** notifications are sent to the group email address.
- **Group mailing list disabled:** notifications are sent to the group members individual email addresses.

Linking to a review

Important:

Adding `#review-<reviewid>` to a changelist description is a special case and does not just add a simple link to the review. It replaces all of the files in the review with the files in the changelist. For information about adding a changelist to a review, see ["Add a changelist to a review" on page 670](#).

To make a simple link to the review, use `review-<reviewid>`.

To create a link to a review, add one of the following patterns to a job description, review description, or comment:

- `#review-<reviewid>`

For example, when you include `#review-12345` in a review description, P4 Code Review automatically turns that text into a link that, when clicked, displays the Review page for review 12345.

- `review-<reviewid>`

For example, when you include `review-12345` in a review description, P4 Code Review automatically turns that text into a link that, when clicked, displays the Review page for review 12345.

- `review <reviewid>`

For example, when you include `review 12345` in a review description, P4 Code Review automatically turns that text into a link that, when clicked, displays the Review page for review 12345.

- `@<reviewid>`

For example, when you include `@12345` in a review description, P4 Code Review automatically turns that text into a link that, when clicked, displays the Review page for review 12345.

Linking to a review, changelist, or numeric userid

If you add `#<numericid>` to a job description, changelist description, review description, or comment. P4 Code Review automatically creates a link to the page for the numeric id specified if a match exists.

P4 Code Review checks the numeric id specified and links to the first match it finds. P4 Code Review checks for the numeric id in the following order:

- Reviews
- Changelists
- Users

For example, when you include `#12345` in a comment, P4 Code Review turns it into a link to the page for either review 12345, change 12345, or user 12345, depending on which of them exists.

Linking to a job

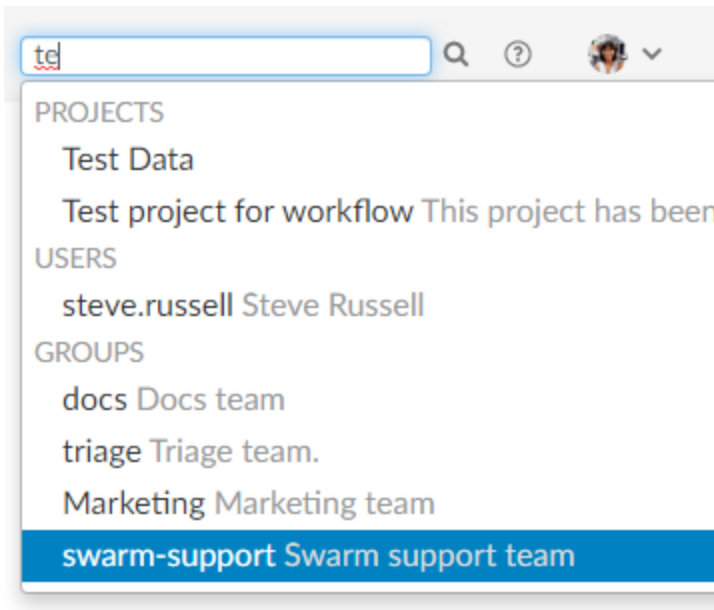
To create a link for a job, add `jobnnnnn` or `@jobnnnnn` to a job description, changelist description, review description, or comment. P4 Code Review automatically creates a link to the job page if a match is found.

For example, when you include `@job12345` in a comment, P4 Code Review turns that text into a link that, when clicked, displays the Job page for job12345.

Tip:

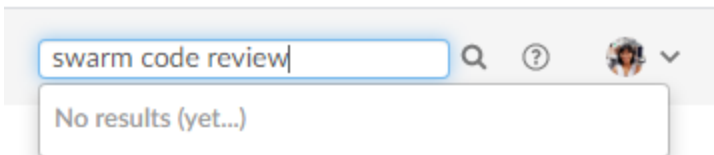
If your job numbers are not formatted as jobnnnnn, your P4 Code Review admin can configure P4 Code Review to recognize other keywords in addition to job. For example, if your job numbers are formatted as defectnnnnnn. For instructions on adding a new job keyword to P4 Code Review, see ["Job keywords" on page 619](#).

Search



P4 Code Review can search for users, groups, projects, and file paths. Enter keywords or path elements into the search box in the P4 Code Review header. Navigate the results with the ↑ (up arrow) and ↓ (down arrow) keys. To display the details for a result, press **Enter** or click the search result.

Full-content searching is only available if your P4 Code Review administrator installs the P4 Server Search Tool. See ["Search" on page 684](#) for details.



P4 Code Review updates the search results as you type. Some results should appear within a second or two. You may have to wait a few seconds for final results to be incorporated into the results list. When P4 Code Review does not yet have any results, it indicates such.

Jira integration

P4 Code Review ships with a Jira module that can:

- create links to Jira when P4 Code Review displays Jira issue identifiers in changelists, jobs, reviews, etc. Jira links in comments are only supported in the classic review page.
- add links within Jira back to P4 Code Review, for Jira issues associated with reviews or committed changes. These links reflect the current status of associated code reviews.

By default, the Jira module is disabled. For instructions about enabling Jira integration, see "[Jira](#)" on [page 535](#).

Avatars

Each event in an activity stream includes an *avatar*, an image that represents the user responsible for the event. Avatars help to visually tie together various events and personalize the history presented in the stream.

Based on the email address entered in a user's or group's P4 Server account, P4 Code Review attempts to fetch an avatar from [gravatar.com](#). P4 Code Review selects an avatar from its collection of default avatars if an avatar is not returned from [gravatar.com](#).

Hover your mouse over an avatar to display the fullname of the user or group.

Following

Whenever you see a **Follow** button, for example when you are viewing a project page or user profile, clicking the button causes P4 Code Review to send you notifications whenever there is activity generated by the current resource.

This is useful if, in the case of a project, you are not a project member but want to know what's happening in the project. Or, in the case of a user, you want to see what activity that user generates.

To stop receiving notifications, visit the project page or user profile and click the **Unfollow** button.

Time

P4 Code Review typically displays the time of an event, such as when a file was created, as about *X units ago*. Hover your mouse pointer over a time display to see a tooltip displaying the exact date and time of the event.

Tip:

Configure P4 Code Review to display the exact time and date of events by setting **Time Display** to **Timestamp** on your user [Settings tab](#).

Keyboard shortcuts

P4 Code Review provides the following file control keyboard shortcuts:

Keyboard shortcut	Usage
Alt + H	Opens the shortcut dialog that displays a list of all the keyboard shortcuts for P4 Code Review. See "Show the shortcut help dialog" on page 434 .
Shift + Alt + L	Toggles to open or close the file list. See "File list" on page 434 .
Shift + Alt + S	Toggles to open or close the Information panel. See "Information panel" on page 423 .
Alt + R	Marks a file as read. See "Mark file as read" on page 436 .
Alt + T	Opens file in a new tab. See "Open content in a new tab" on page 436 .
Alt + N	Scrolls to the next file in the list while viewing a review. See "Navigate to the next file" on page 434 .
Alt + P	Scrolls to the previous file in the list while viewing a review. See "Navigate to the previous file" on page 434 .
Shift + Alt + F	Toggles to expand the Diff view to full screen or collapse the Diff view. See "Expand the Diff view to full screen" on page 432 .
Shift + Alt + E	Toggles to expand or collapse all files in a review. See "Expand or collapse all files in a review" on page 433 .
Alt + L	Toggles to show diffs in-line or side-by-side. See "Show diffs in-line or side-by-side for a file" on page 433 .
Alt + W	Toggles to show or hide whitespace and tab characters for all text files in a review. See "Show or hide whitespace and tab characters for all files in a review" on page 433 .

Keyboard shortcut	Usage
Alt + D	Toggles to show or hide whitespace diffs for all files in a review. See "Show or hide whitespace diffs for all files in a review" on page 434 .
Alt + C	Toggles to show or hide all the inline file comments. See "Show or hide all in-line comments for all files in a review" on page 433 .
Shift + Alt + H	Toggles to show or hide syntax highlighter for a given coding language. See "Disable syntax highlighter" on page 434 or for a list of supported languages, see "Supported syntax highlighting in the Review page" on page 311 .
N	Scrolls to the next difference while viewing "Diffs" on page 322 .
P	Scrolls to the previous difference while viewing "Diffs" on page 322 .
Ctrl + Enter	Submits a comment or form. Enter a comment in the text area and then use Ctrl + Enter to submit the comment or form.
ESC	Closes the dialog while you are viewing a dialog. Stops text entry while you are entering text into a text area.

About P4 Code Review

You can discover the version of P4 Code Review you are using:

1. [Log in](#) to P4 Code Review.
2. Click your userid, found at the right of the main toolbar, and select **About P4 Code Review**.
A dialog appears displaying the P4 Code Review version.

Custom error pages

If P4 Code Review encounters an error during processing, such as when a ["Quick URLs" on page 385](#) is used that points to a non-existent resource, P4 Code Review displays a custom error page.

Short links

P4 Code Review provides a *short link* feature that creates shorter URLs than normal to make sharing specific views within P4 Code Review easier. It is also possible to register or configure an alternate, shorter hostname to have even shorter URLs. See ["Short links" on page 702](#) for details.

Conceptually, this is identical to [TinyURL](#), or the Twitter feature [t.co domain](#), but is restricted to P4 Code Review URLs on a hostname you control.



P4 Code Review displays a *bookmark* button when viewing files or folders in the depot.

Click the button to display a popup containing the short link. Press **CTRL+C** (on Windows and Linux), or **Command+C** (on macOS), to copy the short link. You can then paste the short link anywhere you'd like to share the current file or folder view in P4 Code Review.

Markdown

You can use Markdown to add text styles to comments, review descriptions, and project overview pages. P4 Code Review can also display Markdown files stored in the P4 Server.

Tip:

- The [Review Activity](#) tab is rendered in GFM (Git Flavored Markdown).
- The [User](#), [Group](#), and [Project](#) page **Activity** tabs and the [Activity page](#) are rendered in Parsedown.

In this section:

- ["Common text styles" below](#)
- ["Text styling example" on page 395](#)
- ["Markdown in comments and review descriptions" on page 397](#)
- ["Markdown in projects" on page 398](#)

Common text styles

Markdown renderers supported

P4 Code Review uses GFM (Git Flavored Markdown) or Parsedown to render text styles. The renderer used depends on where you are entering your text in P4 Code Review:

- Review comments and review descriptions support GFM (Git Flavored Markdown), see [Basic writing and formatting syntax](#)
- Project overview pages, changelist comments, and jobs page comments support Parsedown, see the [Parsedown demo](#)

Common GFM and Parsedown styles

A number of the more common text styles are shown below:

- **bold** and *italics* can be specified with ****bold**** or **italics**.
- Unordered lists can be specified with asterisk (*) markers. Plus (+) and minus (-) signs also work:

- * This
- * That
- * Another

- Ordered lists can be specified with numbered markers:

1. First
2. Second
3. Third

- Hypertext links can be specified with [Link text](http://address/page). If you don't need to add anchor text, then a URL in the text without any markup is linkified.
- You can mark inline text as code using `backticks`.
- You can also mark blocks of code using three backticks on a line, like:

```
...
var text = "Some code text";
alert(text);
...
```

- Headers can be defined using hash (#) marks.

```
# Main heading
## Sub heading
### Lesser heading
```

- Images can be added as well. You can either provide the full P4 Code Review *view* URL, or use a relative P4 Code Review *view* URL to reference something in P4 Code Review.

```
![alt text](https://swarm.company.com/view/depot/www/dev/images/jamgraph-jam.gif
"Title Text")
![alt text](/view/depot/www/dev/images/jamgraph-jam.gif "Title Text")
```

Tip:

To find the P4 Code Review view URL for an image file:

1. From P4 Code Review, click on the **Files** tab and navigate to the image file.
2. Click on the image file so that it opens in the **View** tab.
3. Click the **Open** button for the file to open the image in you browser.
4. The URL in your browser address bar is the P4 Code Review *view* URL for the image file.

- Link to a YouTube video using the **Share** --> **COPY** feature for the YouTube video. For example:

```
[[!alt text](https://img.youtube.com/vi/wimka8j4XBk/mqdefault.jpg)]  
(https://youtu.be/wimka8j4XBk)
```

- Tables are supported by using pipe (|) separators between columns and colons (:) for justification.

```
| Tables | Look | Like this |  
|-----| ---: | :-----: |  
| Left   | right | center   |
```

- It is also possible to blockquote paragraphs using the greater than symbol (>).

```
> Blockquotes can be displayed like this, using the  
> the greater than sign at the start of the line.
```

Normal text resumes here.

Text styling example

The following example shows how Markdown text can be used in a comment but it would display the same for a project overview page.

The source and the final result are shown:

- ["Source text" below](#)
- ["Rendered view of Markdown text" on page 397](#)

Source text

Comments can include **Markdown** text, which allows basic styles to be applied to the text.

You can have unordered lists, like this:

- * A line
- * Another line
- * A sub list
- * Again
- * Back again

Or ordered lists:

1. First
2. Second
3. Third

It is possible to mark text as `{ like: this }` or to define a block of text as code:

```
```
```

```
var i = 1;
var j = 10;
```

```
for (i = 1; j != i; i++) {
 print i * j;
}
...
```

You can also [create hyper links](<http://www.perforce.com>) which point to other places.

Use the backslash character \ to escape Markdown syntax characters.

You can escape the following characters:

Asterisk \\*

Underscore \\_

Curly braces \{ \}

Square brackets \[ \]

Brackets \( \)

Hash \#

Plus \+

Minus \-

Period \.

Exclamation point \!



## Rendered view of Markdown text



**claire.brevia** commented 6 minutes ago (edited)



Comments can include **Markdown** text, which allows basic styles to be applied to the text.

You can have unordered lists, like this:

- A line
- Another line
  - A sub list
  - Again
- Back again

Or ordered lists:

1. First
2. Second
3. Third

It is possible to mark text as `code { like: this }` or to define a block of text as code:

```
var i = 1;
var j = 10;
for (i = 1; j != i; i++) {
 print i * j;
}
```

You can also [create hyper links](#) which point to other places.

Use the backslash character \ to escape Markdown syntax characters.

You can escape the following characters:

Asterisk \*  
 Underscore \_  
 Curly braces {}  
 Square brackets []  
 Brackets ()  
 Hash #  
 Plus +  
 Minus -  
 Period .  
 Exclamation point !

[Reply](#) · [Edit](#) · ❤

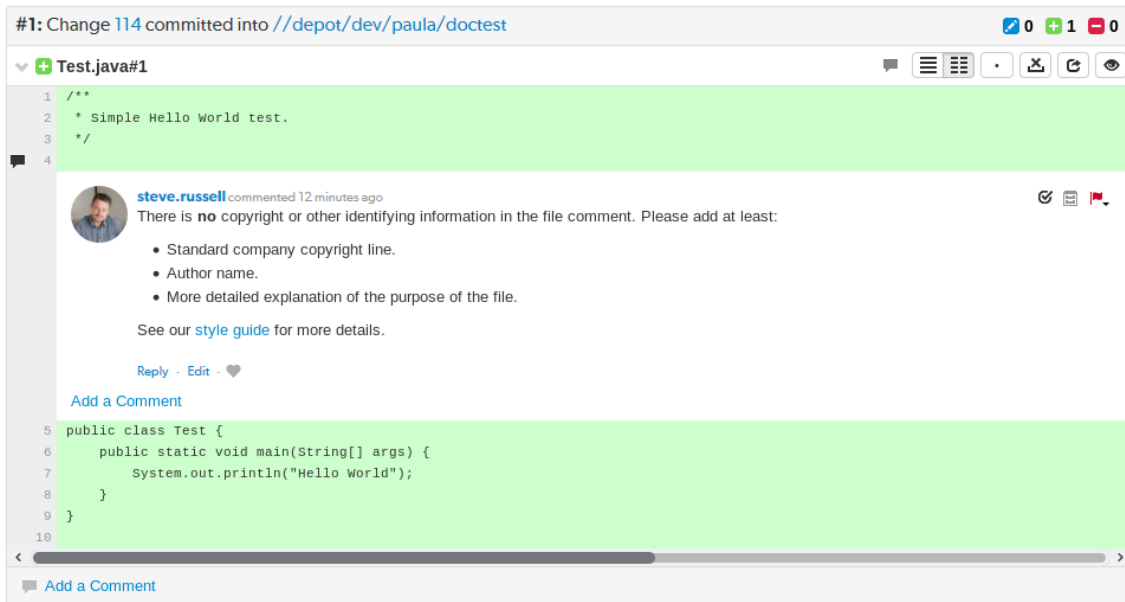
## Markdown in comments and review descriptions

### Tip:

- Try to avoid using styles like headers, images, and tables in comments, they can make comments harder to read this is especially true inline comments.
- **Renderer supported:**
  - Review comments and review descriptions support GFM (Git Flavored Markdown), see [Basic writing and formatting syntax](#).
  - Changelist comments and jobs page comments support Parsedown, see the [Parsedown demo](#).

Markdown content is displayed in comments and review descriptions, but Markdown support is limited to prevent execution of raw HTML and JavaScript content.

If you use Markdown styles in your comments and review descriptions, P4 Code Review renders them when you save the changes. The same markdown is displayed in the P4 Code Review HTML emails.



## Markdown in projects

If a project has a `README` markdown file in the root of its `MAIN` branch, that `README` markdown file is displayed as the overview page for the project.

By default, Markdown content is displayed on the project overview page but is limited to prevent the execution of raw HTML and JavaScript content. This can be configured by the P4 Code Review administrator.

- Markdown support for displaying the project overview page can be disabled by your P4 Code Review administrator, see ["Project readme" on page 660](#).
- The level of Markdown support can be configured by your P4 Code Review administrator, see ["Markdown" on page 631](#).

If you need to change which branch is considered `MAIN`, and therefore from where the `README` is read from, see ["Mainline branch identification" on page 629](#).

### Tip:

- Valid Markdown file extensions are: `md`, `markdown`, `mdown`, `mkdn`, `mkd`, `mdwn`, `mdtxt`, `mdtext`.
- If more than one `README` file is found, P4 Code Review displays the first one it finds based on the order above.
- The Project overview pages support Parsedown, see the [Parsedown demo](#).

MAIN MENU

JPlugin

Overview

Activity

Reviews

Files

Commits

Settings

Projects/Jplugin

About

A Java plugin for continuous integration.

3
MEMBERS

1
FOLLOWER

2
BRANCHES

OWNERS

MODERATORS

MEMBERS

FOLLOWERS

BRANCHES

MAIN

CANDIDATE

## P4 Plugin

Jenkins plugin for a Perforce Helix Versioning Engine (P4D).

### Contents

- [Release notes](#)
- [Setup guide](#)
- [Notes page](#)
- [Jenkins page](#)

### Requirements

- Jenkins 1.642.3 or greater.
- Helix Versioning Engine 2012.1 or greater.
- Minimum Perforce Protection of `open` for the Jenkins user.
- Review Build feature requires Swarm 2014.2 or greater.

### Install

- Open Jenkins in a browser; e.g. [http://jenkins\\_host:8080](http://jenkins_host:8080)
- Browse to 'Manage Jenkins' --> 'Manage Plugins' and Select the 'Available' tab.
- Find the 'P4 Plugin' or use the Filter if needed
- Check the box and press the 'Install without restart' button

If you are unable to find the plugin, you may need to refresh the Update site.

- Select the 'Advanced' tab (under 'Manage Plugins')
- Press the 'Check now' button at the bottom of the page.
- When 'Done' go back to the update centre and try again.

### Building

To build the plugin and run the tests use the following:

399

## 5 | Code reviews

A code review is a process in which other developers can see your code and provide feedback that can suggest ways to improve the code's structure, performance, maintainability, and interaction with other code.

### Benefits

Some of the benefits of code review are:

- Enforcing coding standards: code reviews can catch code that does not meet your team's coding standards. This improves the readability and consistency of your codebase.
- Knowledge and experience sharing: your team can help you learn to code better. This is particularly useful for developers new to the team.
- Early defect detection: small errors can be caught before they become problems later on.
- Code sharing: code reviews spread knowledge of the current codebase, which helps both with maintaining a mental model of the overall project, as well as defending against developer absences.
- Better personal review: knowing that someone might catch a simple coding error often increases the review developers perform of their own code.

P4 Code Review attempts to provide these benefits without adding onerous overhead for developers.

### Facilities

P4 Code Review provides the following code review facilities. In the list, the term *author* refers to the person who creates a change to be reviewed, *reviewer* refers to any authenticated P4 Code Review user performing code review tasks, and *required reviewer* refers to a reviewer whose up-vote is required before a review can be approved.

- Authors can request reviews, and can designate reviewers and required reviewers.
- Reviewers can start a code review on existing changes.
- Reviewers can add themselves to a review to indicate that they are participating in the review and sharing responsibility for the review.
- Reviewers can add comments to a changelist, to a specific file in a changelist, or to a specific line in a file, using [Markdown](#) text.
- Reviewers can vote on a review, to indicate their approval or disapproval.
- Required reviewers can prevent a review from becoming approved until they up-vote the review.
- Project branches with assigned moderators limit review approval to one of the moderators.
- Reviews spanning multiple project branches with assigned moderators, limit review approval to one moderator from any one of the branches by default.

**Note:**

P4 Code Review can be configured to require that one moderator from each branch must approve the review. For more information about moderator behavior, see ["Moderators" on page 455](#).

- Reviewers can mark changes as needing revision, approved, rejected, or to be archived for future consideration.
- Reviewers can commit approved changes if necessary.

## Workflow

By default, P4 Code Review's code review workflow is basically advisory in nature with very few restrictions imposed on the code review workflow. However, P4 Code Review does provide a number of mechanisms to structure or restrict code review workflows, such as:

- A required reviewer, which is any user designated by a review author, project member or moderator, or is any authenticated user that joins a review and makes their vote required, can prevent reviews from being approved until they up-vote the review. See ["Required reviewers" on page 457](#) for details.
- Branch moderators, when configured for one or more branches in a project, prevent reviews from being approved (or rejected) without their involvement. For more information about moderators, see ["Moderators" on page 500](#).
- Administrators can optionally configure P4 Code Review to prevent reviews with open tasks being approved or, approved and committed, see ["Disable approve for reviews with open tasks" on page 676](#).
- **Workflow:** on by default:
  - Global workflow rules can be configured to enforce a minimum code review workflow on the entire P4 Server. For more information on global workflows, see ["Workflow basics" on page 512](#).
  - Global workflow rules can be configured to enforce a minimum code review workflow on projects, and branches that don't have an associated workflow. For more information on global workflows, see ["Workflow basics" on page 512](#).
  - Individual workflows can be configured by users and can be associated with projects, and branches to enforce that code review workflow on them. For more information on workflow, see ["Workflow basics" on page 512](#).

Agile development teams should find sufficient capability within P4 Code Review to make code reviews a regular part of their workflow. P4 Code Review's development team has been using it regularly during development of P4 Code Review. If you have ideas and suggestions for improvement, please [contact us](#).

## Models

There are two code review models: pre-commit and post-commit. Which model you use for code reviews with P4 Code Review is up to you.

## Pre-commit model

The pre-commit code review model allows developers to review code before it is committed. This allows reviewers to look at code early and identify potential issues before they reach production environments.

P4 Code Review uses the shelving feature in P4 Server for a pre-commit code reviews. For more information on shelving, see [Shelve Changelists](#) in [P4 CLI Documentation](#).

Use P4 Code Review to enforce code review procedures and ensure that code is approved before it is committed.

## Post-commit model

In the post-commit code review model, code is committed to the P4 Server before a code review is initiated. This approach does not involve shelving and avoids blocking contributors, as developers are not required to wait for an approved review before committing their changes.

However, this model can make it more difficult to catch and correct errors, especially in environments with continuous integration (CI), since issues may not be identified until after code has been merged. As a result, there is a greater risk that new work may be built upon faulty or unreviewed code, which can complicate debugging and reduce overall code quality.

## Internal representation

### P4 Code Review-managed changelists

**Note:**

**P4 Server 2020.2 and later:** any new file being shelved that has the same content as an existing shelved file refers to the existing archive file instead of creating a duplicate archive file. No P4 Server or P4 Code Review configuration is required for this feature.

This P4 Server feature automatically reduces the space required for the P4 Code Review-managed shelved review changelists. P4 Code Review creates these changelists for its own internal use. P4 Server only updates new shelves, it does not retrospectively update your existing shelves.

A code review consists of one or more shelved changelists that P4 Code Review manages. A shelved changelist is a pending changelist that has a snapshot of its files on a shelf associated with the changelist.

When a review is started, P4 Code Review creates a new changelist that becomes the review changelist. What happens afterwards varies:

- If the review contains uncommitted work (the pre-commit model), P4 Code Review copies the shelved files from the user's changelist that initiated the review into the review's changelist.
- Any time that a user's changelist associated with the review has its shelved files updated, P4 Code Review copies the shelved files into its review changelist and creates an archive changelist. An archive changelist is no different from any other pending changelist with shelved files, but it allows P4 Code Review to provide versioning and diffs within a review.

- If the head version of a review is committed (the post-commit model), the review's changelist is emptied of files.

The review's changelist is never actually committed; this allows the review to be opened later with additional shelved changes.

### Important:

**P4 Code Review's managed review changelists should only be deleted if you are uninstalling P4 Code Review.**

P4 Code Review's review changelists maintain the history of a review and all of its feedback. The deletion of a P4 Code Review shelved changelist causes instability and potentially data loss, and represents a scenario that can be very challenging to recover from, even with the engagement of Perforce consultants.

You can display a list of all of the P4 Code Review-managed changelists using the **p4 changelists** command:

```
$ p4 changelists -u swarm
```

```
Change 1212285 on 2015/07/31 by swarm@swarm-96017af4-5615-9819-7af1-6fc1fa537214 *pending* 'Add requirements and instructions'
```

```
Change 1212284 on 2015/07/31 by swarm@swarm-96017af4-5615-9819-7af1-6fc1fa537214 *pending* 'Add requirements and instructions'
```

```
...
```

*swarm* is the userid with *admin*-level privileges within the P4 Server that P4 Code Review is configured to use. Use the appropriate userid when you run the **p4 changelists** command.

## P4 Code Review clients

Whenever P4 Code Review creates a changelist for a review, it uses a client workspace (or clients) associated with the configured P4 Server userid that has *admin* privileges. Whenever a user commits a change via P4 Code Review's user interface, P4 Code Review uses a client associated with that user.

These clients are named `swarm-{uuid}`, for example `swarm-5ad4a9c0-06e7-20eb-897f-cbd4cc934295`. To learn more about clients, see [P4 Server as a version control implementation](#) in the [P4 Overview](#) Guide.

**Tip:** A P4 Code Review-managed workspace is the same as a P4 Code Review client. These two terms refer to the same worker.

The client that P4 Code Review creates and uses live in the `SWARM_ROOT/data/clients` folder.

Inside the clients folder, P4 Code Review maintains a user-specific folder that contains any client folders that may be required. Each user-specific folder is named by converting their P4 Server userid into hexadecimal to avoid any characters that would be problematic in the filesystem, such as slashes, accents, UTF-8 characters, etc. For example, the folder for the user `steve.russell` would be named `73746576652e72757373656c6c`.

Within the user-specific folder are the folders that become the root of each client. Each of these folders is named with a globally-unique identifier (GUID) prefixed with `swarm-`, for example `swarm-438d482b-f107-9a35-c06c-86ac68136b00`. Accompanying each folder is a lock file with the same name plus the `.lock` extension. Finally, the user-specific clients folder contains a management lock file called `manage.lock`.

Here is an example of the folder structure:

```

SWARM_ROOT/
 data/
 clients/
 73746576652e72757373656c6c/
 manage.lock
 swarm-438d482b-f107-9a35-c06c-86ac68136b00/
 swarm-438d482b-f107-9a35-c06c-86ac68136b00.lock
 swarm-8388362a-233d-0cb9-3e90-895eaaa99f6c/
 swarm-8388362a-233d-0cb9-3e90-895eaaa99f6c.lock
 7061756c612e626f796c65/
 manage.lock
 swarm-da7de4b4-0ecb-12c8-1b35-f3e32bb18033/
 swarm-da7de4b4-0ecb-12c8-1b35-f3e32bb18033.lock

```

Here are the steps P4 Code Review takes when it needs to use a client:

1. Convert the current connection's userid to hexadecimal.
2. Check to see whether a user-specific folder exists within `SWARM_ROOT/data/clients`; if not, create the folder.
3. Within the user-specific folder, loop over any existing client folders and attempt to lock each in turn:

If a lock is acquired skip to the next step. Otherwise, perform the following procedure.

**Create client procedure:**

- a. Check if the max number of clients for the current user has been reached:
    - If so, wait a short amount of time (50 milliseconds), and start [step 3](#) again.
    - If not, proceed to the next step.
  - b. Take control of a lock on `manage.lock`.
  - c. Check if the max number of clients for the current user has been reached:
    - If so, release the `manage.lock`, wait a short amount of time (50 milliseconds), and start [step 3](#) again.
    - If not, proceed to the next step.
  - d. Create a new client folder using a GUID-based filename, and take a lock on the folder.
  - e. Release the `manage.lock` lock.
4. Perform the necessary file operations using the locked client folder.



5. Revert the file content within the client folder to avoid having constantly growing disk space use.

**Note:**

There may occasionally be stray files left; P4 Code Review is not aggressive about cleaning up.

6. P4 Code Review releases the lock on the client folder.

Most users should only require 1-2 clients, and those are only required if they commit from P4 Code Review. The *admin* user that P4 Code Review is configured to use should only use one client per configured worker.

By default, the number of clients that could be active at any given instant is two times the number of configured workers. Since the default worker count is three, P4 Code Review would use at most six clients simultaneously.

If the client limit is reached, further file processing is blocked until a client becomes available. Potentially, this means that users could encounter timeouts. Configuring P4 Code Review to use more workers could solve that issue.

#### Removal considerations when deleting P4 Code Review clients

**Caution:** We advise that you do not delete P4 Code Review clients.

The clients are created to do work within P4 Code Review, such as committing reviews. These reviews are owned by the P4 Code Review clients. If you delete a client, all the reviews it owns will become inaccessible.

This applies to anything owned by a P4 Code Review client.

There are a few considerations that should be assessed prior to removal:

- Ideally, you should stop the web server (taking P4 Code Review out of service) before removing a client from the P4 Code Review server; this eliminates the risk of removing a client that is in use.

If you do not stop the web server first, P4 Code Review may encounter an error during a commit.

- Removal of a client folder does not remove the client spec from the P4 Server. Unless the client spec is removed, that client effectively becomes orphaned. Orphaned clients are, of themselves, not a big concern as the storage and performance impact is negligible.
- Removal of a client's corresponding spec in the P4 Server can be done. However, **you should never remove a client spec that has associated shelved files.**

Usually, the only client specs that should have associated shelved files belong to the *admin* account that P4 Code Review is configured to use. All other clients that may exist for other users are primarily used for committing changes, and so should not have shelved files associated.

- Anything owned by the P4 Code Review client will become inaccessible to users if the client is deleted. This includes reviews and changelists.

## Reviews list

The code review list helps you keep track of code reviews that:

- Have been requested and are awaiting review
- Are underway
- Have been accepted, rejected, or archived

To see all available reviews, click **Reviews** in the menu.

The **Reviews** page lists open and closed reviews for all projects in the P4 Server.

| Description                                                                                                                                         | Created        | State | Tests | Complexity | Comments | Votes   |
|-----------------------------------------------------------------------------------------------------------------------------------------------------|----------------|-------|-------|------------|----------|---------|
| Added Windows to list of build platforms.<br>Allison Clayborne requested a post-commit review 396, 31 minutes ago for jpluginmain                   | 31 minutes ago | 🔄     | 🔴     | 137        | 1        | 👍 0 👎 0 |
| Tidying up some of the code base.<br>Allison Clayborne requested a pre-commit review 391, 3 days ago for jpluginmain                                | 3 days ago     | 🔄     | 🔴     | 18         | 1        | 👍 0 👎 0 |
| Added some state exception checking.<br>Jack Boone requested a pre-commit review 388, 4 days ago for jpluginmain                                    | 4 days ago     | 🔄     | 🟢     | 6          | 2        | 👍 0 👎 2 |
| Update to the 12th entry point, which is defined to measure the latency<br>Jack Boone requested a pre-commit review 385, 4 days ago for makermaster | 4 days ago     | 🔄     | 🔴     | 1462       | 0        | 👍 1 👎 0 |
| Update the README to use markdown rather than the old format.<br>Jack Boone requested a pre-commit review 382, 4 days ago for blue-bookmain         | 4 days ago     | 🔄     | 🔴     | 2          | 0        | 👍 0 👎 0 |

The **Opened** and **Closed** tabs display:

- **Opened** tab: displays a list of all code reviews that have started, are being reviewed, are awaiting revisions, or need to be committed.
- **Closed** tab: displays a list of all code reviews that have been approved and committed, rejected, or archived.

### Tip:

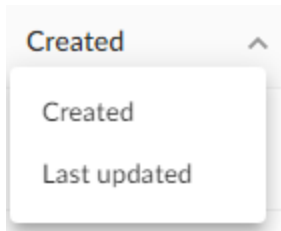
- The **Opened** and **Closed** tabs display the number of reviews in each tab. Initially this can be an over estimate which P4 Code Review dynamically corrects as more information becomes available to it.
- The **Closed** tab initially displays a - character. The - is replaced by a number when you navigate to the **Closed** tab.

Each review displays the following information:

| Description                                                                                                                       | Created        | State | Tests | Complexity | Comments | Votes   |
|-----------------------------------------------------------------------------------------------------------------------------------|----------------|-------|-------|------------|----------|---------|
| Added Windows to list of build platforms.<br>Allison Clayborne requested a post-commit review 396, 31 minutes ago for jpluginmain | 31 minutes ago | 🔄     | 🔴     | 137        | 0        | 👍 0 👎 0 |

- **Avatar** displays the avatar of the review author, click to view the profile of the author
- **Description** contains:
  - first line of the review description, click to open the review. If the review description is too long it is truncated, click on the ellipsis **...** to expand it in the list page.
  - review author, click to view the profile of the author

- review type, pre-commit or post-commit
- review number and version, click to open the review
- review creation date and time
- review project branches, click to open the project
- **Created or Last activity:** when the review was created or when the review was last updated depending on the **Result order** button setting.






To change the sort order, click the **Result order** button and select **Created** or **Last activity** from the dropdown menu:

- **Created:** Reviews sorted by when they were created
- **Last activity:** Reviews sorted by when they were last updated

**Note:**

- If the **Result order** button is not displayed, reviews are sorted by either **Created** or **Activity**, as set by your P4 Code Review administrator, see ["Reviews filter" on page 673](#) for details.
- **Result order** button display is a global setting controlled by the P4 Code Review administrator. See ["Reviews filter" on page 673](#) for details.
- When review results are older than 24 hours they are displayed in numerical order within each day.

- **State:** shows the current review state
- **Tests:** shows the test suite state for the review, either tests in progress , tests passed , or tests failed .
- **Complexity:** a traffic light icon and number shows the relative complexity of the review and the total number of lines changed in the review. [Complexity can be configured](#) by your P4 Code Review administrator but, by default:
  - **Red:**  $\geq 300$  changes
  - **Amber:**  $< 300$  and  $> 30$  changes
  - **Green:**  $\leq 30$  changes

**Tip:**

- Review complexity is only calculated for a review when the review is updated and the file content has changed.

- Review complexity is only stored for the current version of a review.

Hover over the complexity icon to display more detailed information:



- **Comments:** displays the number of open (non-archived) comments that are associated with the review. The icon is filled if there are comments on the review.
- **Votes:** displays the number of up votes and down votes for the review. The appropriate vote icon is filled if you have voted on the review, vote icons with only an outline show votes by others.

### Note:

Hover your mouse over any of the icons to see tooltips.

"Projects" on page 360 and "Groups" on page 352 have their own review lists that display reviews created by their members.

## Review filters

### Tip:

P4 Code Review updates the URL in your browser to reflect the filter options you have selected. This makes it easy to bookmark or share your review list filter settings with the URL. If you share a URL with **Comment** or **Vote** filters selected, those filters are applied to the user running the URL. This means that their reviews list page will contain different reviews to your reviews list page.



The following filters are available (from left to right):

- **Branch** (only available on the project "[Reviews page](#)" on page 363): a dropdown menu that lets you filter which reviews to display based on the current project branches:



- **Select branch:** all reviews for all of the selected branches are displayed. To select multiple branches, click the branches you want to filter by.

- **Branch search:** an auto-complete search field that allows you to choose one of the branches within the current project. Once specified, only reviews for the selected project branch are displayed.
- **Project dropdown:** filter by the project the review is part of:

Project 

- **My projects:** displays all reviews for all of the projects you are participating in, as a member, owner, moderator, or follower.
- **Select project:** displays all reviews for all of the selected projects. To select multiple projects, click the projects you want to filter by.
- **Project search:** an auto-complete search field that allows you to choose one of the projects defined in P4 Server. Once specified, only reviews for the selected project are displayed.

**Tip:**

P4 Code Review creates a projectid based on the project name whenever a project is created. For example, a project called **Test Data** is given a projectid of test-data. If the project name changes, the projectid does not change.

If your search finds a match in a projectid but not in the associated project name, the project name for that projectid is returned in the search results. For example, if the project name was changed from **Test Data** to **Sample Data**, a search for *test* will return the **Sample Data** project in the search results.

- **Role dropdown:** filter reviews based on user involvement, options are:

Role 

- **All reviews:** displays all reviews.
- **Reviews I've authored:** displays reviews that you have authored.
- **Reviews I'm participating in:** displays reviews that you are a reviewer of, but not an author of.
- **Reviews I'm an individual reviewer of:** displays reviews that you are an individual reviewer of, but not a group reviewer of, or an author of.
- **Review I've authored or I'm participating In:** displays reviews that you have authored, or are a reviewer of.
- **Authored by:** an auto-complete search field that allows you to choose one of the user accounts defined in the P4 Server. Once specified, only reviews authored by the user are displayed.

When you select one of the available options, the list of options updates to match the currently selected filter, and the **Role** dropdown indicates the current filter: **All**, **Author**, **Participant**, **Author or Participant**, **Individual Reviewer**, or **user**.

- **Reviewers** buttons, (**Opened** tab only):



- **Has reviewers:** displays reviews that have one or more reviewers.
- **No reviewers:** displays reviews that have no reviewers.
- **States** buttons, (**Opened** tab):



- **Needs Review:** displays reviews that need to be reviewed.
- **Needs Revision:** displays reviews that have been reviewed, but need further revisions before the review can be accepted.
- **Approved:** displays reviews that have been approved, and should be committed.
- **States** buttons, (**Closed** tab):



- **Approved:** displays reviews that have been approved and committed.
- **Rejected:** displays reviews that have been rejected.
- **Archived:** displays reviews that have been archived.
- **Tests** buttons (when automated tests are enabled for the associated project):



- **Tests passed** displays reviews that have passed tests.
- **Tests failed:** displays reviews that have failed tests.
- **Comments** buttons:



- **I have commented on:** displays reviews that you have commented on.
- **I have not commented on:** displays reviews that you are participating in but have not commented on

**Note:**

Filters for commenting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after commenting on it, then this review will not be included in the list.

- **Votes** buttons:



- **I have voted up:** displays reviews that you have voted up.
- **I have voted down:** displays reviews that you have voted down.
- **I have not voted on:** displays reviews that you are participating in but have not voted on.

**Note:**

Filters for voting only apply to reviews which you are a participant of. Commenting on or voting on a review will automatically add you as a participant. If you leave the review after voting on it, then this review will not be included in the list.

- **Clear all:** click to clear all of the filters.
- **Search for a review:** search for a partial match of review description content and an exact match for *reviewid*, *userid*, and *projectid*. The search filter boxes on the **Opened** and **Closed** tabs are independent.

## Review display

**Tip:**

The review page is described in this documentation.

The old review page is now referred to as the classic review page. To use the classic review page, use the **Classic view** toggle switch at the top of the review page.

The P4 Code Review Classic view option will be removed in a future P4 Code Review release.

Help for the classic review page is available in the P4 Code Review 2021.2 documentation. See [Review display](#).

During a code review, reviewers spend most of their time using the review interface.

By default, when you open the Review page for the first time, the file list panel and the information panel are hidden. Use the **Show more context** buttons to view the file list panel and the information panel or use the keyboard shortcuts, **Shift + Alt + L** to view the file list and **Shift + Alt + S** to view the Information panel.

For more information about the show more context buttons, see ["File diff panel" on page 436](#).

The screenshot displays the 'Reviews/644' page in a 'Classic view'. At the top, it indicates that Allison Clayborne requested a pre-commit review 644, about a year ago, for Version 2 of 2 for the 'mercury-dev' branch, last updated 7 months ago. The review is currently in a 'Needs review' state, with a 'Change state' button and a 'Needs review' badge. Below this, a description of the review is provided: 'Worker actor for the breakpoint scanner so that we can improve features.' There are 'Comments (0)' and 'Jobs (0)' listed. The interface includes tabs for 'Files', 'Comments', and 'Activity', with 'Files' being the active tab. A dropdown menu shows the 'Base version for this review' as '#2 by Allison Clayborne' and 'Shelved in 644 7 months ago'. The main content area shows a list of files to be reviewed, including '/db.c#1', '/est/ risus/ jira/ xgen/ rml.c', '/magis/ scm/ vendor\_x86\_64/ page32.c', '/metadata.c', '/stub.c', and '/users.c#1'. The sidebar on the right shows 'Blocking Approval (1)', 'Reviewers' (Claire Brevia, Jack Boone, Steve Russell), 'Tasks' (0 out of 6 read), and 'Tests' (Passed: 1, Failed: 1, Manual: 0, In progress: 0).

P4 Code Review supports stream specs in your workspace using the [Private editing of streams](#) feature. If a changelist or review contains a stream spec, it will be displayed first in **Files** with the prefix **stream: //**, for example: `stream://MyStreamDepotName/MyStreamSpecLocationName`. A changelist/review can only contain one stream spec.

The review interface is very similar to the [changelist interface](#); and provides largely the same functionality, but has several notable differences that are described in the following sections.

**Jump directly to a specific area of the review page using the following links:**

- ["Review summary" on the next page](#)
- ["Review state" on the next page](#)
- ["Vote buttons" on page 414](#)
- ["Change state button" on page 415](#)
- ["Review actions button" on page 415](#)
  - ["Add change" on page 415](#)
  - ["Download zip" on page 417](#)
  - ["Change the review author" on page 418](#)
  - ["Obliterate a review" on page 418](#)
  - ["Join or leave a review" on page 419](#)
  - ["Mark all comments for the review as read" on page 419](#)
  - ["Mark all comments for the review as unread" on page 420](#)
  - ["Refresh projects" on page 420](#)
  - ["Disable notifications" on page 420](#)
  - ["Deployment status" on page 421](#)



- "Review description" on page 421
  - "Edit description" on page 421
  - "Description comments" on page 421
  - "Jobs" on page 422
- "Information panel" on page 423
  - "Blocking Approval or Blocking Commit" on page 423
  - "Reviewers" on page 424
  - "Tasks" on page 427
  - "Tests" on page 428
- "Files tab" on page 430
  - "Select review versions to view" on page 431
  - "File list header" on page 431
  - "File list" on page 434
  - "File stack" on page 435
  - "File content panel header" on page 435
  - "File diff panel" on page 436
- "Comments tab" on page 441
- "Activity tab" on page 442
- "Known limitations with pagination" on page 443

## Review summary

The Review ID is displayed at the top of the review page. A summary of the review is displayed below the review ID, and contains:

- **Review author** avatar and name: The avatar and name of the review author. Hover over the avatar to see the name of the review author and hover over the name to see the ID of the review author. Click on the avatar or name to go to the profile of the review author. See ["Viewing another user's profile" on page 350](#).
- **Review type**: Displays the review type, reviews can be [pre-commit](#) or [post-commit](#).
- **Review ID**: The unique number used to identify the review.
- **Review raised**: When the review was requested.
- **Version**: The version of the review being viewed and the total number of review versions.
- **Project branch**: The project branch the review files are in, click to go to the project page.
- **Last updated**: When the review was last updated.

## Review state

The state of the review is displayed in the review description header. A review can be in one of the following states:

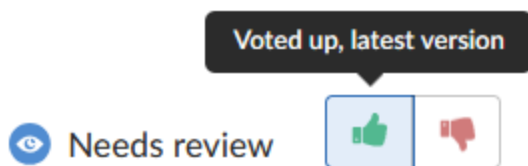
- **Needs review:** The review has started and the changes need to be reviewed.
- **Needs revision:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
- **Approved:** The review has been approved. The changes may need to be committed.
- **Rejected:** The review has been completed. The changes are undesirable and should not be committed.
- **Archived:** The review has been completed for now but it is not rejected, or approved. The review has been filed away in case it is needed in the future.

For information about review states, see ["States" on page 464](#).

## Vote buttons

The vote buttons are in the review description header. Click a vote button to vote the review up or down. If you have already voted on the review, the vote button you clicked is highlighted. Clicking the button again, clears your vote. If you are not a member of a review, voting on the review adds you as a reviewer.

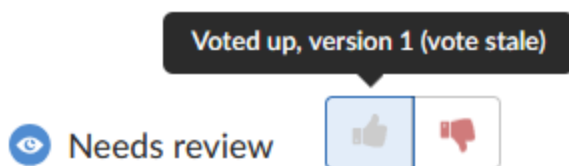
Hover over the vote button to see which version of the review you have voted on.



## Stale votes

When a review is updated, if the review's list of files, file content, or file-types changes, any votes cast on the review become *stale*. The vote counts are reset, and the vote indicators become muted.

If you have voted on a review that has been updated, the tooltip will display the review version that you have voted on and if that vote is now stale.



If you hover your mouse over a reviewer with a stale vote, a tooltip appears displaying the *userid*, how they voted, and on which version of the review; each version is represented as a point on the ["Select review versions to view" on page 431](#).

You can only vote on the latest version of a review. If you try and vote on an older version of a review, you are prompted by P4 Code Review to refresh the page to get the latest version of the review.


## Change state button

The **Change state** button is also used to change the state of the review. Click to select a new state from the dropdown menu. State change options are only displayed if you are authorized to make the state change:

- **Needs revision:** Select to request changes to the files in the review.
- **Needs review:** Select to request further review of the changes.
- **Approve** (only available if the voting requirements for the review are satisfied. For information on voting requirements, see ["Required reviewers" on page 457](#)): select to approve the review.
- **Commit** (only available for [pre-commit](#) reviews that have been approved): Select to commit the review.
- **Approve and commit** (only available for unapproved [pre-commit](#) reviews when the voting requirements for the review are satisfied. For information on voting requirements, see ["Required reviewers" on page 457](#).): Select to approve and commit the review in a single step. See ["Approve and commit" on page 469](#).
- **Reject:** Select to reject the review.
- **Archive:** Select to archive the review.

For information about changing the state of a review, see ["States" on page 464](#).

## Review actions button


The **Review actions**  button gives you access to more actions you can perform on the review. Not all of these actions are available to you, the actions available to you depend on your role on the review, the review type, and what is enabled on your P4 Code Review system:

- ["Add change" below](#)
- ["Download zip" on page 417](#)
- ["Change the review author" on page 418](#)
- ["Obliterate a review" on page 418](#)
- ["Join or leave a review" on page 419](#)
- ["Mark all comments for the review as read" on page 419](#)
- ["Mark all comments for the review as unread" on page 420](#)
- ["Refresh projects" on page 420](#)
- ["Disable notifications" on page 420](#)
- ["Deployment status" on page 421](#)

## Add change

Once a review has been started you can add a changelist to the review. It can be useful to add changelists to an existing review. For example, if follow up changes are made to files in a review or if you need to group a number of changelists under a single review.

By default, for a pre-commit or a post-commit review, when a changelist is updated, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a pending changelist" on page 452](#).

The **Add change** option in the **Review actions**  button is used to add a changelist to an existing review. The option is not available if the review is in a state that is protected from change by the review workflow rules. For details of the **On update of a review in an end state** rule, see ["Workflow rules" on page 510](#).

**Tip:**

You can also add a changelist to a review directly from the changelist description. See ["Review creation and modification outside of P4 Code Review" on page 462](#).

The options available for the **Add change** option depend on whether the review is pre-commit or post-commit:

- **Pre-commit reviews:**

- **Append pending changelist:** when you add a pending changelist to a review, the files in the changelist are appended to the existing files in the review. See ["Append a pending changelist to a review" on page 451](#).

**Note:**

A review can only contain 1 stream spec. If you append a changelist with a stream spec to a review that already contains a stream spec, the spec in the changelist replaces the original one in the review. If it is a different spec from the original spec in the review, P4 Code Review cannot display the diff between them and displays **File content unchanged**.

- **Replace with pending changelist:** when you add a pending changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a pending changelist" on page 452](#).
- **Replace with committed changelist:** when you add a committed changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a committed changelist" on page 453](#).

**Note:**

If you replace a pre-commit review with a committed changelist, the new version of the review will be a post-commit review.

- **Post-commit reviews:**

- **Replace with pending changelist:** when you add a pending changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a pending changelist" on page 452](#).

**Note:**

If you replace a post-commit review with a pending changelist, the new version of the review will be a pre-commit review.


- **Replace with committed changelist:** when you add a committed changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a committed changelist" on page 453](#).

**Tip:**

When the content of a review is changed, P4 Code Review checks to see which branches are in the new version of the review:

- **If a new branch was added to the review:**
  - Default reviewers on the new branch are added to the review.
  - Moderators from the added branch become moderators for the review alongside the existing moderators.
  - **Only if workflow is enabled:** if the new branch is associated with a workflow, the [workflow is merged](#) with the existing workflow. The most restrictive workflow is used for the review.
- **If a branch is no longer part of the review:**
  - Reviewers for the review are not changed.
  - Moderators from the removed branch no longer moderate the review.
  - **Only if workflow is enabled:** if the branch was associated with a workflow, the branch workflow is removed from the review.

## Download zip

The **Download zip** option is available from the **Review actions**  button and is used to download a ZIP archive containing all of the files in the review. The file revisions of the downloaded files are the file revisions in the most recent review version selected in the review version selector. See ["Select review versions to view" on page 431](#).

**Note:**

The **Download zip** option is not displayed if the zip command-line tool is not installed on the P4 Code Review server. For information about installing, and configuring the zip command-line tool, see [Zip archive](#).

When you select the **Download zip** option, P4 Code Review performs the following steps:

1. Scans the files/folders:
  - Checks that you have permission to access their contents, according to the P4 Server protections.
  - Checks that the total file size is small enough to be processed by P4 Code Review.
2. Syncs the file contents to the P4 Code Review server from the P4 Server.

3. Creates the ZIP archive by compressing the file content.
4. Starts a download of the generated ZIP archive.

**Note:**

- You might not see all of the above steps; P4 Code Review caches the resulting ZIP archives so that repeated requests to the same files/folders can skip the sync and compress steps whenever possible.
- If an error occurs while scanning, syncing, or compressing, P4 Code Review indicates the error.


## Change the review author

**Note:**

By default you cannot change the author of a review, this option must be enabled by your P4 Code Review administrator. See ["Allow author change" on page 675](#) for details.

If the author of a review is no longer available, or ownership of a review is passed to a different developer, it is useful to be able to change the author of that review.

### To change the author of a review:

1. Click the **Review actions**  button.
2. Select **Change author**.
3. Select the new review author from the user list.
4. Click **Save** to change author of the review.

## Obliterate a review

**Note:**

- By default, you must be a user with *admin* or *super* user rights to obliterate a review.
- **Optional:** P4 Code Review can be configured to allow users to obliterate reviews that they have authored. This can be configured by your P4 Code Review administrator. See ["Allow author obliterate review" on page 675](#).


Obliterate is used to permanently delete reviews that have been created by mistake. For instance, if a review is associated with the wrong changelist, or a review contains sensitive information that should not be openly available.

For information on what happens to a review when it is obliterated, see ["When you obliterate a review" on page 657](#).

**Important:**

Obliterate must be used with care, the review and all of its associated metadata are permanently deleted. An obliterated review cannot be reinstated, not even by Perforce Support.


**To obliterate a review:**

1. Navigate to the review.
2. Click the **Review actions**  button and select **Obliterate Review**.
3. Click **Yes** on the confirmation dialog to complete the obliterate action.
4. The review is obliterated.

## Join or leave a review

The option available depends on whether you are a reviewer on the review or not.


**Join a review**

1. Open the review you want to join.
2. Click the **Review actions**  button.
3. Select **Join review** from the dropdown menu. Alternatively, you can **Vote Up** or **Vote Down** the review.

Your name is added to the **Reviewers** list in the **Information panel**, and you are now a reviewer.

**Leave a review****Tip:**

You cannot leave the review if you are a retained default reviewer on the review or you are a member of a group that is a reviewer on the review


1. Log in, if you have not already done so.
2. Open the review you want to leave.
3. Click the **Review actions**  button.
4. Select **Leave review** from the dropdown menu.

Your name is removed from the **Reviewers** list in the **Information panel**, and you are no longer a reviewer.

## Mark all comments for the review as read

When you mark a comments as read, the comments are rolled up into single lines to save space. The read flag is remembered independently for each user.


**Tip:**  
**Mark all comments read** will mark all the comments as read including the description comments.

1. Click the **Review actions**  button.
2. Select **Mark all comments read** from the dropdown menu.
3. All of the comments in the review are rolled up into single lines to save space.

|                                                                                   |                                                                                                                                                 |                                                                                                                                                                              |
|-----------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|  | Steve Russell commented on review 644 (version 1) <a href="#">//projects/mercury/dev/src/api/db.c, line 51</a> - 7 months ago                   | read                                                                                      |
|  | Allison Clayborne commented on review 644 (version 1) <a href="#">//projects/mercury/dev/src/api/db.c, line 105</a> - 7 months ago              | read                                                                                      |
|  | Allison Clayborne commented on review 644 (version ) - 10 days ago                                                                              | read                                                                                      |
|  | Steve Russell commented on review 644 (version 2) <a href="#">//projects/mercury/dev/src/api/db.c, line 67</a> - 3 days ago (edited 3 days ago) | read                                                                                      |
|  | Jack Boone commented on review 644 (version 2) - about 15 hours ago                                                                             |  read  |

## Mark all comments for the review as unread

**Tip:**  
**Mark all comments unread** will expand the content of all the comments including the description comments.

1. Click the **Review actions**  button.
2. Select **Mark all comments unread** from the dropdown menu.
3. The content of all of the comments in the review are expanded.

## Refresh projects

When a review is created or updated, P4 Code Review checks to see what projects the review files are associated with and links the review to the projects it finds. If at a later date a new project is created or a new branch is added to a project, P4 Code Review does not check to see if existing reviews are associated with new projects or project branches.

The **Refresh projects** option is used to check if the review files are associated with any projects created or updated after the review was last updated. If any projects are found that are not already associated with the review, the review is linked to them.

## Disable notifications



When you become a review participant, by joining the review or being [@mentioned](#) in a comment or in the review's description, you receive notifications for any events associated with the review. If you find that the notifications become more of a burden than benefit and you wish to continue being a review participant, you can disable notifications:

1. Click your avatar in the reviewers area to display your reviewer options.
2. Click **Disable Notifications**.



Once notifications are disabled, you no longer receive notifications. However, if you are [@mentioned](#) in a subsequent review comment, you do receive a notification for that comment; regular notifications remain disabled. This approach ensures that you don't miss anything that other reviewers or the review author deems important.

## Deployment status

When [automated deployment](#) has been configured for a project, the deployment success  or failure  is indicated in the review's heading. If your deployment program can provide a URL that provides details of the deployment, the indicator becomes linked; click the indicator to see the deployment results.

## Review description

The review description is automatically copied from the changelist description when the review is created. When the review description size exceeds the 25% of the page, only the initial part of the review description is displayed. Select the **Expand description** button to view the full review description and select the **Collapse description** button to hide a part of the review description.

## Edit description

Click the **Edit Description**  link in the review description to update the description to reflect any updates have been made during the review.

### Tip:

- Markdown content is displayed in the review description, but Markdown support is limited to prevent execution of raw HTML and JavaScript content. For information about Markdown, see "[Markdown in comments and review descriptions](#)" on [page 397](#)
- If you use Markdown styles in your review description, P4 Code Review renders them when you update the description.

### Update pending changelist checkbox

Only available if you are editing the description of a pre-commit review and you are the original author of the changelist that created the review.

Select the **Update pending changelist** checkbox in the **Edit Description** dialog to also apply your review description changes to the original changelist description.

## Description comments

You can add comments to the description of the review.

1. Click **Comments (n)** (where **n** is the number of comments that already exist).
2. Click **Add a comment**.

3. Add your comment in the text area.
4. Click **Post**.

**Tip:** Use the keyboard shortcut **Ctrl + Enter** to submit the comment or form.

To close an empty comment text box, click outside the comment text box or select **Esc** on your keyboard.

**Tip:**  
To hide the description comments, click **Comments n** (where **n** is the number of description comments that exist). To display the comments again, click **Comments n**.

## Jobs

A list of jobs that this change *fixes*, if any.

### Adding a job to the review

P4 Code Review does not provide the ability to create new jobs in the P4 Server, but jobs can be added to changelists or reviews:

1. Click **Jobs (n)** (where **n** is the number of jobs that are already part of the review).
2. Click **Add a job**.
3. Scroll through the available jobs, or enter job search criteria to search available jobs.  
For more information on job search criteria, see [Jobs](#) in [P4 CLI Documentation](#).
4. If you find the job you want to add, click it to highlight it and then click **Select** to add it.

**Note:**

If you attempt to add a job to a review that affects a single project, P4 Code Review applies the project's job view filter to display only jobs that affect the project. It is not currently possible to expand the filter to include jobs outside of the project.

### Unlinking a job from the review

P4 Code Review does not provide the ability to delete jobs from the P4 Server, but jobs can be unlinked from reviews:

1. Click **Jobs (n)** (where **n** is the number of jobs that are already part of the review).
2. Click **X** next to the job you want to unlink.

The job is immediately unlinked from the review.

## Send All Notifications

Comment notifications are delayed by default, click the **Send All Notifications (n)** link to manually send all of your delayed notifications for the review.

Where **n** is the number of delayed notifications that will be sent for the review.


For more information about comment notification delay, see ["Comment notification delay" on page 336](#).

## Information panel

By default, when you open the Review page for the first time, the file list panel and the information panel are hidden. Use the **Show more context** buttons to view the file list panel and the information panel or use the keyboard shortcuts, **Shift + Alt + L** to view the file list and **Shift + Alt + S** to view the Information panel.

For more information about the show more context buttons, see ["File diff panel" on page 436](#).

## Blocking Approval or Blocking Commit

Shows the total number of items that are blocking approval or commit for a review. Click  to see details about the items that are blocking approval for a review. Once a review has been approved, the **Blocking Approval** section is not displayed.

For pre-commit reviews, if a user adds a new requirement to an approved review, then that requirement is displayed in the **Blocking Commit** section. For example, if a project initially has the **Minimum up votes** requirement set to one and this requirement is satisfied, and the review is in approved state. Now, if the **Minimum up votes** requirement changes to two then this requirement is displayed in the **Blocking Commit** section. Only when this requirement is satisfied the user can do a commit.

The **Blocking Approval or Blocking Commit** list consists of:

- **Pending Comment Tasks:** Provides a list of pending review comments that are flagged as *tasks*. The pending comment tasks are shown only when the `disable_approve_when_tasks_open` configurable is set to true in the `SWARM_ROOT/data/config.php` file.

### Note:

If P4 Code Review is configured to prevent approval of reviews with open tasks and a review has open tasks, the **Approve**, and **Approve and commit** options will not be available for the review. This option is configured by an administrator. See ["Disable approve for reviews with open tasks" on page 676](#).

To approve, or approve and commit a review with open tasks, you must address the tasks first and then set them to **Task Addressed**, or **Not a Task**. See ["Set a task to Task addressed or Not a task" on page 333](#) for details.

For more information about approving a review with pending tasks, see ["Approve a review with open tasks" on page 335](#).

- **Required Reviewers:** Provides a list of all the Individual and group reviewers that are required to vote on the review. A review can not be approved or committed until the required reviewers have voted up on the review.

A required ["Individual reviewer" on page 426](#) has a star badge over their avatar. A required ["Group reviewer" on page 425](#) either has a star badge or a star badge with a 1 over their avatar.

- **Blocking Tests:** Provides a list of workflow tests that have **Failed** or have an **In progress** status that is blocking approval for a review. A review can not be approved or committed until the review author fixes all the failed tests. For more information about a failed test, see ["Tests" on page 428](#).
- **Minimum Up votes:** Provides the number of **Minimum Up votes** required for each project that you have access to. This includes the maximum number of **Minimum Up votes** required for all branches in a project blocking approval or commit for a review. For more information about **Minimum up votes** for a project see, [General Settings tab](#). For more information about **Minimum up votes** for a project branch, see [Branches tab](#).

For [private project](#), the maximum number of **Minimum Up votes** for all private projects and their branches that is blocking approval for a review is displayed.

- **Moderator Votes:** Provides a list of branch moderators that are blocking approval or commit for a review. Depending on the configuration of `moderator_approval` configurable in the `SWARM_ROOT/data/config.php` file, the branch moderator approval is required. Ensure that the branch moderator approves the review and not just up votes it.


By default, when a review spans multiple branches that have different moderators, only one moderator from any one of the branches needs to approve the review.

P4 Code Review can be configured to require that one moderator from each branch must approve the review, this is a global setting. If a moderator belongs to more than one of the branches spanned by the review, their approval will count for each of the branches they belong to. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

## Reviewers

The list of reviewers for the review is displayed in the **Information panel** to the right of the review description.

The Reviewers list consists of:

- **Edit**  button (if enabled): click to edit the reviewers for the review. See ["Edit reviewers" on page 448](#).
- **Up** vote and **Down** vote count: indicates the number of up votes and down votes the review has.
- **Groups:** lists groups that are reviewers for the review. When at least one person in the group has voted, a vote icon is displayed to the right of the group indicating whether the group, as a whole, has voted up or down. Click on the group to see who has voted, and how they have voted. See ["Group reviewer" on the next page](#).
- **Individuals:** lists individuals that are reviewers for the review. When an individual has voted on the review, a vote icon is displayed to the right of the user indicating whether they voted up or down. For more information about individual reviewers, see ["Individual reviewer" on page 426](#).

**Note:**

By default, reviewer group members are not displayed in the *Individuals* area of the reviews page when they interact with a review (vote, comment, update, commit, archive, etc.). This avoids overloading the *Individuals* area with individual avatars if you have large reviewer groups.

An exception to this behavior is when a member of a reviewer group is also an individual [required reviewer](#), in this case their avatar will be displayed in the *Individuals* area.

See ["Expand group reviewers" on page 678](#) for details on displaying reviewer group members when they interact with a review.


**Group reviewer**

If you are a member of a group that is a reviewer on the review, click a vote button in the review description header to vote up or down. Your vote is registered for the group, and is also displayed to the right of your name in the **Reviewers** list. Click on the group in the **Information panel** to see how individual members have voted on the review.

**Changing group vote settings**

You must be the review author, a project member, a project moderator, or a user with super privileges to change the group settings.

**To change the group settings for the review:**

1. Click the **Reviewers** list **Edit**  button.
2. Click the **Groups** tab.
3. Click the dropdown arrow to the right of the group you want to change.
4. Select one of the following for the group:

**Optional Default:**

- If any group member votes down the review, an icon is displayed to the right of the group to indicate the group has voted the review down.
- If at least one group member votes up the review and **no** members of the group vote down the review, an icon is displayed to the right of the group to indicate the group has voted the review up.

**Require one:** Indicated by a star badge with a 1 on the group avatar.

- If any group member votes down the review, an icon is displayed to the right of the group to indicate the group has voted the review down.
- If at least one group member votes up the review and **no** members of the group vote down the review, an icon is displayed to the right of the group to indicate the group has voted the review up. See [Required reviewers](#) for details.

**Require all:** Indicated by a star badge on the group avatar.

- If any group member votes down the review, an icon is displayed to the right of the group to indicate the group has voted the review down.
- If all the group members vote up the review and **no** members of the group vote down the review, an icon is displayed to the right of the group to indicate the group has voted the review up. See [Required reviewers](#) for details.

**Remove from review:** Removes the group from the review.

#### Note:

If the review includes content that is part of a project or branch with [Retain default reviewers](#) enabled, the following restrictions apply to the review:

- The voting option for a retained default reviewer can only be changed to a stricter level, you cannot reduce the voting level.
- Retained default reviewers cannot be removed from the review.


### Individual reviewer

When an individual reviewer has voted on a review, a vote icon indicating whether they voted up or down is displayed to the right of their name in the **Reviewers** list. [Required reviewers](#) have a star badge over their avatar.

### Changing an individual's vote settings

You must be the review author, a project member, a project moderator, or a user with super privileges to change settings for an individual reviewer.

#### To change an individual's settings for the review:

1. Click the **Reviewers** list **Edit**  button.
2. Click the **Users** tab if it isn't already displayed.
3. Click the dropdown arrow to the right of the user you want to change.
4. Select one of the following for the user:
  - **Optional** Default
  - **Required:** Indicated by a star badge over the user's avatar.
  - **Remove from review:** Removes the user from the review.

#### Note:

If the review includes content that is part of a project or branch with [Retain default reviewers](#) enabled, the following restrictions apply to the review:

- The voting option for a retained default reviewer can only be changed to a stricter level, you cannot reduce the voting level.
- Retained default reviewers cannot be removed from the review.

## Review approval

A review can be approved when the following requirements are met:

- All of the [Required reviewers](#) on the review have voted up.
- The **Minimum up votes** on the review has been satisfied for **each** of the [projects](#) and [branches](#) the review spans.

Use the [Change state](#) button to approve the review.

### Tip:

- If the review is moderated, only the moderator can approve the review. For more information about review moderation, see ["Moderators" on page 455](#).
  - **Automatic approval of reviews:**
    - If the **Automatically approve reviews** workflow rule is enabled for the review's project/branch, the review is automatically approved as soon as the review requirements are satisfied.
    - If the review has a moderator, the review will not be automatically approved. The moderator must approve the review manually.
- For more information about the **Automatically approve reviews** rule, see ["Workflow rules" on page 510](#).

## Tasks

Comments can be flagged as tasks. For more details on working with tasks, see [Tasks](#).

### Note:

If P4 Code Review is configured to prevent approval of reviews with open tasks and a review has open tasks, the **Approve**, and **Approve and commit** options will not be available for the review. This option is configured by an administrator. See ["Disable approve for reviews with open tasks" on page 676](#).

To approve, or approve and commit a review with open tasks, you must address the tasks first and then set them to **Task Addressed**, or **Not a Task**. See ["Set a task to Task addressed or Not a task" on page 333](#) for details.

A summary of the number and status of comments flagged as tasks is displayed in the **Information panel** to the right of the review.

### Note:

Archived comments that are flagged as tasks are not included in the summary or the **Tasks** dialog.







- **Red Flag:** Displays the number of open tasks on the review.
- **Green check mark:** Displays the number of addressed tasks on the review.
- **Blue double-check mark:** Displays the number of addressed and verified tasks on the review.






**Show Task details**  : Click to display a dialog listing all of the tasks associated with the review:

✕

Tasks

Reporter 

| Status    |                                                                                    | Reporter          | Description                                                                                                                                                                                                                                           |
|-----------|------------------------------------------------------------------------------------|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Addressed |   | Allison Clayborne | <a href="#">I have passed this to QA.</a>                                                                                                                                                                                                             |
| Addressed |   | Allison Clayborne | <a href="#">We will need to update this for the next revision.</a>                                                                                                                                                                                    |
| Open      |   | Claire Brevia     | <a href="#">This description needs more detail. I feel the lack of detail makes it very hard to review this work. When will you get some time to change this?</a>  |
| Verified  |  | Claire Brevia     | <a href="#">This makes no sense, please update</a>                                                                                                                                                                                                    |

Within the **Tasks** dialog, you can filter the tasks by the **Reporter** (the userid of the user who created the task), and/or by task state using the buttons at the top of the dialog:

- Click the **Red flag** button to display only open tasks (comments that need to be addressed).
- Click the **Green check mark** button to display only addressed tasks (comments that have been addressed).
- Click the **Blue double-check mark** button to display only verified tasks (comments that have been addressed and verified).





To change the state of a task from the **Tasks** dialog, click the task state dropdown for the task and select the new task state.


To view the full comment text for a task, click the ellipses to the right of the task.

## Tests

When the review has an associated [workflow](#) with [tests](#) configured on it or when [continuous integration](#) is configured for the project, P4 Code Review displays the test results for the current review version in the information panel:












-  **Passed:** Displays the number of the tests that have passed for the review version.
-  **Failed:** Displays the number of the tests that have failed for the review version.
-  **Manual:** Displays the number of **On Demand** tests that can be run manually for the review version.
-  **In Progress:** Displays the number of tests that are running for the review version.

**Show test details**  button: Click to view test run information for tests associated with the review version:

✕









### Tests


| Status                                                                                     | Title                          |                                                                                     |                                                                                                |
|--------------------------------------------------------------------------------------------|--------------------------------|-------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|
|  Passed   | <a href="#">main-codeSniff</a> |  | Rerun test  |
|  Passed   | <a href="#">main-EsLint</a>    |  | Rerun test  |
| ^  Failed | <a href="#">main-Jest</a>      |  | Rerun test  |

---

### Messages

Error contacting server

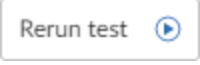
|                                                                                                 |                              |                                                                                       |                                                                                                  |
|-------------------------------------------------------------------------------------------------|------------------------------|---------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
|  Passed      | <a href="#">main-jsLint</a>  |  | Rerun test  |
|  Manual      | <a href="#">main-library</a> |  | Run test    |
|  In progress | <a href="#">main-modules</a> |  |                                                                                                  |


If your continuous integration system calls back to P4 Code Review with a URL, the test run is linked to the URL provided. If the test result contains any messages, click the **Show more**  button to the left of the test to view them

When the content of a review is unchanged between review versions, the tests are not rerun because the files have not changed. In this case, the earlier test results are displayed and marked with the word

**(Copy)**. You can manually run these tests if required by using the  **Run test** button.

The following actions are available from the **Tests** dialog for tests called by a workflow:

- **Rerun test:** If a test has completed you can rerun the test, click  to rerun the test for the most recent version of the review.

- **Run test:** If the review workflow has a test set to run **On Demand**, click  to manually run the test for the most recent version of the review.

**Tip:**

- The **Run test** and **Rerun test** buttons are only available if you are logged in to P4 Code Review, viewing the latest version of the review, and the test was called by a workflow.
- The **Rerun test** button is not available for tests that are in progress.
- When you click **Run test** or **Rerun test**, all of the buttons in the list are temporarily disabled while P4 Code Review starts the test and checks if any other tests are in progress.
- The **Rerun test** and **Run test** buttons are only available for tests that are run by a workflow. They are not available for tests that are run because they are configured on a project.

**Note:**

**Private projects:** if a test for a private project is added to your review because **Iterate tests for affected projects and branches** is selected for the test, P4 Code Review honors the private project's permissions and displays it as **Private project** in the test list to users that do not have permission to view it.

- For information about iterating tests, see ["Iterate tests for affected projects and branches checkbox" on page 530](#).
- For information about private projects, see ["Private projects" on page 366](#).

## Files tab

Use the **Files** tab to view the files in the review and to see how they have changed using the P4 Code Review diff view.

Base version for this review → #2 by Allison Clayborne  
Shelved in 644 7 months ago

#2: Change 644 shelved into // projects/ mercury/ dev/ src/ api 0 out of 6 read +4 1 -1

Filter

./db.c#1

42 lines hidden above

```

43 * perferendis consequuntur qui quidem. Et et aut es
44 t ex.
45 */
46 void * initVoluptatem() {
47 int loop = 1;
48
49 // Harum quaerat in distinctio vitae adipisci impe
50 dit distinctio.
51 printf("Dicta voluptas mollitia est placeat au
52 t.\n");
53 et = tempora + 1;
54 voluptas = atoi(qui)?quas:"eos";
55 runOccaecatI(ut, sed);
56 }
57
58 // Abbazia hodie conubia litora nunc.
59 dolor = flag / 1;
60 printf("Femina sem bis non dolor.");
61 printf("Wisi, mando gladius.");
62
63 // Harum quaerat in distinctio vitae adipisci impe
64 dit distinctio.
65 printf("Dicta voluptas mollitia est placeat au
66 t.\n");
67 et = tempora + 1;
68 voluptas = atoi(qui)?quas:"eos";
69 runOccaecatI(ut, sed);
70 }

```

6 lines hidden

P4 Code Review supports stream specs in your workspace using the [Private editing of streams](#) feature. If a changelist or review contains a stream spec, it will be displayed first in **Files** with the prefix **stream: //**, for example: `stream://MyStreamDepotName/MyStreamSpecLocationName`. A changelist/review can only contain one stream spec.

## Select review versions to view

Base version for this review → #2 by Allison Clayborne  
Shelved in 644 3 days ago

The review version selectors are used to specify which versions of a review you want to *diff*, they are located in the **Files** tab above the list of files.

- **Left dropdown selector:** the base version for the review is selected by default, base is the revision of the file that was checked out of the depot before it was changed for this review. The current review version in the depot, sometimes called Head can be selected, if there are multiple versions of the review a specific version can be selected.
- **Right dropdown selector:** the latest version of the review is selected by default. If there are multiple versions of the review, a specific version of the review can be selected.

### Tip:

If a review consists of one or more P4 Code Review-managed changelists. When comparing versions of a review, P4 Code Review is showing any differences between the selected versions, not the review author's personal changelist. See ["Internal representation" on page 402](#) for details.

## File list header

The file listing header contains the following elements:

### Review version and common file path summary

- Comparing the files in the latest version of the review to Base or Head:

**#3: Change 753398 shelved into //depot/main/swarm/collateral/sphinx/administration**

- The current version of the review.
- Which changelist contains a shelved copy of the review's files.
- The common path for all of the files in the review.
- Comparing two versions of the review:

**#2-3: Changes between shelf 753759 and shelf 753398 into //depot/main/swarm/collateral/sphinx/administration**

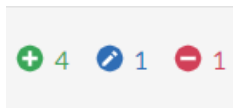
- Which two versions of the review are being compared.
- Which changelists contain the files being compared.
- The common path for all of the files in both versions of the review.

### Total number of read files

Shows the number of files that are marked as read out of the total number of files in a review.

### File change summary


The file change icons in the file listing header indicate the number and type of change for the files in the review. For example, the icons below indicate that:



- One file has been added, branched, or imported
- Six files have been edited or integrated
- Zero files have been deleted


### Expand the Diff view to full screen



Select the  button to expand the diffs view to full screen. Once the diffs view is expanded to full screen, select the button a second time to exit the full screen mode or use the keyboard shortcut **Shift + Alt + F**.

### Diff actions button



The **Diff actions**  button gives you access to more actions you can perform when viewing a file diff.

- **Expand or collapse all files in a review**

Expands or collapses all the files within a review if one or more files are in the collapsed or expanded state. Depending on whether the files are expanded or collapsed, use either of the following options:

- **Expand all files** (only displayed if all or at least one file is collapsed) Select to expand all the files within a review.
- **Collapse all files** (only displayed if all files are expanded) Select to collapse all the files within a review.

Use keyboard shortcut **Shift + Alt + E** to expand or collapse all files in a review.

- **Show or hide all in-line comments for all files in a review**

Shows the total number of comments across all files in a review. Depending on whether the inline comments are expanded or collapsed, use either of the following options:

- **Hide in-line comments**: select to collapse all of the inline file comments.
- **Show in-line comments**: select to expand all of the inline file comments.

Use keyboard shortcut **Alt + C** to hide or show in-line file comments.

- **Show diffs in-line or side-by-side for a file**

Depending on whether the diffs are shown in-line or side-by-side, use either of the following options:

- **Show diffs in-line**: this option only works for text files. Select to display the diffs in inline format.  
  
When a comment is added to a diff line or an unchanged line in the inline diff view, the in-line comment will span the entire file diff panel.
- **Show diffs side-by-side**: this option only works for text files. Select to display the diffs in side-by-side format. The oldest file revision is shown in the left pane, and the newer one is shown in the right pane.  
  
When a comment is added to a diff line in the side-by-side view, the comment will only appear on the side where it was made. If a comment is added to an unchanged line, it will span the entire file diff panel.

Use keyboard shortcut **Alt + L** to toggle viewing the diffs in-line or side-by-side for a file.

- **Show or hide whitespace and tab characters for all files in a review**

Depending on whether the diffs are shown with whitespace and tab characters or not, use either of the following options:

- **Show whitespace and tab characters** (display of CRLF line endings is only supported in the classic review page): shows whitespace and tab characters for all text files. The tab characters are padded to show the correct alignment of tabs.
- **Hide whitespace and tab characters**: hides whitespace and tab characters for all text files.

Use keyboard shortcut **Alt + W** to show or hide whitespace and tab characters for all files in a review.

- **Show or hide whitespace diffs for all files in a review**

Depending on whether the diffs are shown with whitespace or not, use either of the following options:

- **Show whitespace diffs:** whitespace changes are highlighted, makes it easier to identify changes in file types where whitespace is important.
- **Hide whitespace diffs:** whitespace changes are not highlighted, this makes it easier to see the important changes in file types where whitespace changes are not important.

Use keyboard shortcut **Alt + D** to show or hide whitespace diffs for all files in a review.

- **Disable syntax highlighter**

P4 Code Review displays the code in the default syntax colors for that coding language. However, this feature can be disabled to display the code without color. For more information on the supported languages, see ["Supported syntax highlighting in the Review page" on page 311](#).

- **Navigate to the next file**

Select this option to open the next file in the Diff view or use keyboard shortcut **Alt+N**.

- **Navigate to the previous file**

Select this option to open the previous file in the Diff view or use keyboard shortcut **Alt+P**.

- **Show the shortcut help dialog**

Select this option to open the shortcut help dialog to view all the file control keyboard shortcuts or use keyboard shortcut **Alt+H**.

## File list

### Important:

- For a review containing large number of files, using **Ctrl+F** to search for a file does not work. You can use the file filter to search for a specific file in the review. To manually select a file from the file stack, scroll through the Review page.
- By default, when you open the Review page for the first time, the file list panel and the information panel are hidden. Use the **Show more context** buttons to view the file list panel and the information panel or use the keyboard shortcuts, **Shift + Alt + L** to view the file list and **Shift + Alt + S** to view the Information panel.  
For more information about the show more context buttons, see ["File diff panel" on page 436](#).

The file list panel contains the following elements:

### File filter

Filter files by file name, file extension, or folder name to find the files you are interested in. You can see the diff view for the matched search result in the File stack section.

### File list resize

Change the file list width by dragging the split bar left or right. Alternatively, click the split bar to collapse or expand the file list. If you change the file list width for a review, Swarm saves the width locally for that review. The saved width is used next time you view the review.

### View complete file list

Use the scroll bar to view the complete list of files.

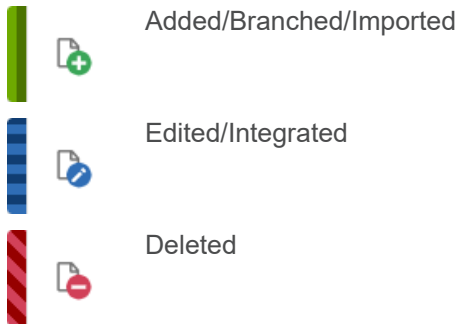
## File stack

All the files in a review are stacked together to enable you to scroll through each of the files and view their diffs. Select a file to expand the diffs view and select the file a second time to collapse the diff view.

The file stack contains the following elements for each file:

### File change type

Each file is marked with a ribbon and an icon indicating whether the file was:



## File content panel header



The file content panel header contains the following elements:

### File name

The name and revision of the file you are currently viewing.

### Total number of file comments

The total number of comments for a file. The comments count includes all inline comments and file-level comments. It does not include the archived comments.


### View full context of a file

This button is only displayed for text file diffs. Toggles between displaying only the portions of the file that have changed and the full file.

### Open content in a new tab

Click the button to open the selected file in a new browser tab or use keyboard shortcut **Alt + T**.

### Mark file as read

Beside each file in a review is a **Mark file as read** button , which help you keep track of which files you have reviewed. The read flag is remembered independently for each user. If the content of a file is changed in an update to the review, the read flag automatically clears. This is particularly useful when a code review consists of many files.

When clicked, color is added to the button background and the read by badge is added to the file in the **File list**.


If a file has been marked as read, click the button a second time to reset the status to unread or use keyboard shortcut **Alt + R**.

## File diff panel

By default, all files larger than 1 MB in size are truncated. When a file is truncated a message is displayed. Use the `"max_size"` on page 613 configurable in the `SWARM_ROOT/data/config.php` file to increase or decrease this limit.

If you rename a file, edit it, and request a review, only the edited content is highlighted in the diff panel.

### Use AI for code analysis

Click the AI button  to get a summary explaining the diff. You can send the whole diff or just a selection for code analysis.



For more information, see ["AI for code analysis" on page 547](#).



**Important:** The AI code analysis feature is disabled by default. Use the `ai_review.enabled` configurable in the `SWARM_ROOT/data/config.php` file to enable AI code analysis.

For more information on how to enable AI code analysis, see ["Configuration overview" on page 585](#).

To discard an AI summary, select the **Discard summary** button.

■ **Text file diff:**

When you view a diff, the changes are highlighted:

- Red indicates lines that have been removed.
- Blue indicates lines that have been modified.
- Green indicates lines that have been added.

For more information on the supported extensions for syntax highlighting in the Review page, see ["Supported syntax highlighting in the Review page" on page 311](#).

When a comment is added to a diff line or an unchanged line in the inline diff view, the in-line comment will span the entire file diff panel.

When a comment is added to a diff line in the side-by-side view, the comment will only appear on the side where it was made. If a comment is added to an unchanged line, it will span the entire file diff panel.

**Example inline diff view:**

db.c#1

4

42 lines hidden above

43

43

\* perferendis consequuntur qui quidem. Et et aut est ex.

44

44

\*/

45

45

void \* initVoluptatem() {

46

46

int loop = 1;

47

47

48

48

// Abbatia hodie conubia litora nunc.

49

49

dolor = flag / 1;

50

50

printf("Femina sem bis non dolor.");

51

51

printf("Wisi, mando gladius.");

Steve Russell

commented on review 644 (version 1) - about a year ago

✓

...

@allison.clayborne

Is this really necessary to spam the console with this?

Add a comment

34 lines hidden

87

92

popFugiat(aperiam, quo);

88

93

addBeataeDevice(iste, ut);

89

94

}

90

95

91

96

97

97

/\*\*

98

98

\* Herbam class tortor justo, augue justo congue. Volutpat a magnis

99

99

\* cursus pupula. Aliquet diam peristo ullamcorper cultura. Suspendisse

100

100

\* nostra.

101

101

\*/

### Example side-by-side diff view:

./db.c#1

4

42 lines hidden above

43

\* perferendis consequuntur qui quidem. Et et aut es

t ex.

44

\*/

45

void \* initVoluptatem() {

46

int loop = 1;

47

48

// Harum quaerat in distinctio vitae adipisci impe

dit distinctio.

49

printf("Dicta voluptas mollitia est placeat au

t.\n");

50

et = tempora + 1;

51

voluptas = atoi(qui)?quas:"eos";

52

runOccaecati(ut, sed);

34 lines hidden

87

popFugiat(aperiam, quo);

88

addBeataeDevice(iste, ut);

89

}

90

91

42 lines hidden above

43

\* perferendis consequuntur qui quidem. Et et aut es

t ex.

44

\*/

45

void \* initVoluptatem() {

46

int loop = 1;

47

48

// Abbatia hodie conubia litora nunc.

49

dolor = flag / 1;

50

printf("Femina sem bis non dolor.");

51

printf("Wisi, mando gladius.");

Steve Russell commented on review 644

(version 1) - about a year ago

@allison.clayborne Is this really necessary to spam the console with this?

Add a comment

52

53

// Harum quaerat in distinctio vitae adipisci impe

dit distinctio.

54

printf("Dicta voluptas mollitia est placeat au

t.\n");

55

et = tempora + 1;

56

voluptas = atoi(qui)?quas:"eos";

57

runOccaecati(ut, sed);

34 lines hidden

92

popFugiat(aperiam, quo);

93

addBeataeDevice(iste, ut);

94

}

95

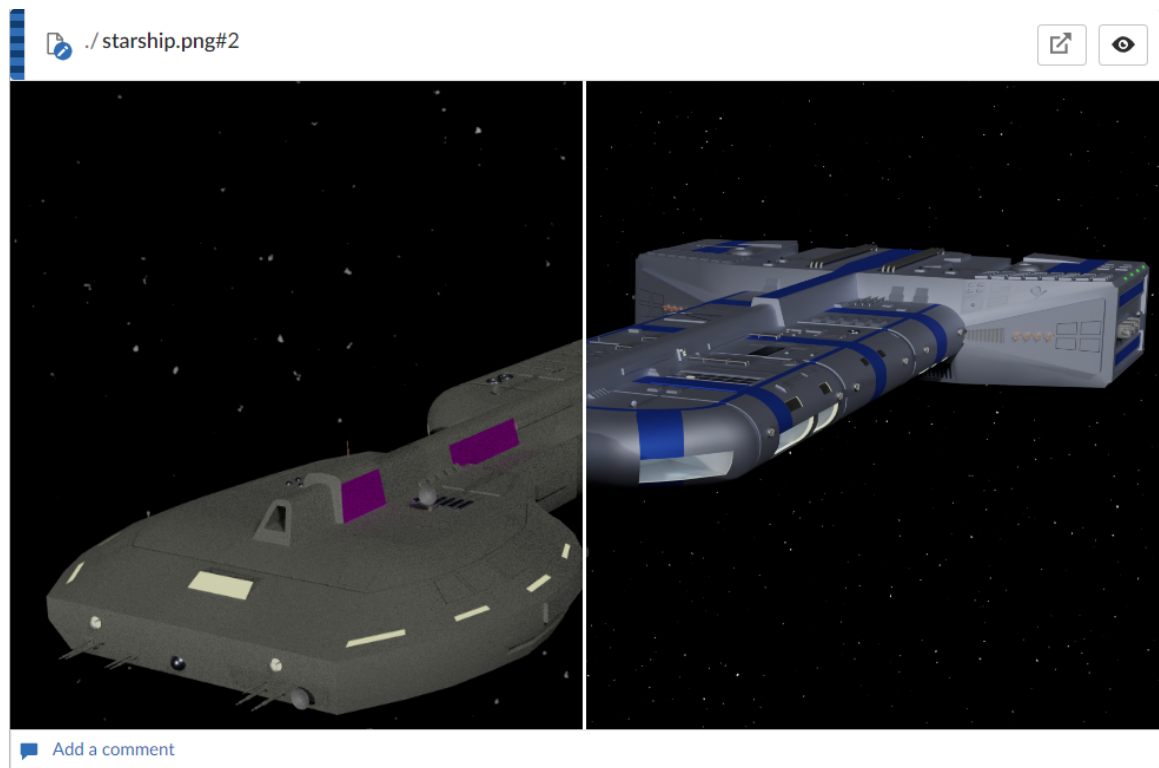
96

#### Image file diff:

Image types that are natively supported by your browser are displayed with a vertical slider on the image.

Drag the slider to the left to reveal more of the newer image and to the right to reveal more of the earlier one. Alternatively, click on the image to move the slider.

**Example image diff:**










- **Show more context buttons:**

Sometimes, the concise diff view needs to be expanded to fully understand the context of the change, use the **Show More** and **Show All** buttons to display extra lines around the change:

**Tip:**

The following buttons are not available for stream specs.


- **Show the file list panel**  button (only displayed on the vertical bar next to the file list panel): click to view the file list panel.
- **Show the information panel**  button (only displayed at the bottom of the information panel): click to view the detailed information panel.
- **Show All Lines to Start of File**  button (only displayed for the first change in the file): click to show all of the lines up to the start of the file.
- **Show More Lines for the Code Below**  button: click to show 10 more lines above the change, the extra lines are displayed in the pane below the button.
- **Show Entire Section**  button: click to show all of the lines between the changes that are above and below the button, the two changes and the lines between them are displayed in a single pane.
- **Show More Lines for the Code Above**  button: click to show 10 more lines below the change, the extra lines are displayed in the pane above the button.
- **Show All Lines to End of File**  button (only displayed for the last change in the file): click to show all of the lines down to the end of the file.


## Comments tab

The **Comments** tab is used to view all of the comments in the review.

Files **6** Comments **4** Activity Send All Notifications (0)

5 archived comments


 **Steve Russell** commented on review 644 (version 1) [//projects/mercury/dev/src/api/db.c, line 51](#) - 11 months ago ✓ read ✓

 **Allison Clayborne** commented on review 644 (version 1) [//projects/mercury/dev/src/api/db.c, line 105](#) - 10 months ago 👁️ ⋮

```
+ */
+int TalisAliquam(long quandoScelerisqueSem, float count, double litora) {
+ // Peristo velit parco.
+ parco = volutpat / 1;
+ printf("Aquam arcu vehicula faucibus umbra magna.");
```


@steve.russell what do you think of this?

👍

 **Steve Russell** commented 4 months ago 👁️ ⋮

This looks good to me. I am happy for it to be checked in.

👍

 **Allison Clayborne** commented 4 months ago 👁️ ⋮

Thanks for the confirmation.

👍

Add a comment

Drag and drop files to attach them or [Choose your files](#)



Post ☐ Flag as Task ☒ Post and notify (0)

For information about working with comments, see "Comments" on page 328.



## Activity tab

The **Activity** tab presents a list of the events on this review.



Files **6** Comments **4** Activity Send All Notifications (1)

  **Allison Clayborne** cleared an issue on review 644 for [mercury:dev](#) 6 minutes ago



This needs another review please ⋮

  **Allison Clayborne** archived comment on review 644 for [mercury:dev](#) 8 minutes ago

This description needs more detail. ⋮

  **Allison Clayborne** replied to a comment on review 644 for [mercury:dev](#) 8 minutes ago

Thanks for the confirmation. ⋮

  **Steve Russell** replied to a comment on review 644 for [mercury:dev](#) 9 minutes ago

This looks good to me. I am happy for it to be checked in. ⋮

Events in the activity tab include:

- When the review was started
- When a new reviewer joins the review
- When the review's state changes
- When the review's files are updated

- When a reviewer votes on the review
- When someone comments on the review, or one of its files
- When tests pass or fail, provided continuous integration is configured

## Known limitations with pagination

P4 Code Review limits the size of files displayed in both reviews and standard file views. This pagination has several known limitations.

### Truncated diffs may hide unchanged lines and additional diffs

When viewing the changes of a large file that has been truncated, if a diff with more than 500 lines is followed by fewer than 10 unchanged lines along with another diff, only the first (truncated) diff is shown initially. The 10 unchanged lines and the second diff are hidden until the first diff is fully expanded using the **Show next** button.

**Workaround:** Reviewers should fully expand any truncated diffs to ensure all subsequent changes and unchanged lines are visible.

### Comment links may fail to navigate to truncated diff sections

Clicking a comment from the **Comments** tab does not navigate you to its location in the **Files** tab if the comment was made in a section of the diff that has been truncated. The comment remains hidden until the diff is manually expanded.

**Workaround:** If you do not see the in-line comment, click **Show next** to expand the truncated diff manually in the **Files** tab to view the comment in context.

### Complexity column displays incorrect line count for large added or deleted files

When a review includes added or deleted files with a large number of lines, the Complexity column on the Review page may display an incorrect line count. Instead of showing the full number of lines in the file (for example, 10,000), it reflects only the number of lines in the first truncated diff chunk (this is usually 3,800). This issue does not affect edited files.

## Activities

This section describes the major activities that affect code reviews, including starting a review, updating a review, and fetching a review's files.

## Start a review

### Important:

If your P4 Server is configured as a commit-edge deployment, and your normal connection is to an edge server, P4 Code Review refuses to start reviews for shelved changes that have not been promoted to the commit server.

Within P4 Code Review, this means that the **Request Review** button does not appear for unpromoted shelved changes. Outside of P4 Code Review, attempts to start reviews for unpromoted shelved changelists appear to do nothing. Ask your P4 Server administrator for assistance if you cannot start a review.

An administrator of the P4 Server can automatically promote shelved changes to the commit server by setting the configurable `dm.shelve.promote` to 1.

**Tip:**

If your changelist only contains a stream spec and its location in the P4 Server is not associated with a P4 Code Review project, the review that you create will not have any default reviewers or workflow rules. To associate the review with a project so that it has default reviewers and obeys the project workflow rules, include a file change from the project path in the changelist when you create the review.

1. Use the P4 command-line (P4) or a client to create the shelved or committed changelist.
2. Start a code review by using one of the following approaches:

## Use P4 Code Review:

1. Use P4 Code Review to view a shelved or submitted changelist.

**Tip:**

To view a shelved or submitted changelist, use a [Quick URL](#). For example, if your change is **54321**, visit the URL: <https://myswarm.url/54321>.

2. Click the **Request Review** button to request a review of that changelist.

Requesting a review on a shelved changelist uses the [pre-commit](#) model and requesting a review on a submitted changelist uses the [post-commit](#) model.

**Important:**

The **Request Review** button is disabled for changelists with a file count greater than the limit set for `max_changelist_files` in the [SWARM\\_ROOT/data/config.php](#) file. This is to prevent out of memory errors within P4 Code Review. For more information on changing this file count limit, see ["Changelist files limit" on page 573](#)



## Use P4 command-line (P4):

When you are about to shelve or submit files:

1. Include `#review` within your changelist (separated from other text with whitespace, or on a separate line).

Once the review begins, P4 Code Review replaces `#review` with `#review-12345`, where 12345 is the review's identifier.

### Note:

The `#review` keyword is customizable. For details, see ["Review keyword" on page 668](#).

2. At this time, you can add reviewers to the code review by using `@mention` for users, and `@@mention` for groups in the changelist description for each desired reviewer.

If your `@mention` or `@@mention` includes an asterisk (\*) before the *userid* or *groupid*, for example `@*userid`, that user or all of the group members become required reviewers. If your `@@mention` includes an exclamation mark (!) before the *groupid*, for example `@@!groupid`, the members of that group become required reviewers but only one member of the group is required to vote. See ["Required reviewers" on page 457](#) for details.

3. Complete your shelve or submit operation.

### Warning:

If you shelve a changelist and subsequently edit the description to include `#review`, a review is **not started**. You must re-shelve the files after adding `#review`.

### Tip:

You can also start a review with P4V, P4 for Visual Studio, and P4 for Eclipse. See below for details:

- **P4V:** see the [P4 Code Review integration features](#) section of the [P4 Visual Client \(P4V\) Documentation](#).
- **P4VS:** see the [Managing files](#) chapter of the [P4VS User Guide](#).
- **P4Eclipse:** see the [Reviewing changes](#) chapter of the [P4Eclipse User Guide](#).

### Note:

If you are using P4V and its P4 Code Review integration, and you encounter the error `Host requires authentication`, ask your P4 Server administrator for assistance. See ["P4V Authentication" on page 584](#) for details.

## Update a review

To update a code review, use one of the following approaches:

- For a [pre-commit](#) review that you authored:

1. Edit the files
2. Shelf the files

You can repeat these steps as many times as necessary.

#### Tip:

If you want to stop P4 Code Review from updating the review until you have completed your changes, add the work-in-progress tag (`#wip` by default) to the changelist description. For information on using the work-in-progress tag, see ["Work-in-progress tag" on page 371](#).

- For a [post-commit](#) review, or a review where you are not the author:

1. [Fetch the review's files](#) into a new changelist
2. Edit the files
3. Update the changelist's description to include `#review-12345` (separated from other text with whitespace, or on a separate line)
4. Shelf the changelist's files

Once these steps are complete, further updates involve editing the files, and then shelving the changelist's files.

#### Warning:

If you use an invalid review identifier, it will appear that nothing happens. P4 Code Review is currently unable to notify you of this situation. If the review has not been correctly updated, use the **Add Change** button in the review heading to add the changelist to the review, see ["Add a changelist to a review" on page 450](#).

## Fetch a review's files

First, determine the changelist containing the review's files:

1. Visit the review's page.
2. The current review version's changelist appears in the file list heading:

**#3: Change 697707 shelved into `//depot/main/swarm`**

In this example, the changelist is 697707. You use the identified changelist in place of shelved changelist below.

3. Decide whether you will use [P4 command-line \(P4\)](#) or [P4V](#) to fetch the files, and follow the instructions in the appropriate section below.

## Using P4 command-line (P4)

1. For a **shelved changelist**, use a command-line shell and type:

```
$ p4 unshelve -s shelved changelist
```

2. For a committed changelist, use a command-line shell and type:

```
$ p4 sync @committed changelist
```

### Note:

Your client's view mappings need to include the changelist's path.

## Using P4V

### For a shelved changelist:

1. Select **Search > Go To**.
2. Change the select box to **Pending Changelist**.
3. Type in the *shelved changelist* number and click **OK**.
4. Select the files in the **Shelved Files** area.
5. Right-click and select **Unshelve**.
6. Click **Unshelve**.

### For a committed changelist:

1. Select **Search > Go To**.
2. Change the select box to **Submitted Changelist**.
3. Type in the *submitted changelist* number and click **OK**.
4. Select the files in the **Files** area.
5. Right-click and select **Get this Revision**.
6. Click **Close**.

## Deleting shelves

If you have a number of unused shelves, you can delete them if you want to tidy up your workspace.

### Important:

By default, when you delete files from a shelved changelist, the files are not removed from the associated review.

However, P4 Code Review can be configured to remove files from a review when they are deleted from an associated shelf, see ["Process shelf file delete when" on page 680](#).

Do not use the P4 Code Review user that is configured in the [P4 Code Review configuration file](#) when deleting shelves, or deleting files from shelves. The P4 Code Review logic processes the *shelve-delete* trigger event, if the event is invoked by the P4 Code Review user it is rejected. The delete operations will fail.

To delete a shelf from a changelist without removing the shelved files from the associated review, use [P4 command-line \(P4\)](#) or [P4V](#) to delete the entire shelf in one operation as described below. This enables you to delete a shelved changelist in your workspace without affecting the associated review even if P4 Code Review is configured to remove files from a review when they are deleted from a shelf.

## Using P4 command-line (P4)

To delete a shelved changelist without removing the files from the associated review:

1. Use a command-line shell and type:

```
$ p4 shelve -d -c changelist
```

### Important:

Do not delete the files from the shelf individually. Deleting the entire shelf ensures that files are not removed from an associated review even if P4 Code Review is configured to do so.

2. The shelved files are deleted from the pending changelist.

## Using P4V

To delete a shelved changelist without removing the files from the associated review:


1. In P4V, right-click on the pending changelist.
2. Select **Delete Shelved Files**.

### Important:

Do not delete the files from the shelf individually. Deleting the entire shelf ensures that files are not removed from an associated review even if P4 Code Review is configured to do so.

3. The shelved files are deleted from the pending changelist.

## Edit reviewers

A review author can edit the reviewers for a review by using the Reviewers **Edit**  button in the review page **Information panel**. Reviewers are able to join or leave reviews, or to change whether their vote is required or optional.

### Note:


If the review includes content that is part of a project or branch with [Retain default reviewers](#) enabled, the following restrictions apply to the review:

- The voting option for a retained default reviewer can only be changed to a stricter level, you cannot reduce the voting level.
- Retained default reviewers cannot be removed from the review.

Additionally, the following individuals can edit reviewers:

- Users with *admin* or *super* privileges.
- If the review is *moderated*, the moderators.
- If the review is part of a project, but not moderated, all project members.
- If the review is not part of a project, any authenticated user.


#### To edit reviewers on a review:

1. Go to the review.
2. From the review **Information panel**, click the **Reviewers** list **Edit**  button.  
The **Manage reviewers** dialog is displayed.
3. Click the **Users** tab to edit a user or the **Groups** tab edit a group. This field auto-suggests users, and groups within P4 Server as you type (up to a combined limit of 25 entries).
4. **Change the voting requirements of a user or group:**
  - a. Click the dropdown arrow to the right of the user or group you want to change.
  - b. Select one of the following:
    - **Users:** select whether their vote is **Required** or **Optional**. A solid star means that their vote is required to approve a review, whereas the outlined star means that their vote is optional.
    - **Groups:** select whether the group vote is **Require all**, **Require one**, or **Optional**. A solid star means that all group member votes are required to approve a review, a solid star with a 1 inside means at least one group member must vote up and no group members vote down to approve a review, and the outlined star means that the group vote is optional.
5. **Add a user or group to a review:**
  - a. Click the **Add user** or **Add group** dropdown icon to display a list of users or groups:
  - b. Start typing the user or group name, the list is filtered as you type.
  - c. Click on the user or group to add to the review.
  - d. Click **+** to add them as a reviewer.
  - e. Set the reviewer type from the dropdown to the right of the user or group.
6. **Remove a user or group from a review:**
  - a. Click the dropdown icon to the right of the user or group to be removed.
  - b. Select **Remove from the review** from the dropdown menu.
7. To close the **Manage reviewers** dialog, click **X**.

## Add a changelist to a review

Once a review has been started you can add a changelist to the review. It can be useful to add changelists to an existing review. For example, if follow up changes are made to files in a review or if you need to group a number of changelists under a single review.

By default, for a pre-commit or a post-commit review, when a changelist is updated, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a pending changelist" on page 452](#).

The **Add change** option in the **Review actions**  button is used to add a changelist to an existing review. The option is not available if the review is in a state that is protected from change by the review workflow rules. For details of the **On update of a review in an end state** rule, see ["Workflow rules" on page 510](#).

### Tip:

You can also add a changelist to a review directly from the changelist description. See ["Review creation and modification outside of P4 Code Review" on page 462](#).

The options available for the **Add change** option depend on whether the review is pre-commit or post-commit:

#### ■ Pre-commit reviews:

- **Append pending changelist:** when you add a pending changelist to a review, the files in the changelist are appended to the existing files in the review. See ["Append a pending changelist to a review" on the next page](#).

### Note:

A review can only contain 1 stream spec. If you append a changelist with a stream spec to a review that already contains a stream spec, the spec in the changelist replaces the original one in the review. If it is a different spec from the original spec in the review, P4 Code Review cannot display the diff between them and displays **File content unchanged**.

- **Replace with pending changelist:** when you add a pending changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a pending changelist" on page 452](#).
- **Replace with committed changelist:** when you add a committed changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a committed changelist" on page 453](#).

### Note:

If you replace a pre-commit review with a committed changelist, the new version of the review will be a post-commit review.

#### ■ Post-commit reviews:

- **Replace with pending changelist:** when you add a pending changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the

review. See ["Replace review with a pending changelist" on the facing page](#).

**Note:**

If you replace a post-commit review with a pending changelist, the new version of the review will be a pre-commit review.

- **Replace with committed changelist:** when you add a committed changelist to a review, all of the files in the review are replaced with the files in the changelist you are adding to the review. See ["Replace review with a committed changelist" on page 453](#).

**Tip:**

When the content of a review is changed, P4 Code Review checks to see which branches are in the new version of the review:

- **If a new branch was added to the review:**
  - Default reviewers on the new branch are added to the review.
  - Moderators from the added branch become moderators for the review alongside the existing moderators.
  - **Only if workflow is enabled:** if the new branch is associated with a workflow, the [workflow is merged](#) with the existing workflow. The most restrictive workflow is used for the review.
- **If a branch is no longer part of the review:**
  - Reviewers for the review are not changed.
  - Moderators from the removed branch no longer moderate the review.
  - **Only if workflow is enabled:** if the branch was associated with a workflow, the branch workflow is removed from the review.

## Append a pending changelist to a review

The **Append pending changelist** option is used to add a pending changelist to a pre-commit review.

**Example:**

- Original review contains the following files: A#1, B#2, C#1, D#1, and E#4
- Changelist contains the following files: A#2, C#3, E#4 (marked for delete), and F#1
- Appending the changelist to the original review generates a new version of the review that contains the following files: A#2, B#2, C#3, D#1, E#4 (marked for delete), and F#1

**Important:**

**Committing a review with P4 Code Review (recommended):** P4 Code Review automatically commits the files in the approved version of the review.

**Committing a review outside of P4 Code Review:**

Before you commit the review:

1. Unshelve the review into the pending changelist associated with the review.
2. Reshelve the files in the pending changelist.
3. Commit the pending changelist.
4. This process ensures that all of the files in the approved version of the review are committed.


**Workflow feature [enabled \(default\):](#)**

P4 Code Review can be configured to automatically check that the files being committed match the files in the approved version of the review by using the **On commit with a review** workflow rule. For information about the **On commit with a review** rule, see "[Workflow rules](#)" on page 510.

**Workflow feature [disabled:](#)**

P4 Code Review can be configured to automatically check that the files being committed match the files in the approved version of the review by using the [strict](#) trigger option.

To append a changelist to a review:

1. Click the **Review actions**  button.
2. Select **Append pending changelist** from the dropdown menu and the append review dialog is displayed with a list of recent pending changelists.
3. Enter the pending changelist number directly in the **@ Change** box, filter by depot path and user, or select a changelist from the list of recent changelists.

**Note:**

The changelist must not be part of another review, if it is P4 Code Review will reject it.

4. Click **Select** to append the changelist to the review.
5. The changelist is appended to the review and the review version is updated.

## Replace review with a pending changelist

The **Replace with a pending changelist** option is used to replace the files in a review with the files in a pending changelist.

**Note:**

If you replace a post-commit review with a pending changelist, the new version of the review will be a pre-commit review.

**Example:**

- Original review contains the following files: A#1, B#2, C#1, D#1, and E#4
- Changelist contains the following files: A#2, C#3, E#4 (marked for delete), and F#1




- Replacing the original review with the changelist generates a new version of the review that contains the following files: A#2, C#3, E#4 (marked for delete), and F#1

**Tip:**

When you replace a review with a changelist, the base revisions of the files in the new version of the review are the base revisions of the files in the replacement changelist.

To replace the files in a review with the files in a changelist:

1. Click the **Review actions**  button.
2. Select **Replace with a pending changelist** from the dropdown menu and the replace review dialog is displayed with a list of recent pending changelists:
3. Enter the pending changelist number directly in the **@ Change** box, filter by depot path and user, or select a changelist from the list of recent changelists.

**Note:**

The changelist must not be part of another review, if it is P4 Code Review will reject it.

4. Click **Select** to replace the review with the changelist.
5. The changelist files replace the files in the review and the review version is updated.

## Replace review with a committed changelist

The **Replace with a committed changelist** option is used to replace the files in a review with the files in a committed changelist.

**Note:**

If you replace a pre-commit review with a committed changelist, the new version of the review will be a post-commit review.


**Example:**

- Original review contains the following files: A#1, B#2, C#1, D#1, and E#4
- Changelist contains the following files: A#2, C#3, E#4 (marked for delete), and F#1
- Replacing the original review with the changelist generates a new version of the review that contains the following files: A#2, C#3, E#4 (marked for delete), and F#1

**Tip:**

When you replace a review with a changelist, the base revisions of the files in the new version of the review are the base revisions of the files in the replacement changelist.

To replace the files in a review with the files in a changelist:

1. Click the **Review actions**  button.
2. Select **Replace with a committed changelist** from the dropdown menu and the replace review dialog is displayed with a list of recent committed changelists:
3. Enter the committed changelist number directly in the **@ Change** box, filter by depot path and user, or select a changelist from the list of recent changelists.

**Note:**


The changelist must not be part of another review, if it is P4 Code Review will reject it.

4. Click **Select** to replace the review with the changelist.
5. The changelist files replace the files in the review and the review version is updated.

## Responsibility

Initially, code reviews have no reviewers.

User and group default reviewers can be set for individual projects and project branches. Each time a new review is created in the project or project branch, the default reviewers will be added to the review. See [default reviewers](#) for details.

Users can show their interest in participating in a code review by clicking the **Review actions**  button and selecting **Join review**, by voting on a review, or by commenting on a review or one of its files. Once a user shows such interest, they are added to the review's list of reviewers and share in the responsibility of performing the code review. Later on, reviewers can change whether their vote is required or optional, or leave a review (perhaps to prevent further notifications).

Looking at a [review list](#) can help you determine which reviews have likely not been started, using the **No reviewers** filter. Once a review has reviewers, it is considered to be active and appears in the review list with the state **Has Reviewers**.


Review participation is *advisory* by default, and is used to inform your team that a code review is being conducted. The disposition of the review is reflected in the review's current state, the badges that may appear over each reviewer's avatar, and any comments reviewers might add.

## Authors

The code review author is the user that created the review from a changelist.

**As a code review author, you can:**

- Add users as reviewers by including an [@mention](#) for each optional reviewer and an [@\\*mention](#) for each required reviewer in the changelist description. See ["Links in descriptions and comments" on page 387](#).
- Add groups as reviewers by including [@@mention](#) for each optional group reviewer, [@@\\*mention](#) for each [required reviewer](#) group (all members required), and [@@!mention](#) for each [required reviewer](#) group (one vote required) in the changelist description. See ["Links in descriptions and comments" on page 387](#).

- Start a review by using P4V, P4 Code Review, or from the command line of P4D. See ["Start a code review" on page 61](#).
- Edit users and groups as reviewers by using the **Edit Reviewers**  button on the review page to add, remove, and edit existing reviewers. See ["Edit reviewers" on page 448](#).
- Add the work-in-progress tag (#wip by default) to a pending changelist so that you can update files in your changelist without updating the review or creating a new review until you are ready. See ["Work-in-progress tag" on page 371](#)
- Add a changelist to your review, click the **Add change** button on the review and select the changelist you want to add to your review. See ["Add a changelist to a review" on page 450](#).
- Commit your review when it has been approved, click the **Review state** button on the review page and select **Commit**. See ["Change review state" on page 468](#).
- Delete your shelf when the review has been committed. See ["Deleting shelves" on page 447](#).

## Moderators

A project moderator is a user assigned to moderate reviews for a specific branch associated with a project.

When you add a moderator to a project branch, only the moderators can approve or reject reviews restriction is enabled for a project branch. See ["Project settings" on page 478](#) for details on adding moderators to project branches.

Changing the state of any review associated with a moderated branch is restricted as follows:

Only moderators can approve or reject reviews on the branch.

To add moderators to a branch:

1. Click the **Manage moderators** button.
2. Choose the **Users** or **Groupstab**.
  - a. If you selected **Users**, start typing a the name of a user. Suggested users on the P4 Server will appear automatically.
  - b. If you selected **groups**, then all of the members of that group will have the same moderator privileges for that project branch. Use this option if you wanted to add several users at once.
3. Click Add.

After you finish setting up the branch and save the project, reviews on this branch can only be approved or rejected by moderators.

Moderators can also transition a review to any other state. Below is a summary of branch moderator roles and review state rules.

| Role                      | Allowed actions                                                                                                                                                                                                  | Restrictions                                                                                                                                             |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Moderators                | <p>Can move a review to any state.</p> <p>Can approve or reject reviews.</p> <p>Can prevent automatic approvals.</p>                                                                                             | <p>None (unless <code>disable_self_approve</code> is enabled for self-approval). For more information, see <a href="#">disable_self_approve</a>.</p>     |
| Authors (not moderators)  | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Needs review</li> <li>Needs revision</li> <li>Archived</li> </ul> <p>Can attach committed changelists.</p>                                    | <p>Cannot approve or reject reviews.</p>                                                                                                                 |
| Authors (also moderators) | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Approve</li> <li>Rejected</li> <li>Needs review</li> <li>Needs revision</li> <li>Archived</li> </ul> <p>Can attach committed changelists.</p> | <p>Cannot approve their own reviews if <code>disable_self_approve</code> is enabled. For more information, see <a href="#">disable_self_approve</a>.</p> |
| Project members           | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Needs review</li> <li>Needs revision</li> </ul> <p>Can attach committed changelists.</p>                                                      | <p>Cannot approve, reject, or archive reviews.</p>                                                                                                       |



| Role        | Allowed actions     | Restrictions                                                                                       |
|-------------|---------------------|----------------------------------------------------------------------------------------------------|
| Other users | None                | Cannot change review states.                                                                       |
| All roles   | Can start a review. | Cannot change state if it's not in their allowed states (for example, can't change from Rejected). |

#### Additional notes

- Moderators can prevent automatic approvals. For more information about automatically approving reviews using workflow rules see ["Workflow rules" on page 510](#).
- By default, if a review spans multiple branches with different moderators, only one moderator from any branch needs to approve it. You can change this (via a global setting) to require one moderator per branch. If a moderator belongs to multiple branches, one approval counts for each. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

To delete a moderator, click the delete icon against their name.

Branch moderators(1), only moderators can approve or reject reviews
Manage moderators

| Name                                                                                                  | Type |                                                                                       |
|-------------------------------------------------------------------------------------------------------|------|---------------------------------------------------------------------------------------|
|  Allison Clayborne | User |  |

Rows per page: 25
1-1 of 1

## Required reviewers

Reviews can optionally have required reviewers. When a review has required reviewers, the review cannot be approved until all required reviewers and required reviewer groups have up-voted the review. If the review is associated with a project that has assigned moderators, even the moderators cannot approve the review without up-votes from all required reviewers (but they can reject the review).

When a group is a required reviewer, it can be set to operate in one of two ways:

- **Require all:** all members of the group must up-vote the review to allow the review to be approved.

- **Require one:** at least one member of the group must up-vote the review to allow the review to be approved. If any member of the group down-votes the review, the review cannot be approved.

Required reviewers are expected to take greater care while performing a review than non-required reviewers, as their votes affect whether a review can be approved or not.


To edit the reviewers for a review and to change whether a reviewer is required or not, see ["Edit reviewers" on page 448](#).

#### Note:

If a review involves a branch with assigned moderators, only a moderator can approve the review, even if all required reviewers have up-voted the review.

See ["Project settings" on page 478](#) for details on adding moderators to project branches.

## Add yourself as a reviewer


1. Open the review you want to join.
2. Click the **Review actions**  button.
3. Select **Join review** from the dropdown menu. Alternatively, you can **Vote Up** or **Vote Down** the review.

Your name is added to the **Reviewers** list in the **Information panel**, and you are now a reviewer.

## Remove yourself as a reviewer

#### Tip:

You cannot leave the review if you are a retained default reviewer on the review or you are a member of a group that is a reviewer on the review

1. Log in, if you have not already done so.
2. Open the review you want to leave.
3. Click the **Review actions**  button.
4. Select **Leave review** from the dropdown menu.

Your name is removed from the **Reviewers** list in the **Information panel**, and you are no longer a reviewer.

## Review workflow

There are many possible code review workflows. The section describes typical scenarios for a code review that P4 Code Review can handle.

## Basic review workflow

This section describes a basic workflow when reviewing code with P4 Code Review.

### Another developer reviews your code

1. You request a code review with a shelved change.
2. Another developer, Bill, sees the email notification from P4 Code Review, clicks the review link in the email, and begins looking at the diffs in the files belonging to the review. Curious about an implementation detail, Bill clicks the line he's curious about and adds his query in a P4 Code Review comment.
3. You receive an email notification regarding Bill's query. His query prompts you to clarify the code, say by renaming some variables and adding some better descriptive text in the surrounding code comments. You then update the review with your changes.
4. Bill sees the email notification that you have updated the review. He checks out the change, likes what he sees, and marks the review **Approved**.
5. You see the email notification that Bill has approved your review, so you commit your code.

### You review the code from another developer

1. Another developer, Charlie, requests a code review with a shelved change.
2. You receive an email notification from P4 Code Review, click the review link in the email, and begin looking at the diffs in the files in the review. You don't like what you see, as Charlie has tried to fix a bug using a technique you have already tried previously and know to be incorrect. You add comments to the code that needs attention, flag your comments as **tasks**, and mark the review **Needs revision**.
3. Charlie receives an email notification regarding your review, but disagrees with you, and adds his own comments justifying his implementation.
4. You receive an email notification regarding Charlie's comment. The technique is somewhat complicated, so rather than attempt to describe how it is incorrect, you unshelve the review's code to your own workspace, change Charlie's code, and shelve your changes. P4 Code Review updates the review with your new code.
5. Charlie receives an email notification regarding your updates to the review. He's still unconvinced, but he unshelves your changes to try them in his local workspace. He finds that your implementation works better, but sees a couple of areas where there could be improvements. He reshelves his latest work to update the review.
6. You receive an email notification regarding Charlie's updates, check out his changes, and realize that Charlie's work is now moot because the customer has revised his plans. You add a comment to the review reporting that fact, and **Reject** the review.

### Additional workflow tasks

This section describes additional workflow tasks that can be used with the [basic workflow](#) described above.

## Remove a file from a review

### Important:

P4 Code Review must be configured to use this function, see ["Process shelf file delete when" on page 680](#).

When configured, P4 Code Review will automatically remove files from a review when they have been deleted from an associated shelved changelist and the review is in a specified state. Typically P4 Code Review is configured to remove files from a review when the review state is **Needs review** and **Needs revision** but not when it is any other state.

### To remove a file from a review using P4V:

1. Right-click the shelved file you want to remove from the review.
2. Select **Delete**, and click **Yes** when prompted to confirm the deletion. The file is removed from the associated review if the review is in one of the states specified in the P4 Code Review configuration.

### Tip:

If you want to tidy up your workspace by deleting an unused shelf without removing the files from an associated review, see ["Deleting shelves" on page 447](#).

## Revert the content of a review to match an earlier review version

If the content of the latest version of a review contains changes you want to rollback, or is created by mistake, you can revert the review content to match an earlier version of the review. This section describes the process for reverting the content of a review.

### Tip:

- Reverting the content of a review increments the review version, it does not change the review version to the version you are reverting to.
- It is good practice to update the review description, or add a comment, so that users know that you have reverted the review with files from an earlier version.

### To revert the content of a review:

1. Find the changelist number for the review version you want to revert to.

### Tip:

The changelist numbers in a review can be found using the review version selector on the review page, see ["Select review versions to view" on page 431](#).

2. Unshelve the changes from the changelist you found in the previous step.
3. Create a new changelist and shelve the changes into the new changelist.
4. Use P4 Code Review to navigate to the review you want to revert.



5. Select the **Add change** option, and select **Replace with pending changelist** from the dropdown list. The **Select a pending changelist to replace the review** dialog is displayed.
6. Select the new changelist you shelved the files in and click **Select**. The new changelist files replace the files in the review and the review version is updated.

## Fix a review if it has been replaced with the wrong changelist

If you or another user has replaced the files in your review with the wrong changelist, you can fix the mistake so that the review contains the correct files.

### For example:

1. Create a pending changelist with files A<sup>#1</sup>, and B<sup>#1</sup>, this is changelist 250.
2. Request a review of changelist 250 using P4 Code Review by clicking the **Request Review** button on the changelist page.
  - Version 1 of review 251 is created containing files A<sup>#1</sup>, and B<sup>#1</sup>.
3. Go to review 251 in P4 Code Review, select the **Add change** option, select **Replace with committed changelist**, and select changelist 241. Changelist 241 contains files X<sup>#1</sup>, and Y<sup>#1</sup>.
  - This replaces the files in the review with the files in changelist 241.
  - Version 2 of review 251 is created, containing only files X<sup>#1</sup>, and Y<sup>#1</sup>.
4. You realize that you have made a mistake, you meant to append changelist 242 to review 251.

### Fix the mistake:

Review-251 should contain files A<sup>#1</sup>, and B<sup>#1</sup> from changelist 250 and files C<sup>#1</sup>, and D<sup>#1</sup> from changelist 242.

#### Tip:

It is good practice to update the review description, or add a comment, so that users know why you have changed the files in the review.

1. Go to review 251 in P4 Code Review, select the **Add change** option, select **Replace with pending changelist**, and select changelist 242. Changelist 242 contains files C<sup>#1</sup>, and D<sup>#1</sup>.
  - This replaces the files in the review with the files in changelist 242.
  - Version 3 of review 251 is created, containing files C<sup>#1</sup>, and D<sup>#1</sup>.
2. Edit the description of changelist 250 to change **#review-251** to **#append-251**, this changes the add mode for review-251 to append.
3. Shelf your files in pending changelist 250. Changelist 250 contains files A<sup>#1</sup>, and B<sup>#1</sup>. This appends the files to the review because that is the default add mode for review-251.
  - Version 4 of review 251 is created, containing files A<sup>#1</sup>, B<sup>#1</sup>, C<sup>#1</sup>, and D<sup>#1</sup>.

## Review creation and modification outside of P4 Code Review

You can create a review and add a changelist to a review from outside of P4 Code Review by adding keywords to the changelist description. The keyword is processed by P4 Code Review when the changelist is shelved for review, when files in a pending changelist that is associated with a review are reshelved, and when a changelist is committed. For details on using a keyword to create a review and add a changelist to a review, see ["Create a review" on page 669](#) and ["Add a changelist to a review" on page 670](#).

### Note:

P4 Code Review acts on the first valid keyword it finds in the changelist description, P4 Code Review then ignores any further valid keywords it finds in the description.

### Tip:

When the content of a review is changed, P4 Code Review checks to see which branches are in the new version of the review:

- **If a new branch was added to the review:**
  - Default reviewers on the new branch are added to the review.
  - Moderators from the added branch become moderators for the review alongside the existing moderators.
  - **Only if workflow is enabled:** if the new branch is associated with a workflow, the [workflow is merged](#) with the existing workflow. The most restrictive workflow is used for the review.
- **If a branch is no longer part of the review:**
  - Reviewers for the review are not changed.
  - Moderators from the removed branch no longer moderate the review.
  - **Only if workflow is enabled:** if the branch was associated with a workflow, the branch workflow is removed from the review.

The following examples show how adding keywords to the changelist descriptions works in practice. In each case, you are collaborating with other users to produce a single review in P4 Code Review:

### Basic workflow using #review and #append in the changelist description

1. You create a review with files A<sup>#1</sup>, B<sup>#1</sup>, and C<sup>#1</sup> by using `#review` in the changelist description.
2. Review-xxxx is created and `#review` is automatically replaced with `#review-xxxx` in the changelist description.
3. User-2 appends their changelist that contains files C<sup>#2</sup>, D<sup>#1</sup>, and E<sup>#1</sup> to review-xxxx by using `#append-xxxx` in the changelist description.

- Review-xxxx now contains files A<sup>#1</sup>, B<sup>#1</sup>, C<sup>#2</sup>, D<sup>#1</sup>, and E<sup>#1</sup>.
  - The default add mode for review-xxxx is now append because User-2 added the changelist to the review using `#append-xxxx`.
4. You edit files A<sup>#1</sup> and B<sup>#1</sup> in your changelist so they become A<sup>#2</sup> and B<sup>#2</sup>. The changelist description still contains `#review-xxxx`.
  5. You shelve your pending changelist which has `#review-xxxx` in the changelist description, this appends the files to the review because that is the default add mode for review-xxxx.
    - Review-xxxx now contains files A<sup>#2</sup>, B<sup>#2</sup>, C<sup>#2</sup>, D<sup>#1</sup>, and E<sup>#1</sup>.

## Advanced workflow using `#review` and `#append` in the changelist description

1. You create a review with files A<sup>#1</sup> and B<sup>#1</sup> by using `#review` in the changelist description.
2. Review-xxxx is created and `#review` is automatically replaced with `#review-xxxx` in the changelist description.
3. User-2 appends their changelist that contains file C<sup>#1</sup> to review-xxxx by using `#append-xxxx` in the changelist description.
  - Review-xxxx now contains files A<sup>#1</sup>, B<sup>#1</sup>, C<sup>#1</sup>.
  - The default add mode for review-xxxx is now append because User-2 added the changelist to the review using `#append-xxxx`.
4. User-3 adds their changelist that contains files A<sup>#2</sup> and D<sup>#1</sup> to review-xxxx by using `#review-xxxx` in the changelist description. This appends the files to the review because that is the default add mode for review-xxxx.
  - Review-xxxx now contains files A<sup>#2</sup>, B<sup>#1</sup>, C<sup>#1</sup>, and D<sup>#1</sup>.
5. You edit files A<sup>#2</sup> and B<sup>#1</sup> in your changelist so they become A<sup>#3</sup> and B<sup>#2</sup>, and you add file E<sup>#1</sup>. The changelist description still contains `#review-xxxx`.
6. You shelve your pending changelist which has `#review-xxxx` in the changelist description, this appends the files to the review because that is the default add mode for review-xxxx.
  - Review-xxxx now contains files A<sup>#3</sup>, B<sup>#2</sup>, C<sup>#1</sup>, D<sup>#1</sup>, and E<sup>#1</sup>.

## Advanced workflow using `#review`, `#append`, and `#replace` in the changelist description

1. You create a review with files A<sup>#1</sup> and B<sup>#1</sup> by using `#review` in the changelist description.
2. Review-xxxx is created and `#review` is automatically replaced with `#review-xxxx` in the changelist description.
3. User-2 appends their changelist that contains file C<sup>#1</sup> to review-xxxx by using `#append-xxxx` in the changelist description.
  - Review-xxxx now contains files A<sup>#1</sup>, B<sup>#1</sup>, C<sup>#1</sup>.
  - The default add mode for review-xxxx is now append because User-2 added the changelist to the review using `#append-xxxx`.

4. User-3 replaces review-xxxx with their changelist that contains files A<sup>#2</sup> and D<sup>#1</sup> to review-xxxx by using **#replace-xxxx** in the changelist description.
  - Review-xxxx now contains files A<sup>#2</sup> and D<sup>#1</sup>.
  - The default add mode for review-xxxx is now replace because User-3 added the changelist to the review using **#replace-xxxx**.
5. You edit files A<sup>#2</sup> and B<sup>#1</sup> in your changelist so they become A<sup>#3</sup> and B<sup>#2</sup>, and you add file E<sup>#1</sup>. The changelist description still contains **#review-xxxx**.
6. You shelve your pending changelist which has **#review-xxxx** in the changelist description, this replaces the files in the review because that is the default add mode for review-xxxx.
  - Review-xxxx now contains files A<sup>#3</sup>, B<sup>#2</sup>, and E<sup>#1</sup>.

## States

Reviews can be in one of several states. The biggest differentiator is whether the review's files have any outstanding, uncommitted changes or not.

Whenever a review state changes, an email notification is sent to all review participants, including:

- The review author
- Any user who comments on the review or its files
- Any user who has changed the review's state previously
- Any user who is [@mentioned](#), or a member of a group that is [@@mentioned](#) in the review's description or comments.

The review state is indicated by the **Review state** icon on the [Review page](#). The **Change state** button is used to change the state of a review, see "[Change review state](#)" on page 468.

Code reviews can be in one of the following states:

- **Needs review:** The review has started and the changes need to be reviewed.
- **Needs revisions:** The changes have been reviewed and the reviewer has indicated that further revisions are required.
- **Approved:** The review has completed. The changes may need to be committed. If the changes have been committed then this review will be Approved and closed, otherwise it will be Approved and open. See the note [below](#).
- **Rejected:** The review has completed. The changes are undesirable and should not be committed.
- **Archived:** The review has completed for now. However, it is neither rejected nor approved; it is simply put aside in case it is needed in the future.

### Note:

By default, when an **Approved** review is committed or updated, P4 Code Review changes the state to **Needs Review** if the files have been modified since the review was approved. Files are considered modified if the list of involved files changes, or if the file content or file-type changes.

If one or more files in a review has the filetype +k (ktext), this behavior is undesirable because the files will appear to be modified when the P4 Server replaces RCS keywords with their current values. See ["Unapprove modified reviews" on page 729](#) to see how to disable this behavior.

## Self-approval by review authors

By default, review authors can approve their own reviews. This behavior is based on P4 Code Review's [advisory nature](#).

Self-approval by authors can be prohibited on a project-by-project basis by specifying moderators for project branches (see ["State change restrictions with moderation" below](#)). However, authors who are moderators can self-approve their own reviews.

Administrators can configure P4 Code Review to prevent all self-approval by review authors. See ["Disable self-approval of reviews by authors" on page 676](#).

## State change restrictions with moderation

Typically, any authenticated user can change the state of a review (remember that the review state is merely advisory in most cases).

When you add a moderator to a project branch, only the moderators can approve or reject reviews restriction is enabled for a project branch. See ["Project settings" on page 478](#) for details on adding moderators to project branches.

Changing the state of any review associated with a moderated branch is restricted as follows:

Only moderators can approve or reject reviews on the branch.

To add moderators to a branch:

1. Click the **Manage moderators** button.
2. Choose the **Users** or **Groupstab**.
  - a. If you selected **Users**, start typing a the name of a user. Suggested users on the P4 Server will appear automatically.
  - b. If you selected **groups**, then all of the members of that group will have the same moderator privileges for that project branch. Use this option if you wanted to add several users at once.
3. Click Add.

After you finish setting up the branch and save the project, reviews on this branch can only be approved or rejected by moderators.

Moderators can also transition a review to any other state. Below is a summary of branch moderator roles and review state rules.

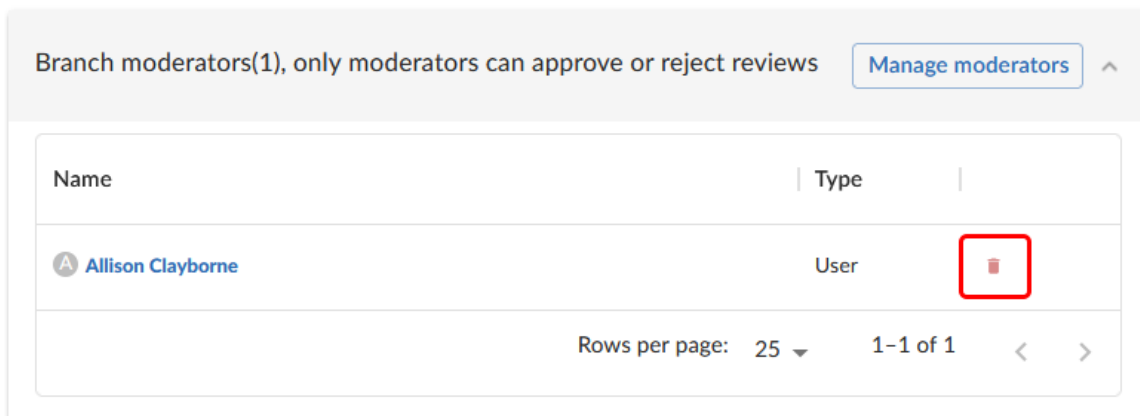
| Role                      | Allowed actions                                                                                                                                                                                                  | Restrictions                                                                                                                                             |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Moderators                | <p>Can move a review to any state.</p> <p>Can approve or reject reviews.</p> <p>Can prevent automatic approvals.</p>                                                                                             | <p>None (unless <code>disable_self_approve</code> is enabled for self-approval). For more information, see <a href="#">disable_self_approve</a>.</p>     |
| Authors (not moderators)  | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Needs review</li> <li>Needs revision</li> <li>Archived</li> </ul> <p>Can attach committed changelists.</p>                                    | <p>Cannot approve or reject reviews.</p>                                                                                                                 |
| Authors (also moderators) | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Approve</li> <li>Rejected</li> <li>Needs review</li> <li>Needs revision</li> <li>Archived</li> </ul> <p>Can attach committed changelists.</p> | <p>Cannot approve their own reviews if <code>disable_self_approve</code> is enabled. For more information, see <a href="#">disable_self_approve</a>.</p> |
| Project members           | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Needs review</li> <li>Needs revision</li> </ul> <p>Can attach committed changelists.</p>                                                      | <p>Cannot approve, reject, or archive reviews.</p>                                                                                                       |

| Role        | Allowed actions     | Restrictions                                                                                       |
|-------------|---------------------|----------------------------------------------------------------------------------------------------|
| Other users | None                | Cannot change review states.                                                                       |
| All roles   | Can start a review. | Cannot change state if it's not in their allowed states (for example, can't change from Rejected). |

#### Additional notes

- Moderators can prevent automatic approvals. For more information about automatically approving reviews using workflow rules see ["Workflow rules" on page 510](#).
- By default, if a review spans multiple branches with different moderators, only one moderator from any branch needs to approve it. You can change this (via a global setting) to require one moderator per branch. If a moderator belongs to multiple branches, one approval counts for each. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

To delete a moderator, click the delete icon against their name.



## State change restrictions without moderation

Changing the state of any review associated with a project that has no moderation is restricted as follows:

- Only members and the review author can change the state of a review or attach commits.
- If a user or group is added in ["Members or groups who can ignore ALL workflow rules" on page 748](#) list in the **Global Workflow** setting, they can transition the review state.

## Required reviewers

Reviews can optionally have required reviewers. When a review has required reviewers, the review cannot be approved until all required reviewers and required reviewer groups have up-voted the review. If the review is associated with a project that has assigned moderators, even the moderators cannot approve the review without up-votes from all required reviewers (but they can reject the review).

When a group is a required reviewer, it can be set to operate in one of two ways:

- **Require all:** all members of the group must up-vote the review to allow the review to be approved.
- **Require one:** at least one member of the group must up-vote the review to allow the review to be approved. If any member of the group down-votes the review, the review cannot be approved.

Required reviewers are expected to take greater care while performing a review than non-required reviewers, as their votes affect whether a review can be approved or not.

To edit the reviewers for a review and to change whether a reviewer is required or not, see ["Edit reviewers" on page 448](#).

### Note:

If a review involves a branch with assigned moderators, only a moderator can approve the review, even if all required reviewers have up-voted the review.

See ["Project settings" on page 478](#) for details on adding moderators to project branches.

## Change review state

The **Change state** button on the ["Review display" on page 411](#) page is used to manually change the state of a review.

To change the state of the review:

1. Navigate to the review you want to update.
2. Click the **Change State** button.
3. Select the new state from the dropdown menu.

The options available depend on the current state of the review and your user permissions:

- **Needs revision:** Select to request changes to the files in the review.
- **Needs review:** Select to request further review of the changes.
- **Approve** (only available if the voting requirements for the review are satisfied. For information on voting requirements, see ["Required reviewers" on page 457](#)): select to approve the review.
- **Commit** (only available for [pre-commit](#) reviews that have been approved): Select to commit the review.
- **Approve and commit** (only available for unapproved [pre-commit](#) reviews when the voting requirements for the review are satisfied. For information on voting



requirements, see ["Required reviewers" on page 457.](#)): Select to approve and commit the review in a single step. See ["Approve and commit" below.](#)

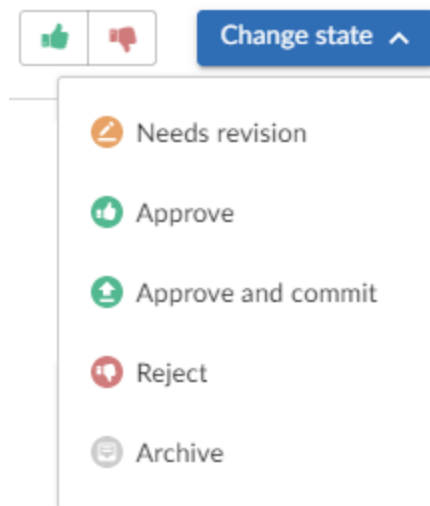
- **Reject:** Select to reject the review.
- **Archive:** Select to archive the review.

## Approve and commit

**Approve and commit** is available in the dropdown menu for unapproved pre-commit reviews, this enables you to approve and commit a review in a single step if required.

### Note:

- By default, any authorized user can commit a review. P4 Code Review can be configured to restrict this behavior so that only the review author can commit the review, see ["Disable commit" on page 695](#) for details.
- **Approve and commit** is only available for unapproved [pre-commit](#) reviews when the voting requirements for the review are satisfied. For information on voting requirements, see ["Required reviewers" on page 457.](#)



### Note:

If P4 Code Review is configured to prevent approval of reviews with open tasks and a review has open tasks, the **Approve**, and **Approve and commit** options will not be available for the review. This option is configured by an administrator. See ["Disable approve for reviews with open tasks" on page 676.](#)

To approve, or approve and commit a review with open tasks, you must address the tasks first and then set them to **Task Addressed**, or **Not a Task**. See ["Set a task to Task addressed or Not a task" on page 333](#) for details.

1. Select **Approve and commit** from the dropdown menu.
2. The **Commit Review** dialog is displayed.

### Commit Review

Use the new P4 Core package names for specifying dependencies.

Also update the copyright and release information on the meta files.

☒ [job083648](#) fixed Need to rename the packages to be helix-insights rather than perforce-...

Job Status on Commit

Fixed ▾

☐ Remove pending changelists

Approve and Commit

Cancel

3. Edit the review description if required.
4. Select which jobs should be associated with the review, and specify the job status on commit.
5. Select **Remove pending changelists**, P4 Code Review will attempt to automatically clean up any changelists left behind after the review has been committed, including removing any shelved files. This option can be removed by an administrator, see ["Review cleanup" on page 666](#) for details.
6. Click **Approve and commit** to approve the review and commit the associated files.

**Note:**

By default, P4 Code Review records that you committed the review on behalf of the review's author. This can be configured by an administrator to only credit the committer and not the review author, see ["Commit credit" on page 582](#) for details.

## 6 | Groups

*Groups* are a feature of P4 Server that makes it easier to manage permissions for users. P4 Code Review can use groups to coordinate review activities and responsibilities. This chapter covers how to manage groups in P4 Code Review, including how to:

- "Add a group" below
- "Edit a group" on page 476
- "Delete a group" on page 476

See "Groups" on page 352 for an introduction to P4 Code Review groups.

### Add a group

#### Important:

You must have *super* privileges in P4 Server (**p4d**), or have *admin* privileges in **p4d** version 2012.1 or later, to create a group. If you do not have sufficient permissions, P4 Code Review does not display the **Add Group** button.

1. To view a list of groups, click **Groups** in the menu.
2. Click the **+ Add Group** button.

The Add Group **Settings** tab is displayed:

#### Add Group

The screenshot shows the 'Add Group' form with the following elements:

- Settings** (active tab) and **Notifications** (inactive tab).
- Name**: A text input field with a red border.
- Description**: A large text area.
- Owners**: A button labeled "Add an Owner".
- Members**: A button labeled "Add a User or Subgroup".
- Save** and **Cancel** buttons at the bottom.

3. Provide a name for the group.

4. **Optional:** provide a description.
5. **Optional:** specify an owner. This field auto-suggests users within P4 Server as you type.

Group owners are not members of their groups. A group owner can be added as a member of a group.

Once specified, modifying the group's definition is restricted to group owners and users with *super* privileges in P4 Server.

If you do not specify an owner, you must specify at least one member (below).
6. **Optional:** specify group members. This field auto-suggests projects, groups, and users within P4 Server as you type (up to a combined limit of 20 entries).

If you specify a project, the project's members become members of the group. If you specify a group, that group becomes a sub-group of your new group, and all of its members (and members of any of its sub-groups) become members of your new group.

If you do not specify any members, you must specify at least one owner (above).
7. Choose one the following options:
  - Click **Save** to finish adding the group. By default group members will be emailed when a new review is requested.

**Note:**

The **Save** button is disabled if any required fields are empty.

or

- **Optional:** configure email notifications for the group in the Add Group, Notifications tab. See the next step for details.
8. **Optional:** click the **Notifications** tab to configure group email notifications.

## Add Group

[Settings](#)
[Notifications](#)
☐ Use mailing list instead of notifying by individual group member's emails (must add email address)

 Email Notifications ☒ Email members when a new review is requested

☐ Email members when a change is committed

Adjust when notifications are sent to group members (use mailing list address to enable).

Email group members when:

 Check all ☐

The group is a member of a project, and

 a review is started in the project ☒

 a review or change is committed ☒

The group is a reviewer on a review, and

 files in the review are updated ☒

 tests on the review have finished ☒

 a vote is cast on a review ☒

 the state of the review changes ☒

 someone joins or leaves the review ☒

 a review or change is committed ☒

 a comment is made on the review ☒

 a comment on the review is updated ☒

The group is a moderator on a project, and

 files in the review are updated ☒

 tests on the review have finished ☒

 a review or change is committed ☒



9. **Optional:** add a group mailing list address by selecting **Use mailing list instead of notifying by individual group member's emails (must add email address)** and entering a valid group email address.

### Note:


- **Group mailing list enabled:** notifications are sent to the group email address.


When a group mailing list is enabled it overrides the group member's preference set for the `notify_self` configurable in the [SWARM\\_ROOT/data/config.php](#) file. So even when the `notify_self` configurable is set to false the group member will receive an email notification.

There is a caveat that if a user is a group member and the same user is added individually to the review and has set the `notify_self` configurable to true, the user receives two email notifications.

- **Group mailing list disabled:** notifications are sent to the group members individual email addresses.

The format of the email address is validated as you type.

agroup@example.c 

agroup@example.com 

10. Group members can be notified when a member of the group starts a review. Group members can be notified when a change is committed by, or on behalf of, a changelist owner who is also a member of this group. These settings are always available even if the group mailing list is not enabled.

Select which actions send a notification to the group:

- **Email members when a review is requested:** When any member of this group creates a review, this group will be notified.
- **Email members when a change is committed:** When a change is committed into Perforce, if the owner of the changelist is a member of this group, this group will be notified.

**Tip:**

When a user commits a changelist in P4 Code Review, it is committed on behalf of the changelist owner. If the changelist owner is a member of this group, this group will be notified.

**Note:**

Members of your group may receive notification emails even if group notifications are disabled as they may be members of a project, or follow a project or user, or P4 Code Review's review daemon functionality may be enabled. See [Notifications](#) for details.

11. Group notification settings allow you to configure which notifications are sent to the group mailing list when events occur within P4 Code Review (the group mailing list must be enabled). This allows you to limit the number of emails sent to the group mailing list. These settings apply across all projects.

**Note:**

The following group notification settings are only available if the group mailing list address is configured.

**Note:**

Defaults for these options are configured by the P4 Code Review administrator, and they may force some of these options to on or off. See ["Global settings" on page 654](#) for how this is configured.

Adjust when notifications are sent to group members.

Email group members when:
Reset to default

Check all ☐

|                                            |                                     |
|--------------------------------------------|-------------------------------------|
| The group is a member of a project, and    |                                     |
| a review is started in the project         | <input checked="" type="checkbox"/> |
| a review or change is committed            | <input checked="" type="checkbox"/> |
| The group is a reviewer on a review, and   |                                     |
| files in the review are updated            | <input checked="" type="checkbox"/> |
| tests on the review have finished          | <input checked="" type="checkbox"/> |
| a vote is cast on a review                 | <input checked="" type="checkbox"/> |
| the state of the review changes            | <input checked="" type="checkbox"/> |
| someone joins or leaves the review         | <input checked="" type="checkbox"/> |
| a review or change is committed            | <input checked="" type="checkbox"/> |
| a comment is made on the review            | <input checked="" type="checkbox"/> |
| a comment on the review is updated         | <input checked="" type="checkbox"/> |
| The group is a moderator on a project, and |                                     |
| files in the review are updated            | <input checked="" type="checkbox"/> |
| tests on the review have finished          | <input checked="" type="checkbox"/> |
| a review or change is committed            | <input checked="" type="checkbox"/> |

Save
Cancel

Toggle notifications for each event on or off to control whether the group receives an email when that event occurs.

Clicking **Reset to default** resets the options back to system defaults.

12. Click **Save**.

**Note:**

The **Save** button is disabled if any required fields are empty.

## Edit a group

### Important:

You must be an owner of the group, or be a user with *super* privileges in P4 Server, to edit the group.

1. Visit the group page you want to edit.
2. Click the **Settings** tab for the group to edit group details. If you do not have permission to edit a group, this tab does not appear.
3. Edit group details as required. See ["Add a group" on page 471](#) for more details.
4. Click the **Notifications** tab for the group to edit group notifications. If you do not have permission to edit a group, this tab does not appear.
5. Edit group notifications as required. See ["Add a group" on page 471](#) for more details.
6. Click **Save**.

### Note:

When a group is edited, the list of associated reviews is not immediately updated to match any changes in membership. The association with a review does not change until the review is updated.

## Delete a group

### Important:

You must be an owner of the group, or be a user with *super* privileges in P4 Server, to delete the group.

1. Visit the group page you want to delete.
2. Click the **Settings** tab for the group. If you do not have permission to edit a group, this tab does not appear.
3. Click **Delete**.  
A tooltip is displayed to confirm that you want to delete this group.
4. Click **Delete** to confirm.



# 7 | Projects

A P4 Code Review project is a group of P4 Server users who are working together on a specific codebase, defined by one or more branches of code, along with a job filter, and automated test integration. This chapter covers P4 Code Review's project management capabilities, including:

- ["Add a project" below](#)
- ["Project settings" on the next page](#)
- ["Edit a project" on page 497](#)
- ["Membership" on page 497](#)
- ["Delete a project" on page 505](#)

See ["Projects" on page 360](#) for an introduction to P4 Code Review projects.

## Add a project

To learn how to create a project in P4 Code Review, watch this video or review the instructions following the video.

1. On the P4 Code Review **Projects** page, click the **+ Add Project** button.  
The **Add Project** page dialog is displayed.

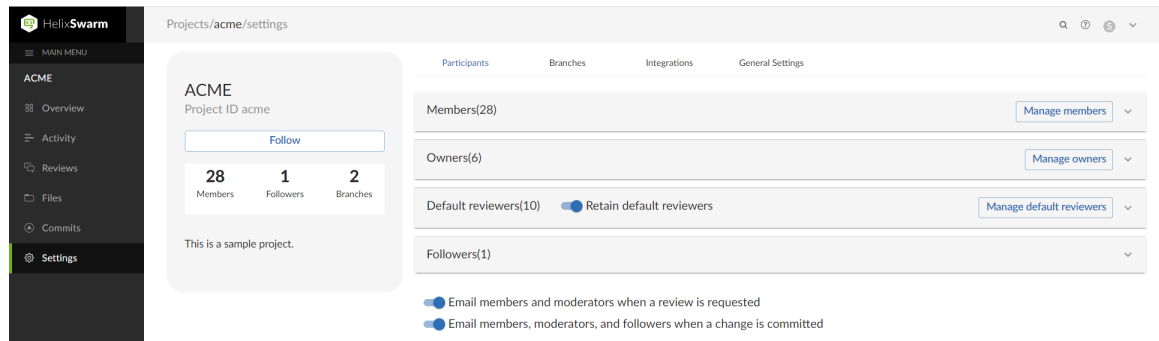
### Note:

- The ability to add projects can be [limited to administrators only](#), or [limited to members of specific groups](#). When limited, users who are not administrators, or a member of the specified group, will not see the **+ Add Project** button.
- By default, any member of a project can edit the project's configuration. Administrators can configure Swarm to prevent changes to the project's name and branch definition(s).

2. Provide a name for the project and click the **Create** button.

A project summary with Participants tab, Branches tab, Integrations tab, and General Settings

tab is displayed.



For more information on how to use the **Settings** page, see ["Project settings"](#) below.

## Project settings

This section describes how to use the new project settings page to view and edit project settings for a project. For more information about how to add a project, see ["Add a project"](#) on the previous page.

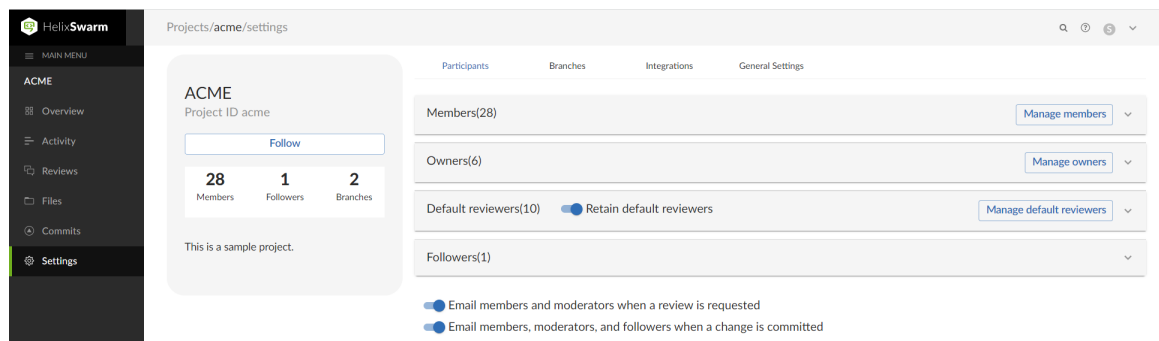
### Note:

Modifying the project's definition is restricted to project owners and administrators (users with *admin*-level or *super*-level privileges in P4 Server). See ["Membership"](#) on page 497 for more details.

To view or edit project settings for a specific project:

1. Click **Projects** in the menu.
2. Click on a project name displayed in **My Projects** list.
3. Click **Settings** in the menu.

A project summary with Participants tab, Branches tab, Integrations tab, and General Settings tab is displayed.



The following information is displayed in the project summary:

- The name of the project.
- The unique associated project identifier.

- A **Follow** button that you can click to follow the project. This button is not available if you are a member of the project.
- Total number of members, followers, and branches associated with the project.
- A short description about the project.

## Participants tab

To learn how to customize participants for a P4 Code Review project, watch this video or review the instructions following the video.

The **Participants** tab shows the following details:

### Members

Displays a list of all the project members. This includes users, groups, and projects.

Participants

Branches

Integrations

General Settings

Members(28)

Manage members

| Name                                                   | Type |             |
|--------------------------------------------------------|------|-------------|
| <div><div>A</div><div>Adena Waters</div></div>         | User | <div></div> |
| <div><div>A</div><div>Adolfo Whipple</div></div>       | User | <div></div> |
| <div><div>A</div><div>Adrienne Fitzpatrick</div></div> | User | <div></div> |
| <div><div>A</div><div>Albina Bona</div></div>          | User | <div></div> |
| <div><div>B</div><div>Becky Hoesly</div></div>         | User | <div></div> |

To add a project member:

1. Click **Manage members** button.
2. Select the **Users**, **Groups**, or **Projects** tab that you wish to add to the project members list.
3. Type in to search the name of a user, group, or project and click **Add** button.

To delete a project member, click the delete icon against their name.

### Owners

Displays a list of all the project owners.













#### Note:

Only users can be added as project owners.

[Participants](#) [Branches](#) [Integrations](#) [General Settings](#)

Members(28) [Manage members](#) ▾

Owners(6) [Manage owners](#) ▲

| Name                                                                                                |                                                                                     |
|-----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|  Aaron Gleber      |  |
|  Abigail Germann   |  |
|  Chassidy Shorts   |  |
|  Domanique Pantoja |  |
|  Suzie Swansbrough |  |
|  Vernon Renno      |  |

Rows per page: 25 ▾ 1-6 of 6 < >

To add a project owner, click **Manage owners** button. Type in the name of the user you wish to add to the project owners list. The field auto-suggests users within P4 Server as you type. Click the **Add** button to add a user to the project owners list.

To delete a project owner, click the delete icon against their name.

### Important:

The P4 Code Review user cleanup API requires the P4 Code Review user to have ownership privileges on the affected projects and groups. If the P4 Code Review user has insufficient permissions, cleanup for those P4 projects or groups will fail. To resolve this, add *Super* or *admin* privileges to the P4 Code Review user or ensure that the user is an owner of all relevant projects and groups.

## Default reviewers

Displays a list of all the default reviewers. A default reviewer can be a user or a group.

Participants   Branches   Integrations   General Settings

---

















Members(28) Manage members ▾

---

Owners(6) Manage owners ▾

---

Default reviewers(9) ☐ Retain default reviewers Manage default reviewers ⌵

| Name                                                                                                | Vote Requirement ↑                                                                              | Type  |
|-----------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------|-------|
|  Abbie Korewdit    | Required     | User  |
|  Ada Garrison      | Optional     | User  |
|  Avril Cottrill    | Optional     | User  |
|  Bennett Duriga    | Optional     | User  |
|  Caitlin Dom       | Required     | User  |
|  Suzie Swansbrough | Optional     | User  |
|  Small             | Require one  | Group |
|  Small1          | Optional   | Group |

To add a new default reviewer:

1. Click **Manage default reviewers** button.
2. Select the **Users** or **Groupstab** that you wish to add to the default reviewers list.
3. Type in to search the name and click **Add** button.
4. Click the drop-down box against the user or group name and select your desired option.

To edit your default reviewer preferences for an existing reviewer, click the drop-down box against the user name and select your desired option.

For a user, you can set the vote requirement to **Optional** or **Required**. When a group is a default reviewer, it can be set to operate in one of three ways:

- **Optional:**  
No members of the group are required to approve the review.
- **Require one** indicated by a star badge with a 1:  
At least one member of the group must up-vote the review to allow the review to be approved. If any member of the group down-votes the review, the review cannot be approved.
- **Required** indicated by a star badge:  
All members of the group must up-vote the review to allow the review to be approved.

To delete a default reviewer, click the delete icon against their name.

### Default reviewers for reviews in multiple projects or branches:

When a review belongs to more than one project or branch:

- The default reviewers from all the related projects and branches are combined and added to the review.
- If a reviewer has different vote requirements in different places, the strictest setting is used.

For example:

- In Project-A, Reviewer X is optional.
- In Project-B, Reviewer X is optional.
- In the project branch, Branch-b, Reviewer X is required.

As a result, Reviewer X will be added as a required reviewer on the review.

### Mentioning default reviewers

If users or groups are [@mentioned](#) in a new changelist description that includes [#review](#), they will be added to the review as reviewers. If any of these reviewers are already specified as default reviewers they will not be added to the review again, the reviewer's most restrictive reviewer option is used for the review.

To retain default reviewers, toggle the **Retain default reviewers** switch. For more details about retaining default reviewers, see ["Membership" on page 497](#).

## Followers

Displays a list of all the followers.

Q ⓘ Ⓔ ⌵

Participants

Branches

Integrations

General Settings

Members(28)

Manage members ⌵

Owners(6)

Manage owners ⌵

Default reviewers(10)

☒ Retain default reviewers

Manage default reviewers ⌵

Followers(1)

⌵

Name

Yael Larez

Rows per page: 25 ⌵ 1-1 of 1 < >

☒ Email members, moderators, and followers when a review is requested

☒ Email members, moderators, and followers when a change is committed

By default, project members and moderators are notified when a new review is started. Project members, moderators, and followers are notified when a change is committed.

Toggle the following switches at the bottom of the **Participants** tab to ensure which actions send a notification:

- **Email members, moderators, and followers when a review is requested:** When a new review is requested for the project, all project members and moderators of the project are added to the email notification list.
- **Email members, moderators, and followers when a change is committed:** When a change is committed for the project, all project members, project moderators and project followers of this project are added to the email notification list.

**Note:**

- When a group is a project member or project moderator, all of the members of that group are notified using the same logic as for individual project members and moderators.
- Any "[Links in descriptions and comments](#)" on page 387 users and groups, or users and groups who are explicitly added to a review or changelist, will receive notifications even if new review/committed review notifications are disabled.

## Branches tab

The Branches tab enables you to add a branch and configure the following options:

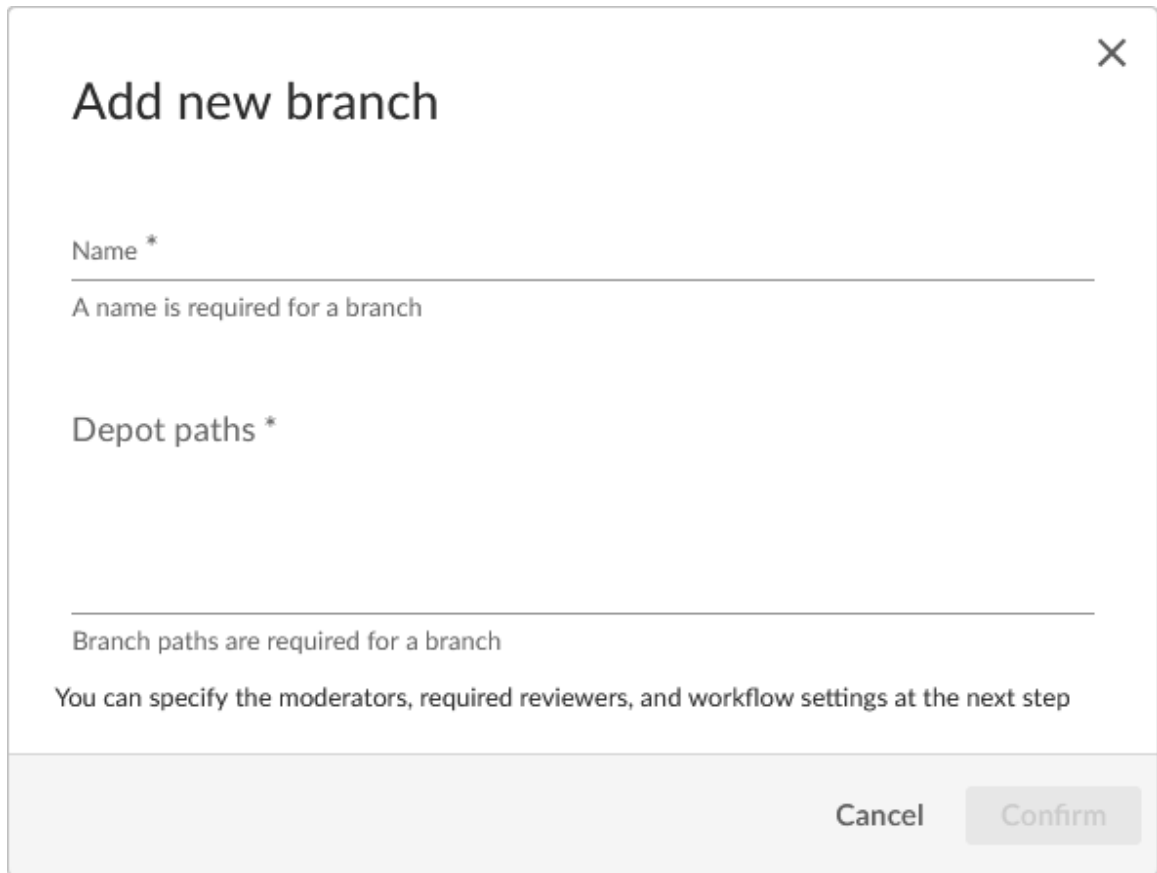
- Associate a workflow with the project branch
- Set minimum up votes
- Manage moderators
- Manage branch specific default reviewers

To learn how to create branches for a P4 Code Review project, watch this video or review the instructions following the video.

## Add a branch

To add a new project branch:

1. Click the **Add new branch** button. This opens an **Add new branch** dialog.



**Add new branch**

Name \*

A name is required for a branch

Depot paths \*

Branch paths are required for a branch

You can specify the moderators, required reviewers, and workflow settings at the next step

Cancel Confirm

2. Enter a short **Name** for your branch.
3. Enter one or more branch paths in the **Depot paths** field using depot syntax, one path per line. For more information on branch paths, see ["Example branch paths" below](#).

**Important:** Branch paths are case sensitive.

4. Click the **Confirm** button.

A new branch is created.

### Example branch paths

If you have entered multiple branch paths into the Depot paths field, these paths are processed in order, starting with the first path in the list.

Branch paths, and files can be excluded by putting a minus symbol - at the start of the path. In the following example:

- The first path includes all of the directories and files under `//depot/main/swarm/` in the project branch.
- The second path excludes all of the files in `-//depot/main/swarm/test/` from the project branch.
- The third path includes the `ResultSummary.html` file from the previously excluded `//depot/main/swarm/test/` directory.



```
//depot/main/swarm/...
-//depot/main/swarm/test/...
//depot/main/swarm/test/ResultSummary.html
```

### Wildcards in branch paths

There are two kinds of wildcards that are supported in project branch paths: `...` and `*`.

Only use the P4 Server wildcard `...` at the end of a branch path.

Branch paths, and files are not checked to see if they are valid when you save the branch:

- If you enter an invalid path ending in the wildcard `...`, the path will not be displayed in the [project file browser](#) until the path is created. This allows you to specify a path before it has been created.
- If you enter a path that ends with a file that has not been committed, or a non-existent file, P4 Code Review displays a 404 error when you navigate the path to the file with the [project file browser](#).

The wildcard `*` is supported within the branch path itself. For example, `VersionA` and `VersionB` could be replaced with the wildcard `*` in the following paths:

```
//ProjectY/UserZ/VersionA/LibA/SubLib1/Data1/...
//ProjectY/UserZ/VersionA/LibA/SubLib2/...
//ProjectY/UserZ/VersionB/LibA/SubLib1/Data1/...
//ProjectY/UserZ/VersionB/LibA/SubLib2/...
```

Resulting in the following filepaths with wildcards.

```
//ProjectY/UserZ/*/LibA/SubLib1/Data1/...
//ProjectY/UserZ/*/LibA/SubLib2/...
```

### Handling new sub-directories

If new sub-directories are added or removed after the initial branch paths have been created, you can update branch paths by clicking the **Expand wildcards** button. This will expand all `*` wildcard paths to the matching single level sub-directories path.

Participants
Branches
Integrations
General Settings

## Associated Branches

Add new branch

|           |                    |
|-----------|--------------------|
| CANDIDATE | Expand wildcards ▾ |
| DEV       | Expand wildcards ▾ |
| MAIN      | Expand wildcards ▾ |

Alternatively, you can expand the wildcards by editing and re-saving a branch to trigger expansion.

**Tip:** The expand wildcards feature can match a large set of directories, including subdirectories and the performance of P4 Code Review should remain unaffected. However, data sets of tens of several thousand may impact performance. We have tested performance when expanding up to 30,000 paths and found no issues.

### Supported wildcard formats

The following are examples where `*` wildcards are supported:

- `//depot/Project/*/src/...`
- `//depot/Project/*`
- `//depot/Project/MA*`
- `//depot/Project/REL*/src`
- `//depot/Project/R*/src`
- `//depot/Project/*/src`
- `//depot/*/*/src`
- `//depot/*/*/*`
- `//depot/*/MAIN/*`
- `//depot/Project/REL*/s*`

The following are examples where `*` wildcards are not supported.

The `*` at the start of the path is not allowed, it is only allowed in the middle or the end of a branch path:

- `*//depot/Jam/*/src/...`

The `*` is only allowed at the end of text, for example `RHEL*`

- `//depot/Jam/R*EL/src`
- `//depot/Jam/*HEL/src`

## Edit a branch

Once you have created a branch, additional options become available for managing the branch further. Click on the name of the branch to expand the branch pane to view the additional options. All settings described in this section are optional.

Participants
Branches
Integrations
General Settings

Associated Branches
Add new branch

Main
^

Name \*
Main
The branch name

Workflow
Minimum up votes
Inherit from project
2

Depot paths \*
//depot/main/swarm/...
-//depot/main/swarm/test/...
//depot/main/swarm/test/ResultSummary.html

Branch moderators(1), only moderators can approve or reject reviews
Manage moderators

Default reviewers(1)
Retain default reviewers
Manage default reviewers

Delete branch

### Workflow

You can associate a workflow with the project branch. If you have disabled the workflow feature, this option will not be displayed.

To associate a workflow with a project branch, select one from the **Workflow** dropdown list, or enter the workflow name in the search field. You will only be able to see the workflows that you own or shared workflows. When a workflow is associated with a project, the workflow is used for all of the branches in that project.

To use the parent project workflow, select **Inherit from project** from the **Workflow** dropdown list. This option is selected by default when you create a new branch. If the parent project is set to **No workflow**, the branch will use the global workflow rules. When a project branch is associated with a workflow, the workflow of the parent project is ignored and the branch workflow is used.

To remove the existing workflow without replacing it with another workflow, select **No Workflow** from the **Workflow** dropdown list. This is the default when you create a new project.

For more information about workflows and how project workflows interact with branch workflows, see ["Workflow basics" on page 512](#).

### Minimum up votes

Use **Minimum up votes** to set the minimum number of up votes required for reviews in this branch.

A review cannot be approved until all of the [Required reviewers](#) have voted up the review and the **Minimum up votes** specified has been satisfied.

When this setting is defined at the project level, minimum number of up votes is automatically applied to all branches. However, if you set **Minimum up votes** to **1** or higher on a specific branch, that branch will use its own setting and ignore the project-level setting.

To use the project-level value, set **Minimum up votes** to **Inherit from project**. This is set by default for new branches.

To override the project setting and use a custom value for a certain branch, set **Minimum up votes** to **1** or higher on that branch.

### Notes on minimum up votes

- If a review includes changes across multiple projects or branches, the minimum number of up votes on each project or branch must be met before the review can be approved.
- Required reviewers are included when counting up votes.
- When the workflow rule **Count votes up from** is set to **Members** for a project or branch, only up votes from members belong to that project or branch will count toward the minimum up votes requirement. For more information on the **Count votes up from** rule, see ["Workflow rules" on page 510](#).

If workflows are disabled, all votes are counted, not just votes from project members.

**Restriction:** If the **Minimum up votes** requirement is higher than the total number of reviewers on a review, it will be impossible to approve it, even if every reviewer votes to approve. Take this into consideration when setting **Minimum up votes**.

### Troubleshooting: Commits not displayed in the top level view

The [Project Commits tab](#) can fail to show some P4 Server commits in the top level view. However, the branch views for individual project display the commits correctly.

The **Project Commits** tab top level client view is made up of all of the branches of the project.

For example, Project Alpha has the following branches:

Branch: QA

- `//depot/alpha/dev/QA/...`

Branch: Dev

- `//depot/alpha/dev/...`
- `-//depot/alpha/dev/QA/...`

The **Project Commits** tab view displays the branches in the order they were created, going through each path from top to bottom within each branch.

For example, in the Alpha project:

- The first path shows everything under `//depot/alpha/dev/QA/` in the top-level commits view.
- The second path shows everything under `//depot/alpha/dev/` in the top-level commits view.
- The third path removes (excludes) everything under `//depot/alpha/dev/QA/` from the top-level commits view.

As a result, commits to `//depot/alpha/dev/QA/` that should appear in the QA branch do not show up in the top-level Project Commits tab.

To avoid this problem, and ensure every branch is displayed in the top-level Project Commits tab, use a simple branch structure and plan the branch order as you create them.

In the example above, if you created the Dev branch first, and then created the QA branch, `//depot/alpha/dev/QA/` would not be excluded, and the commits would show up correctly in the Project Commits tab.

### Branch moderators

Only moderators can approve or reject reviews on the branch.

To add moderators to a branch:

1. Click the **Manage moderators** button.
2. Choose the **Users** or **Groupstab**.
  - a. If you selected **Users**, start typing a the name of a user. Suggested users on the P4 Server will appear automatically.
  - b. If you selected **groups**, then all of the members of that group will have the same moderator privileges for that project branch. Use this option if you wanted to add several users at once.
3. Click Add.

After you finish setting up the branch and save the project, reviews on this branch can only be approved or rejected by moderators.

Moderators can also transition a review to any other state. Below is a summary of branch moderator roles and review state rules.

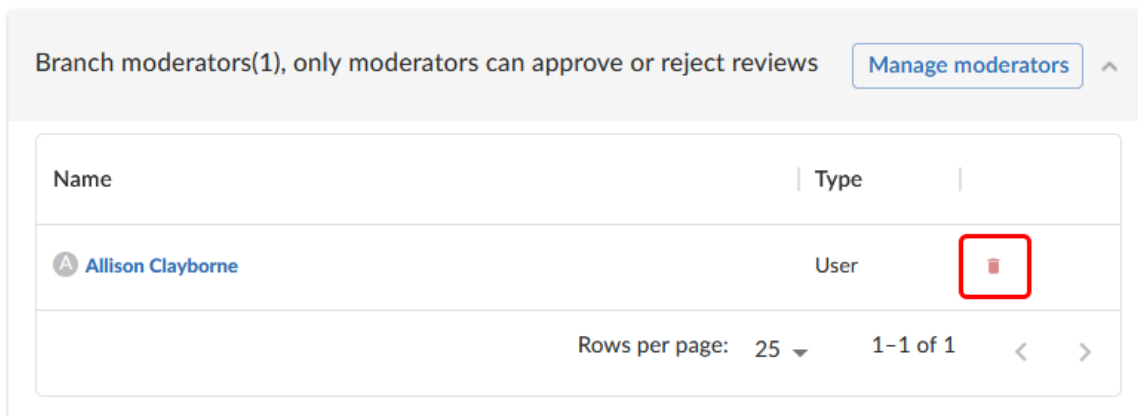
| Role                      | Allowed actions                                                                                                                                                                                                      | Restrictions                                                                                                                                       |
|---------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------|
| Moderators                | <p>Can move a review to any state.</p> <p>Can approve or reject reviews.</p> <p>Can prevent automatic approvals.</p>                                                                                                 | None (unless <code>disable_self_approve</code> is enabled for self-approval). For more information, see <a href="#">disable_self_approve</a> .     |
| Authors (not moderators)  | <p>Can change state to:</p> <ul style="list-style-type: none"><li>■ Needs review</li><li>■ Needs revision</li><li>■ Archived</li></ul> <p>Can attach committed changelists.</p>                                      | Cannot approve or reject reviews.                                                                                                                  |
| Authors (also moderators) | <p>Can change state to:</p> <ul style="list-style-type: none"><li>■ Approve</li><li>■ Rejected</li><li>■ Needs review</li><li>■ Needs revision</li><li>■ Archived</li></ul> <p>Can attach committed changelists.</p> | Cannot approve their own reviews if <code>disable_self_approve</code> is enabled. For more information, see <a href="#">disable_self_approve</a> . |
| Project members           | <p>Can change state to:</p> <ul style="list-style-type: none"><li>■ Needs review</li><li>■ Needs revision</li></ul> <p>Can attach committed changelists.</p>                                                         | Cannot approve, reject, or archive reviews.                                                                                                        |

| Role        | Allowed actions     | Restrictions                                                                                       |
|-------------|---------------------|----------------------------------------------------------------------------------------------------|
| Other users | None                | Cannot change review states.                                                                       |
| All roles   | Can start a review. | Cannot change state if it's not in their allowed states (for example, can't change from Rejected). |

### Additional notes

- Moderators can prevent automatic approvals. For more information about automatically approving reviews using workflow rules see ["Workflow rules" on page 510](#).
- By default, if a review spans multiple branches with different moderators, only one moderator from any branch needs to approve it. You can change this (via a global setting) to require one moderator per branch. If a moderator belongs to multiple branches, one approval counts for each. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

To delete a moderator, click the delete icon against their name.



### Default reviewers

You can add default reviewers in the **Default reviewers** pane. To add a default reviewer:

1. Click the **Manage default reviewers** button.
2. Choose the **Users** or **Groupstab**.
  - a. If you selected **Users**, start typing a the name of a user. Suggested users will appear automatically.
  - b. If you selected **groups**, then all of the members of the selected group will be added as a

default reviewer.

**Restriction:** You can add up to 25 users or groups combined.

3. Click Add.

From now on, each new review on this branch will automatically include these reviewers.

To keep default reviewers from being removed on reviews for this branch, turn on **Retain default reviewers**. For more information about retained default reviewers, see [Retain default reviewers](#).


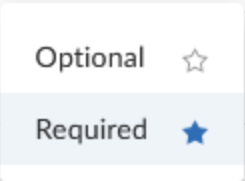
Default reviewers(4) ☒ Retain default reviewers

After you add default reviewers, you can control whether their votes are required or optional for approving a review. This is done using the **Vote Requirement** column.

**Tip:** Click on the Default reviewers pane to expand the section and view the table of default reviewers.

For **users**, select on of the following vote requirements.






- **Optional** - This users vote is not required to approve a review.
- **Required** - This users vote is required to approve a review

| Vote Requirement |                                                                                     | Type |
|------------------|-------------------------------------------------------------------------------------|------|
| Required         |  | User |
| Optional         |  | User |
| Optional         |                                                                                     | User |

For groups, select on of the following vote requirements.

- **Optional** - The vote from users in this group is not required to approve a review.
- **Required one** - At least one member of the group must approve the review, and no member votes down the review either.
- **Required all** - All members of the group must vote up the review to approve it.



| Vote Requirement |                                                                                                                        | Type ↑ |
|------------------|------------------------------------------------------------------------------------------------------------------------|--------|
| Optional         |  ▲                                    | Group  |
| Required         | <div> Optional  </div>                | User   |
| Optional         | <div> Require one <sup>1</sup> </div> | User   |
| Optional         | <div> Require all  </div>             | User   |
| Optional         |  ▼                                    | User   |

#### Default reviewers for reviews in multiple projects or branches:

When a review belongs to more than one project or branch:

- The default reviewers from all the related projects and branches are combined and added to the review.
- If a reviewer has different vote requirements in different places, the strictest setting is used.

For example:

- In Project-A, Reviewer X is optional.
- In Project-B, Reviewer X is optional.
- In the project branch, Branch-b, Reviewer X is required.

As a result, Reviewer X will be added as a required reviewer on the review.

#### Mentioning default reviewers

If users or groups are [@mentioned](#) in a new changelist description that includes [#review](#), they will be added to the review as reviewers. If any of these reviewers are already specified as default reviewers they will not be added to the review again, the reviewer's most restrictive reviewer option is used for the review.

To remove a user or group as a default reviewer, click the delete icon in their row of the table.

**Important:** If a default reviewer is deleted from P4 Server they will not be added to new reviews.

## Integrations tab

**Important:**

The project level test and deploy code features will be deprecated in a later P4 Code Review release. We recommend you use test integration to automatically deploy code within a review. For more information, see ["Add a test" on page 528](#).

**Tip:**

The project **Automated Tests** and **Automated Deployment** details are hidden from project members unless they are an owner or an administrator. This enables project members to check the project settings but not change them.

In the Integrations tab you can configure **Automated tests** and **Automated deployment** options.

**Note:**

You can edit and save all options for automated tests and automated deployment when each of their toggles is disabled. However, the new updates take effect only when the toggles are enabled.

To learn how to enable automated tests and deployment for a P4 Code Review project, watch this video or review the instructions following the video.

### Automated tests

To enable or disable running automated tests for a project when reviews are created or updated, slide **Automated tests** toggle. When you wish to enable automated tests, enter a URL that triggers a test suite execution. Use the special arguments described in the dialog to help compose a URL that informs your test suite about important details.

Select the format of the POST parameters, either **URL Encoded** or **JSON Encoded**.

- **URL Encoded:** POST parameters are parsed into name=value pairs.
- **JSON Encoded:** parameters are passed raw in the POST body.

For more details about automated tests, see ["Automated testing for reviews" on page 563](#).

### Automated deployment

To enable or disable deploying a project when reviews are created or updated, slide the **Automated deployment** toggle. When you wish to enable automated deployment, enter a URL that triggers a deployment of the project's code. Use the special arguments described in the dialog to help compose a URL that informs your deployment program with important details. For more details about automated deployment, see ["Automatically deploy code within a review" on page 87](#).

## General Settings tab

To learn how to customize general settings of a P4 Code Review project, watch this video or review the instructions following the video.

In the General Settings tab, you can configure the following options:

1. Specify a project name.
2. Add a short description about the project.
3. Select a **Workflow** to associate with the project. Your selection is auto saved.

**Note:**

The **Workflow** option is not displayed if the workflow feature is disabled.

To remove the existing workflow without replacing it with another workflow, select **No Workflow** from the **Workflow** dropdown list. This is the default when you create a new project.

You can associate a workflow with the project branch. If you have disabled the workflow feature, this option will not be displayed.

To associate a workflow with a project branch, select one from the **Workflow** dropdown list, or enter the workflow name in the search field. You will only be able to see the workflows that you own or shared workflows. When a workflow is associated with a project, the workflow is used for all of the branches in that project.

To use the parent project workflow, select **Inherit from project** from the **Workflow** dropdown list. This option is select by default when you create a new branch. If the parent project is set to **No workflow**, the branch will use the global workflow rules. When a project branch is associated with a workflow, the workflow of the parent project is ignored and the branch workflow is used.

To remove the existing workflow without replacing it with another workflow, select **No Workflow** from the **Workflow** dropdown list. This is the default when you create a new project.

For more information about workflows and how project workflows interact with branch workflows, see "[Workflow basics](#)" on page 512.

4. Set the **Minimum up votes** for a project. The project setting is used on the project branches unless **Minimum up votes** is set to **1** or higher on a branch. In this case the branch setting is used and the project setting is ignored.

Enter the **Minimum up votes** or use the up and down arrow keys to increase or decrease the number of **Minimum up votes**. To save your changes click the green tick icon. Click the cross icon to delete your changes.

Use **Minimum up votes** to set the minimum number of up votes required for reviews in this branch.

A review cannot be approved until all of the [Required reviewers](#) have voted up the review and the **Minimum up votes** specified has been satisfied.

When this setting is defined at the project level, minimum number of up votes is automatically applied to all branches. However, if you set **Minimum up votes** to **1** or higher on a specific branch, that branch will use its own setting and ignore the project-level setting.

To use the project-level value, set **Minimum up votes** to **Inherit from project**. This is set by default for new branches.

To override the project setting and use a custom value for a certain branch, set **Minimum up votes** to **1** or higher on that branch.

#### Notes on minimum up votes

- If a review includes changes across multiple projects or branches, the minimum number of up votes on each project or branch must be met before the review can be approved.
- Required reviewers are included when counting up votes.
- When the workflow rule **Count votes up from** is set to **Members** for a project or branch, only up votes from members belong to that project or branch will count toward the minimum up votes requirement. For more information on the **Count votes up from** rule, see ["Workflow rules" on page 510](#).

If workflows are disabled, all votes are counted, not just votes from project members.

**Restriction:** If the **Minimum up votes** requirement is higher than the total number of reviewers on a review, it will be impossible to approve it, even if every reviewer votes to approve. Take this into consideration when setting **Minimum up votes**.

#### 5. Specify a **Job filter**.

The job filter allows you to specify criteria that are used to associate jobs with projects. For example, entering Subsystem=ProjectA associates jobs whose subsystem field is set to ProjectA with the current project.

##### Note:

This job filter is simpler than the filters available in other P4 Server clients. The filter must be expressed as **field=value** pairs; bare keywords are not permitted. The asterisk for wildcard matching is permitted, but no other filter expression syntax is permitted.

#### 6. Slide the **Private project** toggle to make this project private. Private projects and their associated reviews are only visible to project owners, moderators, and members, plus users with super privileges in P4 Server.

For more information, see ["Private projects" on page 366](#).

#### 7. Delete a project as follows:

- a. Click **Delete Project**.

A confirmation dialog appears to confirm whether you want to delete this project.

- b. Click **Confirm** to delete the project.

After you have deleted a project, P4 Code Review will attempt to cleanup the deleted project. For more information, see [Delete a project](#).

### Important:

It is possible to create a project that you cannot edit. This can happen if you have specified owners but not yourself as an owner, or if you have not specified yourself as a member. P4 Code Review can detect some (but not all) such situations when you save a project; when it does detect such a situation, a warning dialog is displayed.

If you see this dialog, click **Continue** to save the project without your ownership/membership, or click **Cancel** within the dialog to continue editing the project. The project page's **Save** and **Cancel** buttons are disabled while this dialog is visible.

## Edit a project

1. Visit the project page you want to edit.
2. Click **Settings** in the project menu.

A project summary with Participants tab, Branches tab, Integrations tab, and General Settings tab is displayed.

3. Edit the project's details as required.

For more information on how to use the Settings page, see ["Project settings" on page 478](#).

### Tip:

If sharing is switched off for a workflow, any projects or branches that were associated with the workflow while it was shared will remain associated with it.

If you edit a project or branch associated with that unshared workflow, you will still see the name of the workflow in the **Workflow** field, even if you don't own that workflow. If you remove that workflow from the project/branch you will not be able to see the workflow in the **Workflow** dropdown list unless you own it.

### Note:

By default, any member of a project can edit the project's configuration. Administrators can configure P4 Code Review to prevent changes to the project's name and branch definition(s). See ["Projects" on page 659](#) for details.

## Membership

Membership in a P4 Code Review project identifies users as belonging to the project, making them part of the team.

There are only a few notable differences between project members and non-members:

| Difference                                  | Member | Non-Member | Description                                                                             |
|---------------------------------------------|--------|------------|-----------------------------------------------------------------------------------------|
| <a href="#">"Notifications" on page 376</a> | ✓      | ✗          | Members receive project notifications; non-members do not.                              |
| <a href="#">"Avatars" on page 390</a>       | ✓      | ✗          | The project's home page features member's avatars.                                      |
| <a href="#">"States" on page 464</a>        | ✓      | ✗          | Members, moderators, and authors can transition code review states; non-members cannot. |

There are two ways to become a member of a project in P4 Code Review:

1. Add a project and make yourself a member.
2. Ask a member of an existing project to add you as a member.

**Note:**

If the project has any [owners](#) specified, you need to ask a project owner to add you as a member.

**Note:**

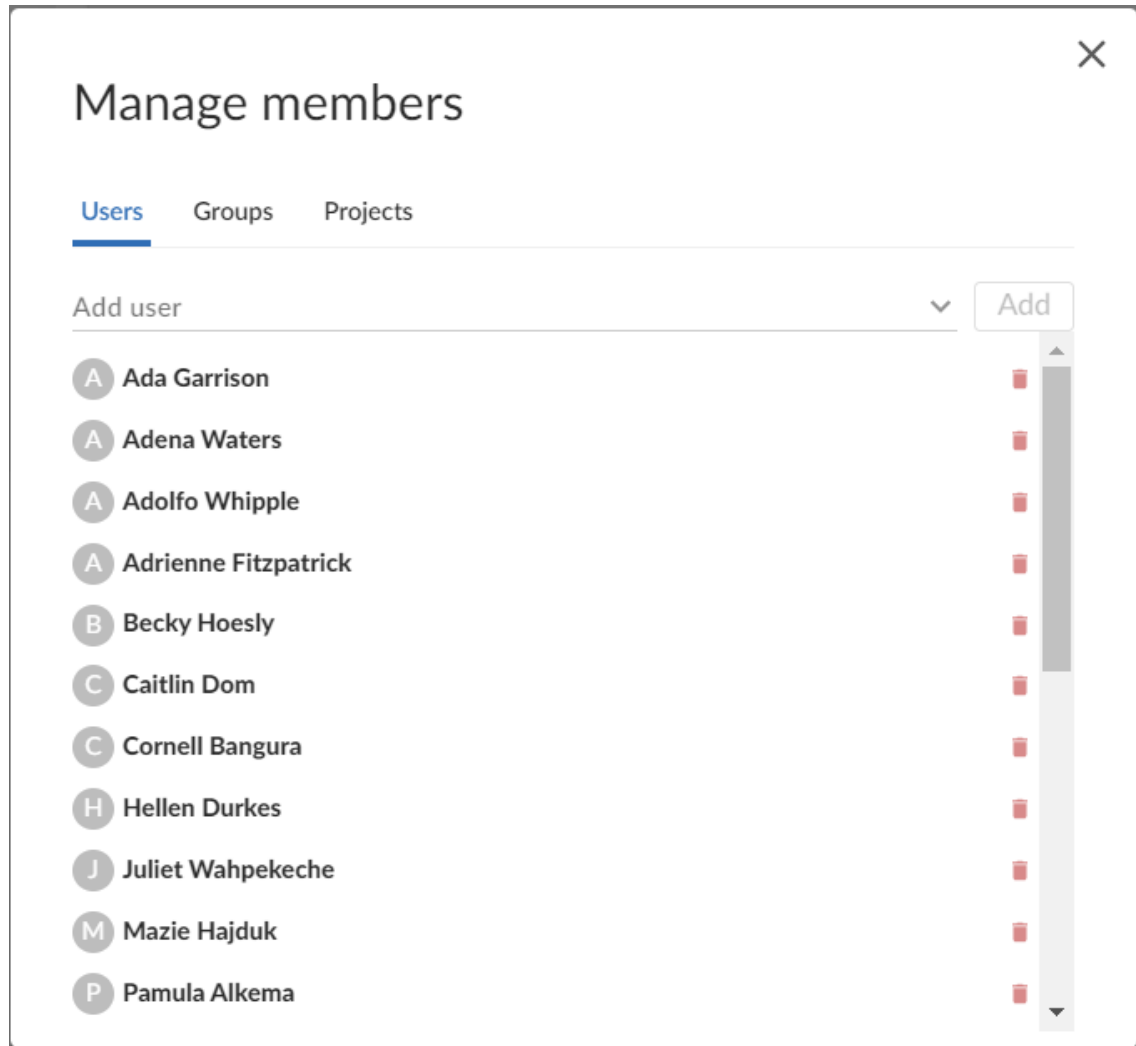
Users with *super* privileges in P4 Server can always adjust the settings for any project, including adjusting membership.

## Add a member

If you are an owner of a project, or a member of a project without specified owners:

1. Visit the project page that needs the new member.
2. Click **Settings** in the project's toolbar.
3. The **Members** pane displays a list of all the project members.
4. Click **Manage members** button and specify a P4 Server user, P4 Server group, or P4 Code Review project to add to the members for this project. The field auto-suggests *projectids*, *groupids*, and *userid*s by matching what you have typed so far against the list of users in the P4 Server.

When you specify a project or group, all of the members of that project or group become members of this project. P4 Code Review does not display all of the individual users, but it does provide a visual separation: project or group names are displayed first, with a darker blue background.



5. Click **Add**.

## Remove a member

If you are an owner of a project, or a member of a project without specified owners:

1. Visit the project page that has a member you want to remove.
2. Click **Settings** in the project's toolbar.

Known members of the project are displayed beneath the **Members** text field, with a medium blue button representing projects or groups and a light blue button representing individual users.

3. Click the **X** next to the project id, group id, or userid you want to remove.
4. Click **Save**.

**Warning:**

You are able to remove your own membership or ownership. Doing so could prevent you from managing the project.

## Owners

A project *owner* is a P4 Server user that controls the configuration for a project. An owner does not need to be a member of a project, but only an owner or user with *super* privileges in P4 Server can edit any project settings.

**Tip:**

The project **Automated Tests** and **Automated Deployment** details are hidden from project members unless they are an owner or an administrator. This enables project members to check the project settings but not change them.

## Moderators

A project *moderator* is a user assigned to moderate reviews for a specific branch associated with a project. See how to [specify moderators](#).

Once the branch specification is complete and the project has been saved, changing the state of any review associated with the moderated branch is restricted as follows:

Only moderators can approve or reject reviews on the branch.

To add moderators to a branch:

1. Click the **Manage moderators** button.
2. Choose the **Users** or **Groupstab**.
  - a. If you selected **Users**, start typing a the name of a user. Suggested users on the P4 Server will appear automatically.
  - b. If you selected **groups**, then all of the members of that group will have the same moderator privileges for that project branch. Use this option if you wanted to add several users at once.
3. Click Add.

After you finish setting up the branch and save the project, reviews on this branch can only be approved or rejected by moderators.

Moderators can also transition a review to any other state. Below is a summary of branch moderator roles and review state rules.



| Role                      | Allowed actions                                                                                                                                                                                                  | Restrictions                                                                                                                                             |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------|
| Moderators                | <p>Can move a review to any state.</p> <p>Can approve or reject reviews.</p> <p>Can prevent automatic approvals.</p>                                                                                             | <p>None (unless <code>disable_self_approve</code> is enabled for self-approval). For more information, see <a href="#">disable_self_approve</a>.</p>     |
| Authors (not moderators)  | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Needs review</li> <li>Needs revision</li> <li>Archived</li> </ul> <p>Can attach committed changelists.</p>                                    | <p>Cannot approve or reject reviews.</p>                                                                                                                 |
| Authors (also moderators) | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Approve</li> <li>Rejected</li> <li>Needs review</li> <li>Needs revision</li> <li>Archived</li> </ul> <p>Can attach committed changelists.</p> | <p>Cannot approve their own reviews if <code>disable_self_approve</code> is enabled. For more information, see <a href="#">disable_self_approve</a>.</p> |
| Project members           | <p>Can change state to:</p> <ul style="list-style-type: none"> <li>Needs review</li> <li>Needs revision</li> </ul> <p>Can attach committed changelists.</p>                                                      | <p>Cannot approve, reject, or archive reviews.</p>                                                                                                       |



| Role        | Allowed actions     | Restrictions                                                                                       |
|-------------|---------------------|----------------------------------------------------------------------------------------------------|
| Other users | None                | Cannot change review states.                                                                       |
| All roles   | Can start a review. | Cannot change state if it's not in their allowed states (for example, can't change from Rejected). |

#### Additional notes

- Moderators can prevent automatic approvals. For more information about automatically approving reviews using workflow rules see ["Workflow rules" on page 510](#).
- By default, if a review spans multiple branches with different moderators, only one moderator from any branch needs to approve it. You can change this (via a global setting) to require one moderator per branch. If a moderator belongs to multiple branches, one approval counts for each. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

To delete a moderator, click the delete icon against their name.

Branch moderators(1), only moderators can approve or reject reviews
Manage moderators

| Name                                                                                                  | Type |                                                                                       |
|-------------------------------------------------------------------------------------------------------|------|---------------------------------------------------------------------------------------|
|  Allison Clayborne | User |  |

Rows per page: 25
1-1 of 1

## Default reviewers

User and group default reviewers can be set for individual projects and project branches. Each time a new review is created in the project or project branch, the default reviewers will be added to the review. See [projects](#) and [project branches](#) for adding default reviewers.

- A user can be set as a [required reviewer](#) or an optional reviewer.
- A group can be set as a [required reviewer \(one vote\)](#), a [required reviewer \(all votes\)](#), or an optional reviewer.

#### Default reviewers for reviews in multiple projects or branches:

When a review belongs to more than one project or branch:

- The default reviewers from all the related projects and branches are combined and added to the review.
- If a reviewer has different vote requirements in different places, the strictest setting is used.

For example:

- In Project-A, Reviewer X is optional.
- In Project-B, Reviewer X is optional.
- In the project branch, Branch-b, Reviewer X is required.

As a result, Reviewer X will be added as a required reviewer on the review.

### Mentioning default reviewers

If users or groups are [@mentioned](#) in a new changelist description that includes [#review](#), they will be added to the review as reviewers. If any of these reviewers are already specified as default reviewers they will not be added to the review again, the reviewer's most restrictive reviewer option is used for the review.

## Retain default reviewers

By default, default reviewers can be removed from an individual review by using the [edit reviewers](#) button on the review display page. Individual projects and branches can be configured to prevent default reviewers from being removed from individual reviews. For instructions on how to enable **Retain default reviewers** for a project or branch, see [project](#) or [project branch](#).

### Retain default reviewers basics

- Retained default reviewers can be added or removed by editing the project or project branch. Each time a review is updated in the project or branch, the list of default reviewers is checked.
  - Any new default reviewers are added to the review.
  - If a default reviewer has been removed from the project/branch, they will remain on the review. They are no longer a retained default reviewer and can be removed from the review if required.
  - If a review is updated and it is no longer associated with the project/branch, the default reviewers for that project/branch will remain on the review. They are no longer retained default reviewers and can be removed from the review if required.

#### A review is checked when:

- The [review state](#) changes
- A new version of the review is created
- A [comment task state](#) is changed
- A user votes on the review, or clears their vote
- [Reviewers are edited](#) on the review
- The [review author is changed](#)

- Retained default reviewers cannot be removed from a review by [editing reviewers](#) in the review.
- The retained default reviewer voting option cannot be reduced in a review by [editing reviewers](#) in the review.

**Example:** if a user is a [Required reviewer](#), the voting option cannot be reduced to a less strict option such as Optional reviewer.

- The retained default reviewer voting option can be made stricter for a review by [editing reviewers](#) in the review.

**Example:** if a group is an Optional reviewer, the voting option can be increased to a stricter option such as Required reviewer (one vote).

## Reviews spanning multiple projects and branches

### Related projects and branches

When a review spans related projects and branches and a user/group is a retained default reviewer in one but a standard default reviewer in the other, the branch voting option is used for the user/group reviewer in the review.

#### Example:

- **Review 5678:** spans **Project A**, and **Branch A-1**.
- **Project A:**
  - **User-X:** Required reviewer
  - **Retain default reviewers:** Enabled.
    - **Branch A-1:** a child of **Project A**:
      - **User-X:** Optional reviewer
      - **Retain default reviewers:** Disabled.
- **Review:**
  - **User-X:** Optional reviewer, not a retained default reviewer.

The project voting option is ignored and the branch voting option is used.

There is no restriction on changing the default reviewer voting option.

The default reviewer can be removed from the review.

### Unrelated projects and branches

When an individual user/group default reviewer is retained for some of the projects and branches the review spans but not for others, the strictest voting option is used for a default reviewer on the review. The strictest retained default reviewer voting option is the minimum voting option for the user/group default reviewer in the review.

#### Example:

- **Review 1234:** spans **Project A**, **Project C**, and **Branch F-1**.
- **Project A:**
  - **Group-D:** Optional reviewer.
  - **Retain default reviewers:** Enabled.
- **Project C:**
  - **Group-D:** Required reviewer (all votes).
  - **Retain default reviewers:** Disabled.
- **Branch F-1:** not related to **Project A** or **Project C**:
  - **Group-D:** Required reviewer (one Vote)
  - **Retain default reviewer:** Enabled.
- **Review:**
  - **Group-D:** Required reviewer (All votes), retained default reviewer.

The strictest voting option is used for the default reviewer.

You can edit the voting option on the review. The minimum voting option is the strictest of the retained reviewer voting options. In this case that is, Required reviewer (one vote).

The reviewer cannot be removed from the review.

## Delete a project

### Note:

- Users with *super* or *admin* privileges in P4 Server can delete projects.
  - If the P4 Code Review user has insufficient permissions, cleanup on the deleted P4 projects will fail.
- Project owners can delete projects that they own.
- If a project has no owners, any member of the project can delete the project.

When you delete a project, P4 Code Review will try to clean it up by doing the following:

- The deleted project is removed from the P4 Code Review project list on the dashboard, and from project searches.
- The deleted project is removed from the profile page of the project owners, members, moderators, and followers.
- The project name is removed from the **Project** column on the ["Reviews list" on page 406](#) page.
- Reviews that belong to the deleted project are not changed. The open and closed reviews remain accessible, their [review states](#), [comments](#), and [tasks](#) can be modified as normal.
- Reviews that belong to the deleted project, the project branch name link in the [review heading](#) is replaced with a link to the common depot location that contains the files included in the review.

**Note:**

The deleted project name cannot be reused for a new project. This behavior is not case sensitive, this means that if you delete a project called **Project B** you cannot create a new project called **project b**.

Use the following steps to delete a project:

1. Visit the project page you want to remove.
2. Click **Settings** in the project menu and navigate to the **General Settings** tab.
3. Click **Delete**.

A confirmation dialog appears to confirm whether you want to delete the project.

4. Click **Confirm** to delete.

## When an owner is deleted outside of P4 Code Review

When a project owner is deleted from P4 Server, the ownership of the project may change depending on the users and owners associated with the project.

- If a project has more than one owner and one of these owners is deleted from P4 Server, the project remains as it is and the deleted user is removed from the project with ownership kept among the remaining owners.
- If a project has one owner and at least one project member, when the owner is deleted from P4 Server, the project remains as it is but project ownership is transferred to P4 Code Review.
- If a project has one owner and no members, when the owner is deleted from P4 Server, the project is deleted by P4 Code Review.

## 8 | Workflows

This chapter covers P4 Code Review workflow management, including:

- ["Why should I use P4 Code Review workflow?" below](#)
- ["Workflow overview" on page 509](#)
- ["Add a workflow" on page 521](#)
- ["Edit a workflow" on page 526](#)
- ["Delete a workflow" on page 526](#)

### Related information:

- For instructions on how to list, search, and view workflows, see ["Workflows" on page 367](#).
- For instructions on how to add a workflow to a project, see [Add a workflow to a project](#).
- For instructions on how to add a workflow to a branch, see [Add a workflow to a project branch](#).
- For a step-by-step walk-through of setting up a workflow, adding it to a project, and using it to progress through a review, see the [P4 Code Review Documentation](#).
- **Administrators:** For instructions on how to configure the global workflow rules, see [Workflow global rules](#).

#### Tip:

For P4 Code Review 2019.2 and later, the workflow feature is enabled by default. If the workflow feature is not working, check that the workflow prerequisites have been met, see ["Workflow prerequisites" on page 102](#).

## Why should I use P4 Code Review workflow?

By default P4 Code Review's review process is basically advisory in nature with very few restrictions imposed on the review process. Team members can work in whatever way they want, but as projects grow in size and complexity, teams may prefer a stricter set of workflow rules over flexibility in order to improve productivity and quality. However, relying on people to stick to the agreed rules is not always enough.

The quality and speed of delivery of a product is heavily dependent on having a standardized and controlled workflow that follows best practice. It is also beneficial to automate the workflow where possible because the benefits of automating the workflow increases exponentially with scale. Teams within your organization will have varied workflow needs, it is important to maintain a balance between a company-wide policy and project specific needs. The P4 Code Review workflow features have been designed to help you achieve this balance.

#### Tip:

Additional Workflow rules and enhancements are on the road map for future P4 Code Review releases.

The P4 Code Review workflow feature enables you to enforce your rules and enables you to define specific rules for different projects and branches, providing you with flexibility and control over your review workflow. Workflows can be shared and they can be used on multiple projects/branches. Changing the workflow rules updates the workflow for all of the projects and branches it is associated with. This makes it easy to make changes in one place and change the workflow of multiple projects/branches. Specific workflow rules can be applied globally, ensuring basic company policies are followed by every project.

As a project owner you want to setup a project using the [Mainline model](#) with 'development', 'main' and 'release' branches. Each of these branches will have workflow rules that are tailored to its branch type.

Typically, you want your development branch to allow quick changes to the project content so that it can be populated quickly. Breaking test is acceptable at this point in the project. It is usually acceptable for a review to be raised after the content has been committed, this is the post-review commit model. Often you would only require one reviewer to vote up a review to trigger automatic approval. This minimal workflow configuration helps the development line to be built up quickly.

**Tip:**

If your developers are pair programming, you might even allow them to commit changes without any review to further speed up the process. In the example shown below this is not the case and reviews are enforced by the workflow rules.

For your main branch you probably want to enforce pre-commit reviews, require more reviewers to vote up a review to trigger automatic approval, and restrict the up votes to project members only. This all helps to increase the stability of the main branch.

For a release branch you want even more protection for the content of the branch by only allowing critical changes to be committed to the branch. This is controlled by adding senior project members as moderators to the branch. Moderators are the only users that can approve or reject a review so they act as the gatekeepers for the branch. Moderators are added to the branch from the [project branch settings](#) pane.

**Development branch example:**

A Development branch needs to be fast moving, allowing new features, collateral, and technologies to be added very quickly with issues fixed as they occur. In essence this is rapid development allowing a lot of work to be completed quickly but the branch is unstable. The workflow for your development branch would be configured to allow for this style of working, typically:

- The branch rules allow post-commit and pre-commit reviews as required:
  - Changes can be committed and a review can be requested later, this is a post-commit review. This approach is particularly useful if you are making large changes and you just want to get content into the depot and then check it.
  - Changes can be shelved with a review, this is a pre-commit review. This is useful when a user is adding standalone content or a feature that needs to be approved before it is committed to the depot.
- Only one reviewer needs to vote up a review to trigger automatic approval.

**Main branch example:**



A main branch needs to be more tightly controlled than the development branch as it is more stable as you move towards release. In essence, the main branch is used to fix any lingering issues and firm up the content and any tests you have in place. The workflow for your main branch would be configured to allow for this style of working, typically:

- Changelists must be shelved with a review, this is a pre-commit review.
- Up votes for reviews are only counted from project members.
- Two reviewers from the project are required to vote up a review to trigger automatic approval.
- The changelist can only be committed if the content of the changelist is identical to the content of the approved review.

#### Release branch example:

A Release branch typically needs to be very tightly controlled to protect the codeline with only a select set of users empowered to approve and commit reviews to the project before the product is released. In essence, the branch should be stable and you should not be making changes unless absolutely necessary. The workflow for your release branch would be configured to allow for this style of working, typically:

- Changelists must be shelved with a review, this is a pre-commit review.
- Senior members of the project are required to vote up the review before it can be approved.
- Moderators act as the final gatekeepers for review approval, only they have the power to approve a review once the required reviewers have voted the review up.
- The changelist can only be committed if the content of the changelist is identical to the content of the approved review.
- Approval is blocked if specific tests fail.

These are three very different workflows, it is easy for team members to forget the rules especially when they are new to the project or team. P4 Code Review workflows enable you to enforce your rules and control your branches more effectively. No more last minute changes breaking your release build and no more post-commit reviews in your pre-commit world.

## Workflow overview

### Note:

- For P4 Code Review 2019.2 and later, the workflow feature is enabled by default. If the workflow feature is not working, ask your P4 Code Review administrator to check that the workflow prerequisites have been met, see ["Workflow prerequisites" on page 102](#).
- The P4 Code Review administrator can give individual groups and users special permission to ignore all of the P4 Code Review workflow rules. Typically, these permissions are given to a small set of trusted groups and users, for example project owners and administrators. For more information about excluding users and groups from being bound by the workflow rules, see ["Members or groups who can ignore ALL workflow rules" on page 748](#).

**Tip:**

To understand the benefits of using the P4 Code Review Workflow feature, see ["Why should I use P4 Code Review workflow?" on page 507](#)

A workflow can be applied to a project or project branch to ensure that changelists and reviews in the project/branch follow the rules specified in that workflow. The ["Workflow basics" on page 512](#) section gives an overview of global workflow rules, workflow rules and how they interact with each other.

**Sections in this chapter:**

- ["Workflow rules" below.](#)
- ["Workflow basics" on page 512.](#)
- ["Example workflows" on page 513.](#)
- ["Merging multiple workflows" on page 515.](#)

## Workflow rules

A workflow is made up of the following workflow rules:

**On commit without a review:**

This rule is applied when a changelist without an associated review is submitted from outside of P4 Code Review.

Select one of the following options:

- **Allow:** the changelist is not checked, the changelist is submitted without a review.
- **Create a review:** the changelist is submitted and a review is created automatically for the changelist.
- **Reject:** the changelist submit is rejected.

**Tip:**

The selected rule is also applied when a changelist is submitted with `#review` in the description. For more information about creating a review by including a keyword in the changelist description, see ["Create a review" on page 669](#).

**On commit with a review:** This rule is applied when:

- A review is committed.
- A changelist with an associated review is submitted from outside of P4 Code Review.

Select one of the following options:

- **Allow:** the changelist review state is not checked. The changelist is committed even if its associated review is not approved.
- **Reject unless approved:** the changelist is only submitted if its associated review is approved and the content of the changelist is identical to the content of the approved review.

**Tip:**

The selected rule is also applied when a changelist is submitted with **#review-nnnnn**, **#replace-nnnnn**, or **#append-nnnnn** in the description (nnnnn = review ID). For more information about adding a changelist to a review by including a keyword in the changelist description, see ["Add a changelist to a review" on page 670](#).

**On update of a review in an end state:**

Used to stop review content being automatically changed for reviews that are in specific states. By default, the protected end states are **Archived**, **Rejected**, and **Approve:Commit**. The end states are set by the P4 Code Review administrator, see [end\\_states](#).

**Tip:**

When a review is in a protected end state, it can still be updated manually by a user from the P4 Code Review UI.

This rule is applied when a changelist is added to a review.

Select one of the following options

- **Allow:** the review is not checked. The changelist is added to the review no matter what the review state is. This is the default setting.
- **Reject:** if the review is in one of the states specified in the `end_state` configurable:
  - The **Add Change** button is disabled for the review.
  - The changelist is rejected if it is added outside of P4 Code Review using **#review-nnnnn**, **#replace-nnnnn**, or **#append-nnnnn** in the description (nnnnn = review ID).

**Count votes up from:**

By default, all of the up votes on a review are counted for the **Minimum up votes** value set on the project/branch the review is associated with. Limit the up votes that are counted to just the members of the project the review is associated with by using this rule.

This rule is applied when a user votes on a review.

Select one of the following options:

- **Anyone:** votes are counted for all reviewers on a review. This is the default setting.
- **Members:** only the up votes of members of the project the review is associated with are counted for the **Minimum up votes** set on projects/branches.

**Tip:**

For instructions on how to set **Minimum up votes** for projects and branches, see [Project minimum up votes](#) and [Branch minimum up votes](#).

**Automatically approve reviews:**

By default, reviews must be manually approved. Enable automatic approval of reviews with this rule.

This rule is applied when a user votes on a review.

Select one of the following options:

- **Never:** reviews are not automatically approved. This is the default setting.
- **Based on vote count:** reviews are automatically approved if:
  - There are no down votes on the review.
  - There are no moderators on the review. If a review has moderators it cannot be automatically approved.
  - All of the [Required reviewers](#) on the review have voted up.
  - The **Minimum up votes** on the review has been satisfied for **each** of the projects and branches the review spans.
  - All of the tests configured to block approval have passed.

**Important:**

Moderators prevent the automatic approval of reviews. For more information about moderators, see ["Moderators" on page 455](#).

**Tip:**

After a review has been automatically approved it needs to be manually committed.

For information about creating a new workflow, see ["Add a workflow" on page 521](#).

## Workflow basics

- **Workflows:** can be associated with projects and/or project branches.
  - A workflow can be created by any P4 Code Review user.
  - If a workflow is associated with any projects or project branches, that workflow cannot be deleted.
  - A workflow can be shared by the workflow owner.
  - A shared workflow can be viewed by any P4 Code Review user.
  - A shared workflow can be applied to a project or project branch by any P4 Code Review user that is authorized to edit the project.
  - If sharing is switched off for a workflow, any projects or branches associated with that workflow will remain associated with it.
- **Projects and branches:**
  - When a workflow is associated with a project, the workflow is used for all of the branches in that project.
  - When a project branch is associated with a workflow, the workflow of the parent project is ignored and the branch workflow is used. The project workflow is replaced by the branch workflow, the workflows are not merged. For examples of why you might use this, see ["Restrictive project workflow with a less restrictive development branch workflow" on page 514](#), and ["Project workflow with a more restrictive release branch workflow" on page 515](#).

- When a branch is associated with a workflow and a global workflow rule is set to **Enforce on** by the administrator, the workflow of the parent project is ignored and the global workflow rule is merged with the branch workflow. The most restrictive rule is used.
- **Modifying a workflow:**
  - If a global workflow rule is set to **Enforce off** and that rule is modified, projects and branches that do not have an associated workflow will use the modified rule.
  - If a global workflow rule is set to **Enforce on** and the rule is modified, the entire P4 Server will use the modified rule.
  - If a workflow is modified, any projects or branches associated with that workflow will use the modified workflow.
- **Global workflow rules:** configured by the P4 Code Review administrator in [Global workflow rules](#), each rule can be set in one of two modes:
  - **Enforce off:** the global workflow rule is used for projects and branches that do not have an associated workflow.
    - If a project or branch has an associated workflow, the global rule is ignored for that project or branch.
    - If projects are not used, the global workflow rule applies to the entire P4 Server.
  - **Enforce on:** the global workflow rule is used for the entire P4 Server. This allows a minimum workflow to be set for the P4 Server.
    - The global workflow rule is merged with the workflow associated with a project or branch. The most restrictive rule is used.
- **Tests:**
  - If a test is associated with a workflow, it can be configured to run when a review associated with that workflow is either created/updated or submitted, or run manually from the review.
  - If a test is associated with the global workflow, it can be configured to run when a review is either created/updated or submitted, or run manually from the review.
  - If a test is associated with a workflow or a global workflow, it can be configured to block approval if it fails.

## Example workflows

This section contains examples of typical workflows that can be configured in P4 Code Review.

### Enforce a pre-commit review workflow

To enforce a [pre-commit](#) review workflow set the rules to:

- **On commit without a review:** set to **Reject**, ensures that changelist cannot be submitted without an approved review.
- **On commit with a review:** set to **Reject unless approved**, ensures that if a changelist already has an associated review, that review is approved and the content of the changelist is identical to the content of the approved review.

## Allow a post-commit review workflow

To allow a [post-commit](#) review workflow set the rules to:

- **On commit without a review:** set to **Create a review**, ensures that all submitted changelists have a review.
- **On commit with a review:** set to **Allow**, enables changelists that already have an associated review to be submitted.

## Restrictive project workflow with a less restrictive development branch workflow

The release and main branches need a restrictive workflow to protect the codelines. All submissions must have an associated approved review and the content of the changelist must be identical to the content of the approved review.

The development branch can be less restrictive because you want to get features in quickly and review them after they have been committed. In this scenario you would probably want to ensure that all submitted changelists have an associated review.

### Project workflow:

- **On commit without a review:** set to **Reject**, ensures that changelist cannot be submitted without an approved review.
- **On commit with a review:** set to **Reject unless approved**, ensures that if a changelist already has an associated review, that review is approved and the content of the changelist is identical to the content of the approved review.
- **On update of a review in an end state:** set to **Reject**, ensures that the content of reviews that are considered complete cannot be changed. These end states are set by the P4 Code Review administrator, see ["Protected end states" on page 677](#).
- **Count votes up from:** set to **Members**, ensures that only the up votes of members of the project the review is associated with are counted for the **Minimum up votes** set on projects/branches.
- **Automatically approve reviews:** set to **Never**, ensures that reviews are not automatically approved.

### Development branch workflow:

- **On commit without a review:** set to **Create a review**, ensures that all submitted changelists have a review.
- **On commit with a review:** set to **Allow**, enables changelists that already have an associated review to be submitted.
- **On update of a review in an end state:** set to **Reject**, ensures that the content of reviews that are considered complete cannot be changed. These end states are set by the P4 Code Review administrator, see ["Protected end states" on page 677](#).
- **Count votes up from:** set to **Anyone**, ensures that votes of all reviewers on a review are counted for the **Minimum up votes** set on projects/branches.
- **Automatically approve reviews:** set to **Based on vote count**, ensures that reviews are automatically approved when all of the review requirements have been satisfied.

## Project workflow with a more restrictive release branch workflow

The project is set up with workflow that allows post commit reviews, this allows you to get features in quickly and review them after they have been committed. In this scenario it is a good idea make all submitted changelists have an associated review.

The release branch needs a more restrictive workflow to protect the codeline. All submissions must have an associated approved review and the content of the changelist must be identical to the content of the approved review.

**Project workflow:** defines the workflow for the project and its branches.

- **On commit without a review:** set to **Create a review**, ensures that all submitted changelists have a review.
- **On commit with a review:** set to **Allow**, enables changelists that already have an associated review to be submitted.
- **On update of a review in an end state:** set to **Allow**, allows the content of reviews that are considered complete to be changed.
- **Count votes up from:** set to **Anyone**, ensures that votes of all reviewers on a review are counted for the **Minimum up votes** set on projects/branches.
- **Automatically approve reviews:** set to **Based on vote count**, ensures that reviews are automatically approved when all of the review requirements have been satisfied.

**Release branch workflow:** defines the workflow for the release branch. The workflow of the parent project is ignored and the development branch workflow is used.

- **On commit without a review:** set to **Reject**, ensures that changelist cannot be submitted without an approved review.
- **On commit with a review:** set to **Reject unless approved**, ensures that if a changelist already has an associated review, that review is approved and the content of the changelist is identical to the content of the approved review.
- **On update of a review in an end state:** set to **Reject**, ensures that the content of reviews that are considered complete cannot be changed. These end states are set by the P4 Code Review administrator, see "[Protected end states](#)" on page 677.
- **Count votes up from:** set to **Members**, ensures that only the up votes of members of the project the review is associated with are counted for the **Minimum up votes** set on projects/branches.
- **Automatically approve reviews:** set to **Never**, ensures that reviews are not automatically approved.

## Merging multiple workflows

If a changelist spans multiple projects that have different workflows, the workflows are merged and the most restrictive workflow rules are used for the changelist. All of the tests associated with the merged workflows are run. If there are any duplicate tests they are only run once.

## Example 1: Changelist 1234 spans Project A, Project B, and Project C

- **Project A:** //depot/jam/lib/...
- **Project B:** //depot/jam/docs/...
- **Project C:** //depot/common/src/...
- **Changelist 1234** contains the following files:
  - //depot/jam/lib/tests/checkLoad.sh
  - //depot/jam/docs/basics.workflow.html
  - //depot/common/src/index.html

The files span all three locations so the changelist is subject to the combined workflow rules for the three locations. The most restrictive workflow is used for the changelist:

|                                      | Chang<br>elist<br>without<br>a<br>review | Chang<br>elist<br>with a<br>review | On<br>upd<br>ate<br>of a<br>revie<br>w in<br>an<br>end<br>state | Coun<br>t<br>votes<br>up<br>from | Automati<br>cally<br>approve<br>reviews | Workfl<br>ow<br>Tests              |
|--------------------------------------|------------------------------------------|------------------------------------|-----------------------------------------------------------------|----------------------------------|-----------------------------------------|------------------------------------|
| <b>Project<br/>A</b><br>workflo<br>w | Allow                                    | Reject<br>unless<br>approve<br>d   | Allow                                                           | Anyon<br>e                       | Based on<br>vote count                  | Smoke<br>Test A                    |
| <b>Project<br/>B</b><br>workflo<br>w | Allow                                    | Reject<br>unless<br>approve<br>d   | Rejec<br>t                                                      | Memb<br>ers                      | Never                                   | Smoke<br>Test B                    |
| <b>Project<br/>C</b><br>workflo<br>w | Create a<br>review                       | Allow                              | Allow                                                           | Anyon<br>e                       | Based on<br>vote count                  | Smoke<br>Test B<br>Smoke<br>Test C |



|                          | Changelist without a review | Changelist with a review | On update of a review in an end state | Count votes up from | Automatically approve reviews | Workflow Tests                               |
|--------------------------|-----------------------------|--------------------------|---------------------------------------|---------------------|-------------------------------|----------------------------------------------|
| Changelist 1234 workflow | Create a review             | Reject unless approved   | Reject                                | Members             | Never                         | Smoke Test A<br>Smoke Test B<br>Smoke Test C |

**Tip:**

The three workflows are combined and the most restrictive workflow rules are used for the changelist. All of the tests associated with the three workflows are run but **Smoke Test B** on **Project A** and **Project B** is only run once.

## Example 2: Changelist 6789 spans Project M, Project N, and Project branch N-1

- **Project M:** //depot/thirdparty/lib/...
- **Project N:** //depot/thirdparty/lib/tests/...
- **Project branch N-1:**
  - //depot/projectN/...
  - //depot/thirdparty/lib/...
- **Changelist 6789** contains the following files:
  - //depot/thirdparty/lib/tests/checkReturns.sh
  - //depot/projectN/src/index.html
  - //depot/projectN/tests/pageLoadTest.php

The changelist files span all three locations so the changelist is subject to the combined workflow rules for the three locations. The most restrictive workflow is used for the changelist:

|                                    | Changelist without a review | Changelist with a review | On update of a review in an end state | Count votes up from | Automatically approve reviews | Workflow Tests                 |
|------------------------------------|-----------------------------|--------------------------|---------------------------------------|---------------------|-------------------------------|--------------------------------|
| <b>Project M</b> workflow          | Create a review             | Reject unless approved   | Allow                                 | Anyone              | Never                         | Smoke Test M                   |
| <b>Project N</b> workflow          | Reject                      | Reject unless approved   | Reject                                | Members             | Never                         | Smoke Test N                   |
| <b>Project branch N-1</b> workflow | Create a review             | Allow                    | Allow                                 | Anyone              | Based on vote count           | Smoke Test N-1                 |
| <b>Changelist 6789</b> workflow    | Create a review             | Reject unless approved   | Allow                                 | Anyone              | Never                         | Smoke Test M<br>Smoke Test N-1 |

**Tip:**

It is important to understand that the rules for **Project N** are ignored because **Project branch N-1** is a branch of **Project N** and it has its own workflow rules. This is why the result for **Changelist without a review** is **Create review**, the result for **On update of a review in an end state** is **Allow**, and the result for **Count votes up from** is **Anyone**.

### Example 3: Changelist 3456 spans Project X, and Project Y

- **Project X:** //depot/main/product/...
- **Project Y:** //depot/main/product/docs/...

- **Changelist 3456** contains the following files:
  - `//depot/main/product/tests/fileSelector.sh`
  - `//depot/main/product/docs/admin.html`

The changelist files span both locations so the changelist is subject to the combined workflow rules for the two locations, and the [Global workflow rules](#) set by the administrator. The most restrictive workflow is used for the changelist:

|                                           | Chang<br>elist<br>without<br>a<br>review         | Chang<br>elist<br>with a<br>review                              | On<br>upda<br>te of<br>a<br>revie<br>w in<br>an<br>end<br>state | Coun<br>t<br>votes<br>up<br>from               | Automati<br>cally<br>approve<br>reviews | Workfl<br>ow<br>Tests                              |
|-------------------------------------------|--------------------------------------------------|-----------------------------------------------------------------|-----------------------------------------------------------------|------------------------------------------------|-----------------------------------------|----------------------------------------------------|
| <b>Project<br/>X</b><br>workflo<br>w      | Create a<br>review                               | Allow                                                           | Allow                                                           | Anyon<br>e                                     | Based on<br>vote count                  | Smoke<br>Test X                                    |
| <b>Project<br/>Y</b><br>workflo<br>w      | Reject                                           | Allow                                                           | Rejec<br>t                                                      | Anyon<br>e                                     | Based on<br>vote count                  | Smoke<br>Test Y                                    |
| <b>Global<br/>workfl<br/>ow<br/>rules</b> | Create a<br>review<br>(<br><b>Enforce</b><br>on) | Reject<br>unless<br>approve<br>d<br>(<br><b>Enforce</b><br>off) | Allow<br>(<br><b>Enfor</b><br><b>ce</b><br>on)                  | Memb<br>ers (<br><b>Enfor</b><br><b>ce</b> on) | Never<br>( <b>Enforce</b><br>off)       | Global<br>Smoke<br>Test<br><br>Global<br>Full Test |

|                                        | Chang<br>elist<br>without<br>a<br>review | Chang<br>elist<br>with a<br>review | On<br>upda<br>te of<br>a<br>revie<br>w in<br>an<br>end<br>state | Coun<br>t<br>votes<br>up<br>from | Automati<br>cally<br>approve<br>reviews | Workfl<br>ow<br>Tests                                                                            |
|----------------------------------------|------------------------------------------|------------------------------------|-----------------------------------------------------------------|----------------------------------|-----------------------------------------|--------------------------------------------------------------------------------------------------|
| Chang<br>elist<br>3456<br>workflo<br>w | Reject                                   | Allow                              | Rejec<br>t                                                      | Memb<br>ers                      | Based on<br>vote count                  | Smoke<br>Test X<br><br>Smoke<br>Test Y<br><br>Global<br>Smoke<br>Test<br><br>Global<br>Full Test |

**Tip:**

- Global workflow rules:
  - When a global workflow rule is set to **Enforce** off, the rule is only used if the project/branch does not have an associated workflow. If the project/branch has an associated workflow, the global rule setting is ignored. This is why the result for the **Changelist with a review** rule is **Allow**, and the **Automatically approve reviews** is **Based on vote count**.
  - When a global workflow rule is set to **Enforce** on, the rule is merged with project/branch workflow and the most restrictive rule setting is used. This is why the result for the **Changelist without a review** rule is **Reject**, the result for the **On Update of a review in an end state** rule is **Reject**, and the result for **Count votes up from** is **Members**.

The global workflow rule mode determines how the global rule interacts with the workflow rules. For more information about global rule modes, see ["Workflow basics" on page 512](#).

- If one or more of the project branches has a moderator, the moderator prevents automatic approval of the review. For more information about moderators, see ["Moderators" on page 455](#).

## Add a workflow

### Note:

- P4 Code Review workflows can be created by any P4 Code Review user.
- Shared P4 Code Review workflows can be viewed by any P4 Code Review user.
- Shared P4 Code Review workflows can be applied to a project or project branch by any P4 Code Review user that is authorized to edit the project.
- The global workflow can be viewed by any P4 Code Review user but can only be edited by a P4 Code Review user with *super* or *admin* privileges, or users that have been made an owner.

### To add a workflow:

1. On the P4 Code Review **Workflows** page, click the **+ Add Workflow** button.

The **Add Workflow** page is displayed:

The screenshot shows the 'Add Workflow' form. It has a title bar 'Add Workflow' with a close button 'x'. The form is divided into several sections:

- Name:** A text input field with a red border.
- Description:** A large text area for describing the workflow.
- Owners:** A section with a button 'Add an Owner'.
- Individuals:** A section with a button 'swarm' and a close button 'x'.
- Shared with others:** A checkbox labeled 'Shared with others'.
- Rules:** A section with five rules, each with a dropdown menu:
  - On commit without a review: Allow
  - On commit with a review: Allow
  - On update of a review in an end state: Allow
  - Count votes up from: Anyone
  - Automatically approve reviews: Never
- Tests:** A section with a button '+Add Test' and a message 'There are no tests associated with this workflow'.
- Footer:** A message 'No projects use this workflow' and two buttons: 'Save' and 'Cancel'.

2. Enter a name for the workflow.
3. **Optional:** provide a description for the workflow.
4. You are automatically added as the owner of the workflow.

**Optional:** add more owners if required. This field auto-suggests groups, and users within P4 Server as you type (up to a combined limit of 20 entries).

**Important:**

- A workflow must have at least one owner.
- If you remove yourself as an owner, you cannot edit this workflow configuration later unless you have *super* user rights.

5. **Optional:** if you want other P4 Code Review users to be able to use this workflow, select the **Shared with others** checkbox.

**Tip:**

Leave the checkbox unselected until you have proved that the workflow rules work as expected. This keeps the workflow private and stops other P4 Code Review users using the workflow until you are happy with it. Once you are happy with the workflow, select the checkbox to share the workflow with other users.

6. **Rules:**

**On commit without a review:**

This rule is applied when a changelist without an associated review is submitted from outside of P4 Code Review.

Select one of the following options:

- **Allow:** the changelist is not checked, the changelist is submitted without a review.
- **Create a review:** the changelist is submitted and a review is created automatically for the changelist.
- **Reject:** the changelist submit is rejected.

**Tip:**

The selected rule is also applied when a changelist is submitted with **#review** in the description. For more information about creating a review by including a keyword in the changelist description, see ["Create a review" on page 669](#).

**On commit with a review:** This rule is applied when:

- A review is committed.
- A changelist with an associated review is submitted from outside of P4 Code Review.

Select one of the following options:

- **Allow:** the changelist review state is not checked. The changelist is committed even if its associated review is not approved.
- **Reject unless approved:** the changelist is only submitted if its associated review is approved and the content of the changelist is identical to the content of the approved review.

**Tip:**

The selected rule is also applied when a changelist is submitted with `#review-nnnnn`, `#replace-nnnnn`, or `#append-nnnnn` in the description (nnnnn = review ID). For more information about adding a changelist to a review by including a keyword in the changelist description, see ["Add a changelist to a review" on page 670](#).

**On update of a review in an end state:**

Used to stop review content being automatically changed for reviews that are in specific states. By default, the protected end states are **Archived**, **Rejected**, and **Approve:Commit**. The end states are set by the P4 Code Review administrator, see [end\\_states](#).

**Tip:**

When a review is in a protected end state, it can still be updated manually by a user from the P4 Code Review UI.

This rule is applied when a changelist is added to a review.

Select one of the following options

- **Allow:** the review is not checked. The changelist is added to the review no matter what the review state is. This is the default setting.
- **Reject:** if the review is in one of the states specified in the `end_state` configurable:
  - The **Add Change** button is disabled for the review.
  - The changelist is rejected if it is added outside of P4 Code Review using `#review-nnnnn`, `#replace-nnnnn`, or `#append-nnnnn` in the description (nnnnn = review ID).

**Count votes up from:**

By default, all of the up votes on a review are counted for the **Minimum up votes** value set on the project/branch the review is associated with. Limit the up votes that are counted to just the members of the project the review is associated with by using this rule.

This rule is applied when a user votes on a review.

Select one of the following options:

- **Anyone:** votes are counted for all reviewers on a review. This is the default setting.
- **Members:** only the up votes of members of the project the review is associated with are counted for the **Minimum up votes** set on projects/branches.

**Tip:**

For instructions on how to set **Minimum up votes** for projects and branches, see [Project minimum up votes](#) and [Branch minimum up votes](#).

**Automatically approve reviews:**

By default, reviews must be manually approved. Enable automatic approval of reviews with this rule.

This rule is applied when a user votes on a review.

Select one of the following options:

- **Never:** reviews are not automatically approved. This is the default setting.
- **Based on vote count:** reviews are automatically approved if:
  - There are no down votes on the review.
  - There are no moderators on the review. If a review has moderators it cannot be automatically approved.
  - All of the [Required reviewers](#) on the review have voted up.
  - The **Minimum up votes** on the review has been satisfied for **each** of the projects and branches the review spans.
  - All of the tests configured to block approval have passed.

**Important:**

Moderators prevent the automatic approval of reviews. For more information about moderators, see ["Moderators" on page 455](#).

**Tip:**

After a review has been automatically approved it needs to be manually committed.

## 7. Tests

**Optional:** add tests to the workflow, the tests are run when a review associated with the workflow is either created/updated or submitted. Selected from the **When** dropdown for the test.

**Tip:**

- Tests are only saved on the workflow when you click the **Save** button.
- If a review is associated with multiple workflows, all of the tests on all of those workflows are run for the review. If the same test is on more than one of the associated workflows, it is only run once.



**Add a test:**

- a. Click **+Add Test** in the Tests area.
- b. Select the test to run from the **Tests** dropdown list.
- c. Select when you want the test to run from the **When** dropdown list.

**Tip:**


- Review content change is when the files or content of files in a review change. A change to the review description does not trigger a test.
- **On Demand** tests are unaffected by the review content change check.

- **On Update** the test will run when:



- a review is created.
- a review is submitted and the review content has changed compared to the previous review version.
- a review is updated and the review content has changed since the previous review version.

**Tip:**

If the review content has not changed since a test last successfully reported a pass, it will not be re-run when a review is versioned or submitted. If a test is still running or has not updated P4 Code Review with a pass state, it will be re-run, even on a submit.

- **On Submit** the test will run when a pre-commit review is submitted or a post-commit review is created.
  - **On Demand** the test will not run automatically, but you can run it manually from the ["Tests" on page 428](#) dialog on the review page.
- d. Select whether P4 Code Review blocks approval if the test fails from the **Blocks** dropdown list:
    - **Nothing** if the test fails it will not block approval.
    - **Approve** if the test fails it will prevent the review from being approved.
  - e. Click the **Accept**  button to add the test to the workflow.

**Edit a test on the workflow:**

- a. Click the **Edit**  button next to the test you want to edit.
- b. The **Test**, **When**, and **Blocks** dropdown lists are displayed for the test.
- c. Make changes as required.
- d. Click the **Accept**  button to confirm your changes.

### Remove a test from the workflow:

Click the **Delete**  button next to the test to remove it from the workflow.

The test is removed immediately, no confirmation is requested. You must save the workflow to complete the test removal.

8. Click **Save**.

#### Note:

The **Save** button is disabled if any required fields are empty.

## Edit a workflow

### Important:

Changes made to a workflow will change the workflow for projects and project branches that use that workflow.

#### Note:

- A workflow can only be edited by the owner of the workflow or a user with *super* user rights.
- To edit the **Global Workflow**, see ["Workflow global rules" on page 742](#).

1. Visit the **Workflow** page you want to edit.
2. Edit the workflow details as required, see ["Add a workflow" on page 521](#) for more details.
3. Click **Save**.

## Delete a workflow

#### Note:

- A workflow can only be deleted by the owner of the workflow, or a user with super user rights.
- The **Global Workflow** cannot be deleted.

### To delete a workflow:

1. Navigate to the workflow you want to delete.
2. Make sure that there are no projects, or branches associated with the workflow.

If projects, or branches are associated with the workflow, the **Delete** button is muted, and the workflow cannot be deleted.

For instructions on how to remove the workflow from a project, or branch, see [Associate a workflow with the project](#) and [Associate a workflow with the project branch](#).

**Note:**

Due to bug in P4 Code Review 2019.3 and earlier, a workflow might falsely report that it is linked to a project when it is not. To remove the false link, navigate to the [Settings page](#) of the project reported as linked and click the **Save** button.

3. Click **Delete** to delete the workflow.

## 9 | Tests

**Important:**

The **Tests** page is only available if Workflow is enabled, workflow is enabled by default.

This chapter covers P4 Code Review test management, including:

- ["Add a test" below](#)
- ["Edit a test" on page 534](#)
- ["Delete a test" on page 534](#)
- To add a test to a workflow, see ["Add a workflow" on page 521](#).

**Related information:**

- For instructions on how to list, search, and view tests, see ["Tests" on page 373](#).
- **Administrators:** For instructions on how to add tests the global workflow, see ["Global workflow" on page 744](#).

### Add a test

**Note:**

- P4 Code Review tests can be created by any P4 Code Review user.
- Shared P4 Code Review tests can be viewed by any P4 Code Review user.
- Shared P4 Code Review tests can be added to a workflow by any P4 Code Review user that is authorized to edit the workflow.
- Shared P4 Code Review tests can be added to the global workflow by any P4 Code Review user that is authorized to edit the global workflow.

**Note:**

In earlier versions of P4 Code Review, global tests were set in the P4 Code Review config.php file. From P4 Code Review 2020.1, tests are added to the global workflow on the P4 Code Review **Workflows** page so that they operate as global tests. If you upgrade from an earlier version of P4 Code Review, your global tests are automatically migrated. Each global test is migrated as follows: the owner is set as the P4 Code Review admin user (not shared), the name is set to the original test name, the description is set to the original test title, and it is added to the **Global Workflow**.

Associate a test with a [workflow](#) to ensure that the test is run when a review associated with that workflow is either created/updated or submitted. Associate a test with the [global workflow](#) to ensure that the test is run whenever a review is either created/updated or submitted. This ensures that the global tests are enforced for all changes even if they are not part of a project.

**To add a test:**

1. On the P4 Code Review **Tests** page, click the **+ Add Test Definition** button.

The **Add Test Definition** page is displayed:

2. Enter a name for the test. This should be a unique name that is 1 to 32 characters in length.
3. **Optional:** provide a description for the test.
4. You are automatically added as the owner of the workflow.

**Optional:** add more owners if required. This field auto-suggests groups, and users within P4 Server as you type (up to a combined limit of 20 entries).

#### Important:

- A test must have at least one owner.
- If you remove yourself as an owner, you cannot edit this test configuration later unless you have *super* user rights.

5. **Optional:** if you want other P4 Code Review users to be able to use this test, select the **Shared with others** checkbox.

#### Tip:

Leave the checkbox unselected until you have proved that the test works as expected. This keeps the test private and stops other P4 Code Review users using the test until you are happy with it. Once you are happy with the test, select the checkbox to share the test with other users.

6. Enter the test request **URL** that will trigger your test suite in the URL text box.

Special arguments are also available to pass details from P4 Code Review to your test suite, see ["Pass special arguments to your test suite" on the facing page](#).

**Note:**

**P4 for Jenkins 1.10.11 and later:** P4 Code Review must send the parameters for the build to Jenkins as a POST request. To do this, enter all the ParameterName=ParameterValue pairs in the **Body** and select **URL Encoded**. Separate each ParameterName=ParameterValuepair with an ampersand & sign. For more information about the supported parameters, see [Building jobs](#) in the *Helix Plugin for Jenkins Guide*.

7. **Optional:** set a **Timeout (in seconds)** as an integer. The timeout is used when P4 Code Review connects to your test suite. If a timeout is not set, the [http\\_client\\_options](#) timeout setting is used. By default, this is 10 seconds.

8. **Optional:** specify parameters that your test suite requires in the request **Body**, select the encoding method using the radio buttons:

- **URL Encoded:** POST parameters are parsed into name=value pairs. This is the default.
- **JSON Encoded:** parameters are passed raw in the POST body.
- **XML Encoded:** parameters are passed in XML format in the POST body.

Special arguments are also available to pass details from P4 Code Review to your test suite, see ["Pass special arguments to your test suite" on the facing page](#).

9. **Iterate tests for affected projects and branches** checkbox

**Optional:** select to create a separate test run for each branch and project the review is associated with. The details generated by the {projects}, {branches}, and {branch} arguments in the **URL** or **Body** are used in the test name displayed in the ["Tests" on page 428](#) dropdown on the review page.

**For example:** if you have the following {projects} and {branches} arguments in your test **URL** or **Body**:

```
{projects} => 'foo,bar',
```

```
{branches} => 'foo:dev,foo:main,bar:dev,bar:candidate',
```

P4 Code Review will iterate through the arguments and create a test run for each of these project branches:

- Project foo dev branch
- Project foo main branch
- Project bar dev branch
- Project bar candidate branch

If **Iterate tests for affected projects and branches** is not selected, only one test will be run for the review, even if multiple branches and projects are associated with the review.

**Note:**

**Private projects:** if a test for a private project is added to a review because **Iterate tests for affected projects and branches** is selected for the test, P4 Code Review honors the private project's permissions and displays it as **Private project** in the review's test list to users that do not have permission to view it.

- For information about the reviews test list, see ["Tests" on page 428](#).
- For information about private projects, see ["Private projects" on page 366](#).

10. **Headers**

**Optional:** enables you to specify name=value pairs to pass to the test suite.

**Tip:**

Headers are only saved on the test when you click the **Save** button.

**Add a header name=value pair:**

- a. Click **+Add header** in the headers area.
- b. Enter the name=value pair in the empty **Header** and **Value** boxes.
- c. The new header is saved when you click **Save** on the test page.

**Edit a header name=value pair:**

- a. Edit the name=value pair in the **Header** and **Value** boxes.
- b. The changes to the header are saved when you click **Save** on the test page.

**Remove a header name=value pair from the test:**

Click the **Delete**  button next to the header to remove it from the test.

The header is removed immediately, no confirmation is requested. You must save the test to complete the header removal.

11. Click **Save**.**Note:**

The **Save** button is disabled if any required fields are empty.

## Pass special arguments to your test suite

You can include special arguments in the request **URL** and **Body** to pass P4 Code Review information about the change that triggered the test run to your test suite. P4 Code Review automatically replaces the arguments with the relevant P4 Code Review information when it calls the test:

**Note:**

The curly braces {} are part of the arguments and must be used.

- {change} the change number
- {status} the status of the shelved change, shelved or committed

- `{review}` the review identifier
- `{version}` the version of the review
- `{reviewStatus}` the P4 Code Review status of the review: **needsReview**, **needsRevision**, **archived**, **rejected**, **approved**, **approved:commit**
- `{description}` the change description of the change used to generate this update. **{description}** cannot be used in the **URL**, it can only be used in the request **Body**.
- `{test}` the name of the test
- `{testRunId}` the test run id
- `{projects}` the project identifiers of projects that are part of the review, null if the review is not part of a project. Comma separated if more than one project is involved in the review.
- `{branches}` the branch identifiers of branches that are part of the review. Branch identifiers are prefixed by a project identifier (with a colon `:` separator by default) if the branch is part of a project, null if the branch is not part of a project. Comma separated if more than one branch is involved in the review.

**Tip:**

Some CI systems, such as Jenkins, escape the default `:` character resulting in a failed test request. If your CI system needs a different separator character, your P4 Code Review *admin* can configure P4 Code Review to use that character.

Be careful when selecting your separator character, some characters can cause issues when calling tests. For example, a `#` character in a URL call means the branch is treated as an anchor and a `%` character in a JSON body encoding is escaped. Both of these will result in a failed test request.

For instructions on configuring the separator character, see "[project\\_and\\_branch\\_separator](#)" on page 722.

- `{branch}` the branch identifiers impacted by the review, comma-separated.

**Note:**

This argument is intended for use in [project tests](#) and is supported by workflow tests for backward compatibility. However, we recommend you use the `{branches}` argument for workflow tests going forwards.

- `{update}` the update callback URL. You can include any or all of the following when calling the update url to update the test run: status, messages, and a url in the body that links to the CI system for that run. They should be formatted in JSON in the body of the POST request. **Status:** valid status values are `running`, `pass`, and `fail`. **Messages:** you can pass a maximum 10 messages, if you provide more than 10 messages only the first 10 are saved. Each message can contain a maximum of 80 characters, any messages with more than 80 characters will be automatically truncated. `{update}` is the preferred option for P4 Code Review 2019.3 and later.



**Important:**

If your test system cannot POST to P4 Code Review, you cannot use update and you must use pass and fail instead.

**Note:****When using {update}:**

- If your build script has access to any messages related to the test execution, pass the messages to P4 Code Review using the {update} URL. P4 Code Review uses the provided message(s) to add to the test results.
- If your build script has access to the results of test execution, include a POST parameter called url when calling the update URL. P4 Code Review uses the provided url to link reviews to the test results. Valid test status values are running, pass, and fail.
- The {update} callback url accepts a JSON body where you can specify any or all of the following: messages, url, and status.

```
{
 "messages": ["My First Message", "My Second Message"],
 "url": "http://jenkins_host:8080/link_to_run",
 "status": "pass"
}
```

- {pass} the tests pass callback URL. From P4 Code Review 2019.3 and later, {update} is preferred. For more details, see the note below.
- {fail} the tests fail callback URL. From P4 Code Review 2019.3, {update} is preferred. For more details, see the note below.

**Note:**

- {update}, {pass}, and {fail} are composed automatically by P4 Code Review, they include P4 Code Review's own per-review authentication tokens.
- **When using {pass} and {fail}:** if your build script has access to the results of test execution, include a GET or POST parameter called url when calling the pass or fail URLs. P4 Code Review uses the provided url to link reviews to the test results.

**Tip:**

P4 Code Review 2019.3 and later still supports {pass} and {fail}, however {update} is preferred because you can also include a message with the test status.

- {projectNames} the project names of projects that are part of the review, null if the review is not part of a project. Comma separated if more than one project is involved in the review.

**Note:**

Not iterated when the **Iterate tests for affected projects and branches** checkbox is selected for the test. If you want to iterate tests, use the {projects} argument.

- {branchName} the branch name(s) impacted by the review, comma-separated.

**Note:**

Not iterated when the **Iterate tests for affected projects and branches** checkbox is selected for the test. If you want to iterate tests, use the {branches} argument.

## Edit a test

**Important:**

Changes made to a test will change the test for workflows that use it.

**Note:**

A test can only be edited by the owner of the test or a user with *super* user rights.

1. Navigate to the **Test** page you want to edit.
2. Edit the test details as required, see ["Add a test" on page 528](#) for more details.
3. Click **Save**.

## Delete a test

**Note:**

A test can only be deleted by the owner of the test or a user with super user rights.

**To delete a test:**

1. Navigate to the test you want to delete.
2. Make sure that there are no workflows associated with the test.  
  
If workflows are associated with the test, the **Delete** button is disabled, and the test cannot be deleted.  
  
For instructions on how to remove the test from a workflow, see ["Add a workflow" on page 521](#).
3. Click **Delete** to delete the test.

# 10 | Integrations

P4 Code Review integrates with a variety of other applications and processes to provide important functionality, such as automated testing and deployment of code in reviews, issue tracking, file previewing, and downloading archives. This chapter describes each of the integrations available with P4 Code Review.

Included integrations:

- [JIRA](#)
- [LibreOffice](#)
- [Zip archive](#)
- [Automated deployment for reviews](#)
- [Automated testing for reviews](#)
- [P4V](#)
- [Slack integration](#)
- [TeamCity](#)

## Jira

P4 Code Review's Jira integration allows code reviews and committed changes to be associated with Jira issues, making it easy to reference associated issues, and see the state of a code review or committed change within Jira. To associate a code review with a Jira issue, include a Jira issue identifier in the review's description, for example SW-1234. P4 Code Review links to the Jira issue and creates a link within the Jira issue back to the code review in P4 Code Review. Multiple Jira issues can be included in the changelist description. As a code review progresses, P4 Code Review updates each associated Jira issue with the review's current status.

For instructions on how to configure basic Jira integration, see ["Jira integration" on the next page](#).

Additionally, if you have the P4 Defect Tracking Gateway (P4 DTG) configured to integrate Perforce jobs with Jira issues. P4 Code Review can be configured to add a link to a Perforce job within a Jira issue. Perforce job links are created when a new Jira or job is created or updated. This gives you direct access to the Perforce job from a link in the **Issue Links** section of the Jira issue.

For instructions on how to configure Jira integration and Perforce job links, see ["Jira integration and Perforce job links" on page 538](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## Jira integration

Basic Jira integration enables:

- **In P4 Code Review:** Hyperlinks are created to Jira issues referenced in changelist descriptions and review descriptions. Jira links in comments are only supported in the classic review page. For example, if Jira issue **SW-1234** is found in the text, the text is linked to the Jira issue in P4 Code Review.

For the legacy Jira linkify, the links for Jira issues in changelist descriptions, review descriptions, and comments open in a new tab in the web browser.

- **In the Issue Links section of the Jira issue:** Creates hyperlinks to the associated P4 Code Review changelist (commit) and review pages.

### Note:

P4 Code Review fetches Jira project identifiers using a worker, once per the worker process' lifetime. See ["Worker configuration" on page 741](#). New Jira projects will not auto-link to Jira until the project identifiers have been updated. By default, project identifiers are updated once every ten minutes.

## Prerequisites for basic Jira integration

- The P4 Code Review workers must be working, see ["Worker status" on page 741](#).
- The Jira module must be enabled, see ["Enabling the Jira module" below](#).
- The Jira user specified in the [Jira module](#) configuration must have permission to view issues, update issues, and add remote links to issues in Jira projects where the Jira integration is required.

To troubleshoot the permissions:

1. Log in to Jira as the P4 Code Review Jira user.
2. Retrieve an issue.
3. Try to create a link.

When a user is provided in the [Jira module](#) configuration, P4 Code Review will use HTTP Basic authentication with the provided username and password. If the username is left blank, P4 Code Review will use Bearer authentication with the password (token) instead.

## Enabling the Jira module

By default, the Jira module is not enabled. To enable the Jira module, add the following configuration block to your [SWARM\\_ROOT](#)/data/config.php file:

```
<?php
// this block should be a peer of 'p4'
'jira' => array(
 'host' => "", // URL for your installed Jira web interface (start with https:// or http://)
 'api_host' => "", // URL for Jira API access, 'host' is used for Jira API access if 'api_host' is
not set
```

```

 'user' => "", // Jira Cloud: the username or email address used to connect to your
Atlassian account
 // Jira on-premises: the username required for Jira API access
 // Jira on-premises with Personal Access Tokens (PATs): the username must
be blank
 'password' => "", // Jira Cloud: a special API token obtained from
https://id.atlassian.com/manage/api-tokens
 // Jira on-premises: the password required for Jira API access
 // Jira on-premises with Personal Access Tokens (PATs): the Personal Access
Token (PAT) obtained from Jira On-premises.
 'relationship' => "", // Jira subsection name links are added to defaults to empty, links added
to the "links to" subsection
 'ignored_users'=> array(), // Reviews or changes by users specified here are not linked to
any Jira issue.
),

```

- **host:** URL for your Jira web interface starting with `https://` or `http://`  
For example, for a Jira cloud instance it might be: `https://my-company-jira.atlassian.net`
- **api\_host** (optional): URL for Jira API access, the **host** URL is used for Jira API access if **api\_host** is not set.
- **user:** The user must have permission to view issues, update issues, and add remote links to issues in Jira projects where the Jira integration is required.
  - **Jira Cloud:** Username or email address used to connect to your Atlassian account.
  - **Jira on-premises:** Username for Jira API access.
  - **Jira on-premises with Personal Access Tokens (PATs):** The username must be blank.
- **password:**
  - **Jira Software Cloud:** A special API token. To obtain your API token:
    - a. Go to [API Tokens](#).
    - b. Log in with your Atlassian administrator credentials.
    - c. Click **Create** and name your API token something recognizable, for example `swarmjira`.
    - d. The API token is displayed in a dialog.
    - e. Copy the token and paste it into the **password** configurable. It is a good idea to save the API token somewhere safe in case you need it in the future.
  - **Jira on-premises:** Password required for Jira API access.
  - **Jira on-premises with Personal Access Tokens (PATs):** A Personal Access Token (PAT) obtained from Jira On-premises. For more information about PATs, see [Using Personal Access Tokens](#).
- **relationship:** sets which subsection of the Jira **Issue Links** section P4 Code Review commit, review, and Perforce job links are added to. P4 DTG is required for Perforce job links, see ["Jira integration and Perforce job links" on the next page](#).

- "": empty: links are added to the **links to** subsection of **Issue Links** section of Jira issues. This is the default value.
- <JobLinksSubsectionName>: creates a new subsection in the **Issue Links** section of Jira issues. Links are added to this subsection.

**Note:**

Existing links are not moved to the subsection defined by the relationship configurable. However, if a link is updated or a new link is added, the link is added to the new subsection. The link is not deleted from its original location.

- ignored\_users: reviews or changes by users specified here are not linked to any Jira issue.

## Jira SSL client configuration

**Tip:**

If your Jira site certificate is from one of the more common Certificate Authorities (such as the CA used by Jira Software Cloud), the following configuration changes are not usually required.

If your Jira site certificate is not from one of the more common Certificate Authorities, you might need to configure the P4 Code Review [http\\_client\\_options](#) configuration block in your [SWARM\\_ROOT/data/config.php](#) file.

For example, if your root and intermediate certificates are stored in a pem format file in /path/to/certs/jira.pem, point to the file with:

```
<?php
// this block should be a peer of 'p4'
'http_client_options' => array(
 'hosts' => array(
 'jira.example.com' => array(
 'sslcafile' => '/path/to/certs/jira.pem',
 'sslpassphrase' => 'keep my JIRA secure',
 'timeout' => 15,
),
),
),
)
```

For more information about configuring your Jira HTTP client options, see ["HTTP client options" on page 698](#).

If you need to troubleshoot P4 Code Review's connection with your Jira instance, see [P4 Code Review to JIRA Connection Troubleshooting](#) in the Perforce Knowledge Base.

## Jira integration and Perforce job links

Jira integration and Perforce job links enable:

- **In P4 Code Review:**
  - Creates hyperlinks to Jira issues referenced in changelist descriptions and review descriptions. Jira links in comments are only supported in the classic review page. For example, if Jira issue **SW-1234** is found in the text, the text is linked to the Jira issue in P4 Code Review.
  - Creates hyperlink to associated Jira issues on the P4 Code Review job page.
- **In the Issue Links section of the Jira issue:**
  - Creates hyperlinks to the associated P4 Code Review changelist (commit) and review pages.
  - When a Perforce job is created, creates a hyperlink to the P4 Code Review job page.
  - If a job is added to a review, creates a hyperlink to the review page.
  - For fixes (p4 fix) to a job, when the `job_field` is set, creates a hyperlink to the P4 Code Review changelist (commit) page.
  - When P4 DTG replication detects a change to an issue, creates a job link for the associated Jira issue if it does not already exist.

For a summary of Jira integration and Perforce job link workflow, see ["Jira integration and Perforce job link workflow" on page 542](#).

## Prerequisites for Jira integration and Perforce job links

- The P4 Defect Tracking Gateway (P4 DTG) must be configured for P4 Code Review, Jira, and P4D, see the [P4 Defect Tracking Gateway Documentation](#).
- The P4 Code Review workers must be working, see ["Worker status" on page 741](#).
- The Jira module must be enabled, see ["Enabling the Jira module" below](#).
- To add Perforce job links to Jira issues, job links must be enabled and the Jira `job_field` must be set, see ["Enabling the Jira module" below](#).

## Enabling the Jira module

### Important:

If P4 Code Review fails to add a Perforce job link to a Jira issue because that Jira issue does not exist, the failure is logged by P4 Code Review. P4 Code Review will not try to add the job link to that Jira issue again.

### Note:

If a Perforce job is updated and the `DTG_DTISSUE` field value is changed, a job link will be added to the new Jira issue. The Perforce job link will not be removed from the original Jira issue.

By default, the Jira module is not enabled. To enable Perforce job links in Jira issues, add the following configuration block to your [SWARM\\_ROOT/data/config.php](#) file:

```

<?php
// this block should be a peer of 'p4'
'jira' => array(
 'host' => "", // URL for your installed Jira web interface (start with https:// or http://)
 'api_host' => "", // URL for Jira API access, 'host' is used for Jira API access if 'api_host'
is not set
 'user' => "", // Jira Cloud: the username or email address used to connect to your
Atlassian account
 // Jira on-premises: the username required for Jira API access
 // Jira on-premises with Personal Access Tokens (PATs): the username must be blank
 'password' => "", // Jira Cloud: a special API token obtained from
https://id.atlassian.com/manage/api-tokens
 // Jira on-premises: the password required for Jira API access
 // Jira on-premises with Personal Access Tokens (PATs): the Personal Access Token
(PAT) obtained from Jira On-premises.
 'job_field' => "", // if P4DTG is replicating Jira issue IDs to a job field, list that field here
 'link_to_jobs' => false, // set to true to enable Perforce job links in Jira, P4DTG and job_
field required
 'delay_job_links' => 60, // delay in seconds, defaults to 60 seconds
 'relationship' => "", // Jira subsection name links are added to defaults to empty, links
added to the "links to" subsection
 'ignored_users' => array(), // Reviews or changes by users specified here are not linked
to any Jira issue
),

```

- **host:** URL for your Jira web interface starting with `https://` or `http://`  
For example, for a Jira cloud instance it might be: `https://my-company-jira.atlassian.net`
- **api\_host** (optional): URL for Jira API access, the **host** URL is used for Jira API access if **api\_host** is not set.
- **user:** The user must have permission to view issues, update issues, and add remote links to issues in Jira projects where the Jira integration is required.
  - **Jira Cloud:** Username or email address used to connect to your Atlassian account.
  - **Jira on-premises:** Username for Jira API access.
  - **Jira on-premises with Personal Access Tokens (PATs):** The username must be blank.
- **password:**
  - **Jira Software Cloud:** A special API token. To obtain your API token:
    - a. Go to [API Tokens](#).
    - b. Log in with your Atlassian administrator credentials.
    - c. Click **Create** and name your API token something recognizable, for example `swarmjira`.
    - d. The API token is displayed in a dialog.



- e. Copy the token and paste it into the password configurable. It is a good idea to save the API token somewhere safe in case you need it in the future.
- **Jira on-premises:** Password required for Jira API access.
- **Jira on-premises with Personal Access Tokens (PATs):** A Personal Access Token (PAT) obtained from Jira On-premises. For more information about PATs, see [Using Personal Access Tokens](#).
- `job_field`: If P4 DTG is replicating Jira issue IDs to a job field, list the job field here.
- `link_to_jobs`:
  - `false`: disable Perforce job links in Jira. The default setting is `false`.
  - `true`: enable Perforce job links in Jira.
- `delay_job_links`: delay set in seconds. Sets the delay between a new Jira or job being created or updated, and P4 Code Review adding the job link to the Jira issue. This configurable is used to avoid a race condition between Jira and the P4 Server. By default `delay_job_links` is set to 60 seconds, if a race condition is experienced increase the delay time.
- `relationship`: sets which subsection of the Jira **Issue Links** section P4 Code Review commit, review, and Perforce job links are added to. P4 DTG is required for Perforce job links.
  - `"` empty: links are added to the **links to** subsection of **Issue Links** section of Jira issues. This is the default value.
  - `<JobLinksSubsectionName>`: creates a new subsection in the **Issue Links** section of Jira issues. Links are added to this subsection.

**Note:**

Existing links are not moved to the subsection defined by the relationship configurable. However, if a link is updated or a new link is added, the link is added to the new subsection. The link is not deleted from its original location.

- `ignored_users`: reviews or changes by users specified here are not linked to any Jira issue.

## Jira SSL client configuration

**Tip:**

If your Jira site certificate is from one of the more common Certificate Authorities (such as the CA used by Jira Software Cloud), the following configuration changes are not usually required.

If your Jira site certificate is not from one of the more common Certificate Authorities, you might need to configure the P4 Code Review `http_client_options` configuration block in your `SWARM_ROOT/data/config.php` file.

For example, if your root and intermediate certificates are stored in a pem format file in `/path/to/certs/jira.pem`, point to the file with:

```
<?php
// this block should be a peer of 'p4'
'http_client_options' => array(
```

```
'hosts' => array(
 'jira.example.com' => array(
 'sslcafile' => '/path/to/certs/jira.pem',
 'sslpassphrase' => 'keep my JIRA secure',
 'timeout' => 15,
),
),
),
)
```

For more information about configuring your Jira HTTP client options, see ["HTTP client options" on page 698](#).

If you need to troubleshoot P4 Code Review's connection with your Jira instance, see [P4 Code Review to JIRA Connection Troubleshooting](#) in the Perforce Knowledge Base.

## Jira integration and Perforce job link workflow

This section describes the workflow for Jira integration and job links in Jira issues when you request a new review from P4V.

### Tip:

The process for adding a Perforce job and creating a review differs for the other clients but the P4 Code Review worker and P4 Defect Tracking Gateway (P4 DTG) workflow is the same.

1. Create a new Jira issue in your Jira system. P4 DTG detects the new Jira issue and creates a new Perforce job for that Jira issue.
2. Add the new Perforce job to a pending changelist in P4V. See [Add a job to a pending changelist](#) in the [P4 Visual Client \(P4V\) Documentation](#).
3. Request a new review for the pending changelist in P4V. See [Request a review](#) in the [P4 Visual Client \(P4V\) Documentation](#).

P4V passes the review request to the P4 Code Review worker queue. The review request is processed by the next available P4 Code Review worker, the new review is created. P4 Code Review checks its Jira configuration. If the `job_field` exists in the Perforce job, P4 Code Review adds a link to the new Perforce job in the **Issue Links** section of the Jira issue.

### To check that these steps have been completed and to view the links:

1. Open the new review. The associated Perforce jobs are displayed on the review page below the review description.
2. Click the Perforce job link to open the job. The review link is displayed on the job page below the job description.
3. Click the **Details** tab on the Perforce job page to view the **DTG\_DTISSUE** field.
4. Click the Jira link in the **DTG\_DTISSUE** field to view the Jira issue. The review and Perforce job links are displayed in the **Issue Links** section of the Jira issue page.

## LibreOffice

LibreOffice is a free power-packed open source personal productivity suite. When LibreOffice is installed on the server hosting P4 Code Review, P4 Code Review automatically detects its presence and uses LibreOffice to prepare PDF previews of a variety of file types, including:

- Word documents (.doc, .docx)
- PowerPoint presentations (.ppt, .pptx)
- Excel spreadsheets (.xls, .xlsx)
- Visio diagrams (.vsd)
- Rich-text files (.rtf).



Depending on your server's platform and distribution, LibreOffice may be provided as multiple packages and not all packages may be installed by default. If certain filetypes do not preview as expected in P4 Code Review, you may need to install these optional packages to include all of the file handling capabilities of LibreOffice.

For more information on LibreOffice, see [LibreOffice website](#).

## Limitations

The LibreOffice integration has several limitations:

- Document previews in P4 Code Review may appear different than on a desktop system if the document's fonts are not installed on the server hosting P4 Code Review and LibreOffice. In addition, LibreOffice has some limitations rendering Microsoft Office file types, so LibreOffice-generated previews may differ from what you see using Microsoft Office.

- Large files may require a notable amount of time to preview. Very large files may exhaust the resources of the server hosting P4 Code Review, causing the preview to fail and temporarily impacting P4 Code Review performance.
- P4 Code Review is currently not able to detect or show differences in LibreOffice-supported file types.

## Installation

We recommend that you install LibreOffice from your OS distribution, via `apt-get`, `yum`, etc.

The minimal packages, and their transitive dependencies required for P4 Code Review are:

- `libreoffice-calc`
- `libreoffice-draw`
- `libreoffice-impress`
- `libreoffice-writer`
- `libreoffice-headless` (RHEL only)

## Zip archive

When the `zip` command-line tool is installed on the P4 Code Review server, P4 Code Review allows users to download ZIP archives of files and folders. You can download a Zip archive using the **Download zip** option from the following P4 Code Review pages:

- ["Review display" on page 411](#) page.
- ["Changelists" on page 319](#) page.
- ["Files" on page 303](#) page.

## Installation

Install the `zip` command-line tool from your operating system distribution using `apt-get`, `yum`, etc.

## Configuration

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

Configure the archiving feature with the following configuration block in the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
// this block should be a peer of 'p4'
```

```
'archives' => array(
 'max_input_size' => 512 * 1024 * 1024, // 512M (in bytes)
 'archive_timeout' => 1800, // 30 minutes
 'compression_level' => 1, // 0-9
 'cache_lifetime' => 60 * 60 * 24, // 1 day
),
```

The `max_input_size` key specifies the maximum file/folder content size that can be processed into a ZIP archive. The default value permits up to 512 megabytes of content to be compressed. Smaller values limit the amount of file/folder content but provide faster downloads; larger values can allow increased scanning, syncing, compressing, and downloading times.

The `archive_timeout` key specifies the amount of time, in seconds, to allow P4 Code Review to prepare the ZIP archive for downloading. Shorter times can limit the practical size of a ZIP archive, depending on the performance of your network and the filesystem hosting P4 Code Review; even with a generous `max_input_size` setting, if `archive_timeout` seconds have elapsed, the archive operation is terminated.

The `compression_level` key specifies the compression level to use, and must be within the range 0 to 9. 0 means no compression, 9 means maximum compression. As this value is increased, smaller ZIP archives may result, but may require greater compression time. P4 Code Review uses the default of 1, which provides a reasonable tradeoff of fast compression times with light compression that can still result in an archive notably smaller than the original file/folder content.

The `cache_lifetime` key specifies the desired maximum age of cached ZIP archives. Increasing the value increases the amount of time that ZIP archives exist in the cache, which can improve the user experience for frequently downloaded files. However, ZIP archives can be quite large (depending on the size of your depot within the P4 Server) and can require significant disk storage. Decreasing the value can mitigate the amount of disk space required for the cache; the tradeoff is that frequently accessed ZIP archives may need to be generated more frequently, which can have an impact on CPU and disk resources.

## Download files as a Zip archive

To download the zip archive navigate to the review, changelist, file or folder:

### Note:

The **Download zip** option is not displayed if the zip command-line tool is not installed on the P4 Code Review server. For information about installing, and configuring the zip command-line tool, see [Zip archive](#).

When you select the **Download zip** option, P4 Code Review performs the following steps:

1. Scans the files/folders:
  - Checks that you have permission to access their contents, according to the P4 Server protections.
  - Checks that the total file size is small enough to be processed by P4 Code Review.
2. Syncs the file contents to the P4 Code Review server from the P4 Server.

3. Creates the ZIP archive by compressing the file content.
4. Starts a download of the generated ZIP archive.

**Note:**

- You might not see all of the above steps; P4 Code Review caches the resulting ZIP archives so that repeated requests to the same files/folders can skip the sync and compress steps whenever possible.
- If an error occurs while scanning, syncing, or compressing, P4 Code Review indicates the error.

## TeamCity

P4 Code Review's TeamCity integration allows you to configure TeamCity to post build statuses as comments to your code reviews of shelved files.

For instructions on how to configure P4 Code Review with TeamCity, see [Integrating TeamCity with Perforce](#).

# 11 | Administration

This section covers administration and configuration of P4 Code Review.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the "Reload Configuration button" on page 720.

In this section:

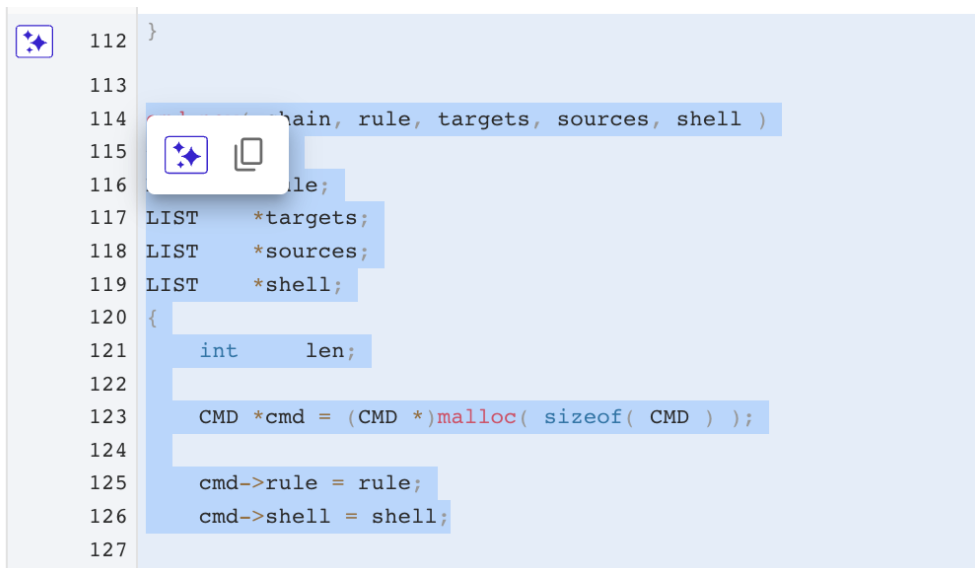
## AI for code analysis

P4 Code Review enhances the code review process by using an AI model (similar to those from OpenAI) to generate explanations for code diffs in a file, making it easier for developers to understand and review changes.

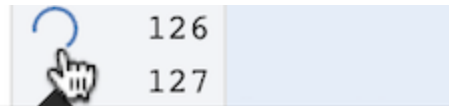
You can send the whole diff of a file for AI code analysis or just a selection.

To send the whole diff for AI code analysis, click the AI button  in the numbered panel.

To send a selection of code, first select the code to send for AI code analysis, then click the AI button



A loading tooltip will appear while the AI summary is being generated.



## Fetching AI summary for line numbers 114-126

The AI code analysis is returned as text in a pop window.

114

✦ Open AI With Explain Code On GPT4 changed summary of changes in the following diff ^

This code defines two C functions, `cmd_free1` and `cmd_free2`, which are almost identical. Each takes a single argument, a pointer to a `CMD` object, and frees the memory allocated for it. The `CMD` type is not defined in the provided code, but it seems to contain two fields: `args` and `shell`.

- `lol_free(&cmd->args)`: This function call is intended to free the memory allocated for the `args` field of the `CMD` object. This suggests that `args` is likely a pointer to some dynamically allocated memory or a complex structure which itself contains dynamically allocated memory. Note that the `&` operator is used to pass a pointer to the `args` field to the function (so its original position can be NULL'd out after free).
- `list_free(cmd->shell)`: This function call is intended to free the memory allocated for the `shell` field of the `CMD` object. This suggests that `shell` is likely a pointer to some sort of list data structure with dynamically allocated memory.
- `free((char *)cmd)`: This function call frees the memory allocated for the `CMD` object. Here, the `cmd` pointer is cast to a `char *` before it is freed.

In C, it's important to deallocate any memory you've dynamically allocated once you're done using it, to prevent memory leaks. These two functions are presumably part of a larger program where `CMD` objects are dynamically allocated and deallocated regularly.

Note that the style of function declaration is K&R C ("Kernighan & Ritchie"), an older style that pre-dates ANSI C. In modern C, the functions' arguments would be declared inside the parentheses in the function header: `void cmd_free1(CMD *cmd) { ... }`.

[Read less](#)

[Discard summary](#)

Only the diff (or selected diff content), filename, and the file extension are sent to the AI vendor for analysis. P4 Code Review does not send the whole file.

P4 Code Review has been tested with the following AI vendors and models:

- OpenAI - gpt-4
- LM Studio - all provided models which include:
  - lmstudio-community/gemma-3-1B-it-qat-GGUF
  - lmstudio-community/Qwen3-1.7B-GGUF



- `lmstudio-community/DeepSeek-R1-Distill-Llama-8B-GGUF`
- `lmstudio-community/Meta-Llama-3.1-8B-Instruct-GGU`

The configuration to integrate AI into P4 Code Review is specified within an `ai_review` block in the `SWARM_ROOT/data/config.php` file. By default, `ai_review` block is disabled. To enable AI-driven code explanations for a diff, set `enabled` to `true`.

## How to set up OpenAI integration for P4 Code Review

To integrate OpenAI with P4 Code Review, follow these steps:

1. Create an OpenAI account or log in to your an OpenAI account [here](#).
2. Once logged in, create a new project within the OpenAI platform.
3. Generate a secret API key:
  - a. Navigate to the project settings.
  - b. Create a new secret key.
  - c. Copy the generated key.
4. To configure AI review integration for P4 Code Review, add the `ai_review` block configuration in `SWARM_ROOT/data/config.php` file, as in the following example:

```
<?php
'ai_review' => array(
 // Please read Perforce's Generative AI policy before enabling this feature.
 // See https://www.perforce.com/generative-ai-policy
 'enabled' => false, // set to true to enable the AI review feature
 'data_retention_lifetime' => '30 days', // Delete AI summary records that are older
 // than the
 // value provided. By default, this value is 30 days.
 // Enter a value in 'days' or 'months'.
 // For example, '30 days' or '2 months'.
 // A cron job is required to remove the
 // AI summaries on a schedule.
 // See "Set up a cron job to delete AI summaries" on page 212.
 'timeout' => 30, // Setting timeout for the request sent from p4 code review to AI
 vendor
 // Timeout set in seconds
 'ai_vendors' => array(
 'ai_model1' => array(
 'ai_vendor' => 'openAI', // Name of the AI provider being used
 'ai_package_id' => '1', // Ensure this remains as '1'. Do not modify the ai_
 package_id.
 'ai_package_key' => 'openaiwithexplaincodeongpt4', // This is intended for
 future
 // use when we will support
 // multiple models and have an
```

```

// AI configuration page in the UI.
'ai_package_value' => 'Open AI With Explain Code On GPT4', // This is used
to display

// the model type in the
// summary of the
// AI vendor's response
'ai_model' => 'gpt-4',
'ai_package_type' => "Explain the following code:", // This is the prompt for
the AI.

// You can modify the prompt for
// purposes other than explaining
// the code, or to request the
// output in a specific language.
'ai_key' => $SECRET_KEY, // Paste your copied OpenAI API key here
'ai_min_char_limit'=> 4, // Minimum number of characters required in the
content

// submitted to the AI vendor for analysis.
// Defaults to 4 characters.
'ai_max_char_limit'=> 31000, // The maximum number of characters
// that can be submitted to the AI vendor for analysis
// Defaults to 31000 characters.
),
),
),

```

Replace `$SECRET_KEY` with the API key you copied earlier.

5. Make sure your OpenAI account has enough tokens to use the API. If needed, recharge your account.

This setup will enable OpenAI's GPT-4 model to explain the diff code in your P4 Code Review.

## How to set up LM Studio integration for P4 Code Review

To integrate LM Studio with P4 Code Review, follow these steps:

1. Launch LM Studio.
2. In LM Studio, download the AI model you want to use.
3. In the Developer tab, select the relevant AI model and configure your server settings.
4. To configure AI review integration for P4 Code Review, add the `ai_review` block configuration in `SWARM_ROOT/data/config.php` file, as in the following example:

```

<?php
'ai_review' => array(
 // Please read Perforce's Generative AI policy before enabling this feature.
 // See https://www.perforce.com/generative-ai-policy
 'enabled' => false, // set to true to enable the AI review feature
 'data_retention_lifetime' => '150 days', // Delete AI summary records that are

```

older than the

```

// value provided. By default, this value is 30 days.
// Enter a value in 'days' or 'months'.
// For example, '30 days' or '2 months'.
// A cron job is required to remove the
// AI summaries on a schedule.
// See "Set up a cron job to delete AI summaries" on page 212.
'timeout' => 500, // Setting timeout for the request sent from p4 code review to AI
vendor
 // Timeout set in seconds
 'ai_vendors' => array(
 'ai_model1' => array(
 'ai_vendor' => 'lmstudioai', // Name of the AI provider being used
 'ai_package_id' => '1', // Ensure this remains as '1'. Do not modify the ai_
package_id.
 'ai_package_key' => 'LMStudioPackage', // This is intended for future
// use when we will support
// multiple models and have an
// AI configuration page in the UI.
 'ai_package_value' => 'AI LM Studio', // This is used to display
// the model type in the
// summary of the
// AI vendor's response
 'ai_package_type' => "Explain the following code:", // This is the prompt for
the AI.
// You can modify the prompt for
// purposes other than explaining
// the code, or to request the
// output in a specific language.
 'api_end_point' => 'http://{LMStudioRunningHost}/v1/chat/completions',
// Replace {LMStudioRunningHost} with your
// LM Studio server IP address.
 'ai_min_char_limit' => 4, // Minimum number of characters required in the
content
// submitted to the AI vendor for analysis.
// Defaults to 4 characters.
 'ai_max_char_limit' => 31000, // The maximum number of characters
// that can be submitted to the AI vendor for analysis
// Defaults to 31000 characters.
),
),
),

```

This setup will enable LM Studio to explain the diff code in your P4 Code Review.

## How to set up a generic AI model integration for P4 Code Review

To integrate a generic AI model with P4 Code Review, do the following:

1. [Modify the AI configuration for P4 Code Review](#)
2. [Use an API endpoint which resembles the OpenAI vendor's request and response](#)

## Modify the AI related configuration for P4 Code Review

Modify the `ai_review` block in the `SWARM_ROOT/data/config.php` file as follows:

```
<?php
'ai_review' => array(
 // Please read Perforce's Generative AI policy before enabling this feature.
 // See https://www.perforce.com/generative-ai-policy
 'enabled' => false, // set to true to enable the AI review feature
 'data_retention_lifetime' => '30 days', // Delete AI summary records that are older than
the
 // value provided. By default, this value is 30 days.
 // Enter a value in 'days' or 'months'.
 // For example, '30 days' or '2 months'.
 // A cron job is required to remove the
 // AI summaries on a schedule.
 // See "Set up a cron job to delete AI summaries" on page 212.
 'timeout' => 30, // Setting timeout for the request sent from p4 code review to AI vendor
 // Timeout set in seconds
 'ai_vendors' => array(
 'ai_model1' => array(
 'ai_vendor' => 'genericAI', // Name of the AI provider being used
 'ai_package_id' => '1', // Ensure this remains as '1'.
 // Do not modify the ai_package_id.
 'ai_package_key' => 'GenericAIPackage', // This is intended for future
 // use when we will support
 // multiple models and have an
 // AI configuration page in the UI.
 'ai_package_value' => 'Generic AI with explain code on custom AI model', // This is
used to
 // display the model type
 // type in the summary of
 // the AI vendor's response
 'ai_model' => 'GenericAIModel',
 'ai_package_type' => "Explain the following code:", // This is the prompt for the
 // AI. You can modify the prompt
 // for purposes other than
 // explaining the code, or to
 // request the output in a
 // specific language.
 'api_end_point' => 'https://apitoconnectaimodel.com/dummyAnalyzeCode', //
Replace with
 // you AI vendor
 // end point
 'ai_min_char_limit' => 4, // Minimum number of characters required in the content
 // submitted to the AI vendor for analysis.
```

```

 // Defaults to 4 characters.
 'ai_max_char_limit'=> 31000, // The maximum number of characters
 // that can be submitted to the AI vendor for analysis
 // Defaults to 31000 characters.
),
),
),

```

## Example API endpoint request and response format

Use an AI model that has a request and response structure. The following example is for the OpenAI model, however, you should configure the API request to match your the request and response structure of your AI vendor.

### Example API endpoint

POST /api/v10/AiAnalysis/dummyanalyzeCode

### API endpoint request format

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/AiAnalysis/dummyanalyzeCode"
```

The "mybodyfilename.txt" file contains:

```
{
 "content": "Explain the following code of file nodeofNotequal.c: static void
checkForLowerCase();"
}
```

### API endpoint response format

```
{
 "data": {
 "id": "chatcmpl-Aqe3vlnskpD30pkUMzBTJ9ONxou7l",
 "object": "chat.completion",
 "created": 1737110419,
 "model": "gpt-4o-2024-08-06",
 "choices": [
 {
 "index": 0,
 "message": {
 "role": "assistant",
 "content": "The line of code `static void checkForLowerCase();` in a C file named
`nodeofNotequal.c` is a function
 declaration. Here's what this line signifies:\n\n1. **`static` Keyword**: \n - In the
context of a
 function declaration, `static` means that the function has internal linkage. This
```

means the function

can only be called within the same source file (in this case, ``nodeofNotequal.c``). It is not visible

or accessible from other source files that might be linked together to form an executable.

`\n\n2. **`void` Return Type**:` `\n - `void`` indicates that the function

``checkForLowerCase`` does not

return any value.`\n\n3. **Function Name**:` `\n - `checkForLowerCase`` is the name of the function.

It is customary to choose a name that indicates the purpose or action of the function. In this case,

it suggests that the function may be used to check for lowercase characters or strings, though the

exact behavior would depend on the function's implementation.`\n\n4. **Parameter List**:` `\n -`

The empty parentheses ``()`` indicate that the function does not take any arguments.`\n\nThis is just a`

declaration, meaning it informs the compiler about the existence of the function

``checkForLowerCase``

and its signature (return type and parameters) before it is used in the code. The actual logic or body

of the function would appear later in the file, outside of the declaration.`\n\nTo fully understand what`

``checkForLowerCase`` does, you would need to look at the function's definition, which would specify the

code that executes when the function is called."

```
"refusal": null
},
"logprobs": null,
"finish_reason": "stop"
}
],
"usage": {
 "prompt_tokens": 27,
 "completion_tokens": 338,
 "total_tokens": 365,
 "prompt_tokens_details": {
 "cached_tokens": 0,
 "audio_tokens": 0
 },
 "completion_tokens_details": {
 "reasoning_tokens": 0,
 "audio_tokens": 0,
 "accepted_prediction_tokens": 0,
 "rejected_prediction_tokens": 0
 }
},
"service_tier": "default",
```

```
"system_fingerprint": "fp_50cad350e4"
}
}
```

## How to write the Custom AI Adapter in P4 Code Review to support an in-house AI model

Use this guide to write and integrate a Custom AI Adapter for your in-house or on-premise AI vendor with P4 Code Review.

Currently, the Custom AI Adapter has been tested with the following AI vendors and models:

- `lmstudio-community/gemma-3-1B-it-qat-GGUF (/v1/completions)`
- `granite-embedding-278m-multilingual-GGUF (/v1/embeddings)`

If you are developing a Custom AI Adapter for a different vendor or model, additional configuration may be required. Refer to the vendor's documentation for specific configuration details.

**Tip:** We strongly recommend implementing and testing it first in a sandbox environment, test account, or a new instance of P4 Code Review before integrating it into your production environment.

To write a Custom AI Adapter in P4 Code Review, complete these three stages:

- ["Copy and modify GenericAIAdapter" below](#)
- ["Validate the response" on page 558](#)
- ["Add custom AI adapter to config files" on page 560](#)

### Copy and modify GenericAIAdapter

1. Go to the folder `module/AiAnalysis/src/Service/` and create the copy of file `GenericAIAdapter.php` within the same folder.
2. Name the copied file `CustomAIAdapter.php`. Below is an example of `GenericAIAdapter.php`.

```
<?php
/**
 * Perforce Swarm
 *
 * @copyright 2024 Perforce Software. All rights reserved.
 * @license Please see LICENSE.txt in top-level folder of this distribution.
 * @version <release>/<patch>
 */
```

```

namespace AiAnalysis\Service;

use AiAnalysis\Filter\IAiAnalysis;
use AiAnalysis\Helper\IAiAnalysisHelper;
use Application\Config\ConfigException;
use Application\Config\ConfigManager;
use Application\Config\IConfigDefinition;
use Application\Log\SwarmLogger;
use Laminas\Http\Response;
use OpenAI;

class GenericAIAdapter extends AbstractAIAdapter implements
AiServiceInterface
{

 const LOG_PREFIX = GenericAIAdapter::class;

 /**
 * performAI, note this uses the content body and ignores any data provided
 * Login to Swarm using the credentials provided
 *
 * @param mixed $data
 * @return array
 */
 public function executeAIRequest(mixed $data): array
 {
 $logger = $this->services->get(SwarmLogger::SERVICE);
 try {
 $fileName = ($data[IAiAnalysis::FILE_NAME]) ?
 'of the file: ' . $data[IAiAnalysis::FILE_NAME] . '': '';
 $content = $data[IConfigDefinition::AI_PACKAGE_TYPE] . $fileName
 .
 $data[IAiAnalysisHelper::CONTENT_TO_ANALYZE];
 $args['content'] = $content;
 $result = $this->request($this->getApiUrl(), false, $args);

 if ($this->validateResult($result)) {
 $logger->trace(sprintf("[%s]: Content generation successful ",
self::LOG_PREFIX));
 return [IAiAnalysisHelper::CONTENT_GENERATED => $result-
>choices[0]->message->content,

```



```

 self::ERROR => null,
 self::CODE => Response::STATUS_CODE_200];
 } else {
 $logger->trace(sprintf("[%s]: Failed at content generation ", self::LOG_
PREFIX));
 return [IAiAnalysisHelper::CONTENT_GENERATED => "",
 self::ERROR => is_object($result) && property_exists($result, 'error')
? $result->error :
 'Response not received from AI Vendor',
 self::CODE => is_object($result) && property_exists($result, 'code') ?
$result->code :
 Response::STATUS_CODE_500];
 }
} catch (\Exception | \Throwable $exception) {
 $logger->debug(
 sprintf(
 "[%s]: Error occurred at Generic AI Adapter %s",
 self::LOG_PREFIX,
 $exception->getMessage()
)
);
 $statusCode = $exception->getCode();
 return [
 IAiAnalysisHelper::CONTENT_GENERATED => null,
 'error' => $exception->getMessage(),
 'code' => $statusCode === 0 ? Response::STATUS_CODE_500 :
$statusCode,
];
}
}

/**
 * This method will fetch the API key from config, required to execute the
generic AI request
 * @throws ConfigException
 * @return string
 */
private function getApiUrl() : string
{
 $config = $this->services->get(IConfigDefinition::CONFIG);

```

```

 return ConfigManager::getValue($config, IConfigDefinition::AI_REVIEW_
 AI_VENDORS_AI_MODEL1_API_END_POINT);
 }
}

```

3. In CustomAIAdapter.php, change the class from GenericAIAdapter to CustomAIAdapter.
4. In CustomAIAdapter.php, within the executeAIRequest function, replace \$arg with the request format used by your chosen AI model. By default, this is \$args['content'] = \$content; however, below is an example of a different \$arg request format.

```

$args = [
 'messages' => [
 [
 'role' => 'user',
 'content' => $content
],
],
];
//Note here $content is prompt ($data[IConfigDefinition::AI_PACKAGE_TYPE])
+ filename with its extension + diff from review

```

5. Save the changes to in CustomAIAdapter.php and build the request. The following line of code in CustomAIAdapter.php sends the request to the API endpoint (api\_end\_point) specified in config.php file.

```

$result = $this->request($this->getApiUrl(), false, $args);

```

## Validate the response

1. Go to the folder module/AiAnalysis/src/Service/, open the file AbstractAiAdapter.php and copy the function validateResult.
2. Open CustomAIAdapter.php and overwrite validateResult with the copied function.
3. Modify the copied function in CustomAIAdapter.php with your AI model's response format. An example of an unmodified validateResult function is as follows:

```

protected function validateResult(object $result): bool

```

```

{
 if (!property_exists($result, 'error') &&
 property_exists($result, 'choices') &&
 isset($result->choices[0]) && is_object($result->choices[0]) &&
 property_exists($result->choices[0], 'message') &&
 is_object($result->choices[0]->message) &&
 property_exists($result->choices[0]->message, 'content')) {
 return true;
 }
 return false;
}

```

If your AI model response is not a object, you need to modify the `validateResult` function in `CustomAIAdapter.php` so that it can handle the response.

For example, if the AI vendor returns the response as an array, update the `validateResult` function to parse the response. An example of an updated `validateResult` function that can handle the response as an array is shown below.

```

protected function validateAIResponse(array $result): bool
{
 if (is_array($result) && isset($result[0]) && is_object($result[0]) &&
 property_exists($result[0], 'embedding'))
 {
 return true;
 }
 return false;
}

```

4. If your AI model's response format is different from the example above, modify the format accordingly. You will also need to update `IAiAnalysisHelper` within the `CustomAIAdapter.php` file. An example of different AI response format is shown below:

```

{
 "id": "chatcmpl-B9MBs8CjcvOU2jLn4n570S5qMJKcT",
 "choices": [
 {
 "index": 0,
 "message": {
 "role": "assistant",
 "content": "Hello! How can I assist you today?",
 }
 }
]
}

```

```

 "refusal": null,
 "annotations": []
 },
}
],
}

```

5. Update the `Services.php` file located in `module/Application/src/Config/` to include an entry for `CustomAIAdapter` alongside the other AI adapters. Below is an example for the AI adapter entries in this file:

```

const LM_STUDIO_AI = 'lmStudioAI';
const CUSTOM_AI = 'customAI';

```

6. Update the `AIServiceFactory.php` file located in `module/AiAnalysis/src/Factory/` to include an entry for `CustomAIAdapter` alongside the other AI adapters. Below is an example for the AI adapter entries in this file:

```

const OPENAIADAPTER = 'openAI';
const GENERICAIDAPTER = 'genericAI';
const LMSTUDIOADAPTER = 'lmStudioAI';
const CUSTOMADAPTER = 'customAI';

```

## Add custom AI adapter to config files

Two config files require updating with entries for the custom AI adapter. These files are:

- The `module.config.php` file found in `module/AiAnalysis/config/`.
- The `config.php` file found in `SWARM_ROOT/data/`.

To update the files, follow these steps:

1. Open the `module.config.php` file found in `module/AiAnalysis/config/`.
2. At the top of the `module.config.php` file, add a `use` statement for the `CustomAIAdapter` above the current `use` statement for the `GenericAIAdapter`. Below is an example of these `use` statements:

```

use AiAnalysis\Service\CustomAIAdapter;
use AiAnalysis\Service\GenericAIAdapter;

```

3. Inside the same `module.config.php` file, find the `service_manager` array and add the following entries.

In the `aliases` array, add:

- `AIServiceFactory::CUSTOMAIADAPTER => CustomAIAdapter::class`

In the `factories` array, add:

- `CustomAIAdapter::class => AIServiceFactory::class`
- `Services::CUSTOM_AI => CustomAIAdapter::class`

Below is an example of the updated `service_manager` array in the `module.config.php` file.

```
'service_manager' => [
 'aliases' => [
 IDao::AI_ANALYSIS_DAO => AiAnalysisDAO::class,
 AIServiceFactory::OPENAIADAPTER => OpenAIAdapter::class,
 AIServiceFactory::GENERICAIADAPTER => GenericAIAdapter::class,
 AIServiceFactory::LMSTUDIOAIADAPTER => LMStudioAIAdapter::class,
 IAIAnalysis::NAME => AiAnalysis::class,
 IAIAnalysis::DISCARD_ANALYSIS_FILTER => DiscardAnalysis::class,
 AIServiceFactory::CUSTOMAIADAPTER => CustomAIAdapter::class
],
 'factories' => [
 AiAnalysisDAO::class => InvokableServiceFactory::class,
 OpenAIAdapter::class => AIServiceFactory::class,
 Services::OPEN_AI => OpenAIAdapter::class,
 GenericAIAdapter::class => AIServiceFactory::class,
 Services::GENERIC_AI => GenericAIAdapter::class,
 LMStudioAIAdapter::class => AIServiceFactory::class,
 Services::LM_STUDIO_AI => LMStudioAIAdapter::class,
 AiAnalysis::class => InvokableServiceFactory::class,
 AiAnalysisChecker::class => InvokableServiceFactory::class,
 AiAnalysisCharLimitChecker::class => InvokableServiceFactory::class,
 AiAnalysisDataRetentionLifetimeChecker::class =>
 InvokableServiceFactory::class,
 DiscardAnalysis::class => InvokableServiceFactory::class,
 CustomAIAdapter::class => AIServiceFactory::class,
 Services::CUSTOM_AI => CustomAIAdapter::class,
],
],
```

4. Open `config.php` file found in `SWARM_ROOT/data/`.
5. Update the `ai_review` module with the values for the custom AI Adapter.

**Important:** Make sure you update `api_end_point` with your AI vendor end point .

Below is an example of a modified `ai_review` module:

```
'ai_review' => array(
 // Please read through the https://www.perforce.com/generative-ai-policy
 // before you enable this feature
 'enabled' => true,
 'data_retention_lifetime' => '150 days',
 'timeout' => 500,
 'ai_vendors' => array(
 'ai_model1' => array(
 'ai_vendor' => 'customAI',
 'ai_package_id' => '1', // id should be one & should not modify it
 'ai_package_key' => 'CustomAIPackage', // This is for future purposes
 // when we will be supporting multiple models & will have an AI configuration page
 // on UI
 'ai_package_value' => 'Custom AI', // This is used for displaying the model
 // type on the AI vendor response summary
 'ai_package_type' => "Explain the following code ", // This is prompt type
 'api_end_point' => 'http://myAIVendorAPI/chat/completion',
 'ai_min_char_limit' => 10,
 'ai_max_char_limit' => 70000
),
),
),
```

After completing these steps, a custom AI adapter has been added to P4 Code Review that supports your in-house AI model.

## Automated deployment for reviews

### Important:

The project level test and deploy code features will be deprecated in a later P4 Code Review release. We recommend you use test integration to automatically deploy code within a review. For more information, see ["Add a test" on page 528](#).

Deploying code in a code review automatically involves enabling **Automated Deployment** in your project's configuration and providing a *trigger URL*. When the *trigger URL* is requested, P4 Code Review expects a deployment program to be executed.

When the deployment processing ends, P4 Code Review expects either a *success callback URL* or *failure callback URL* to be requested by your deployment program. These callback URLs should include a url parameter (either via GET or POST); when a valid-looking URL is included, clicking the deployment status indicator directs the user to the specified URL. This is intended to facilitate easy viewing of the successfully deployed review, or a report indicating why the deployment failed. The url parameter is mandatory for successful deployments, but is optional for failures.

1. Navigate to the project page.
2. Click the project **Settings** tab to display the **Project Settings** page.
3. **Automated Deployment** checkbox: select **Enable** to display the configuration field:

Automated Deployment ☒ Enable

`http://deploy-server/deploy?change={change}`

A URL that will trigger a deployment when reviews are created or updated.  
Some special [arguments](#) are supported. [See help for more details.](#)

4. Provide a URL that triggers your deployment execution.

Special arguments are available to inform your deployment program of various details from P4 Code Review:

**Note:**

**P4 for Jenkins 1.10.11 and later:** P4 Code Review must send the parameters for the build to Jenkins as a POST request. To do this, enter the parameters in the **Post Body** and select **URL Encoded**.

`{change}`  
The change number

`{status}`  
Status of the change, *shelved* or *submitted*

`{review}`  
The review's identifier

`{project}`  
The project's identifier

`{projectName}`  
The project's name

`{branch}`  
The branch identifier(s), comma-separated

`{branchName}`  
The branch name(s), comma-separated

`{success}`  
Deployment successful callback URL

`{fail}`  
Deployment failure callback URL

## Automated testing for reviews

**Important:**

The project level test and deploy code features will be deprecated in a later P4 Code Review release. We recommend you use test integration to automatically deploy code within a review. For more information, see ["Add a test" on page 528](#).

**Tip:**

From P4 Code Review 2020.1, the preferred method for defining tests is on the P4 Code Review ["Tests" on page 528](#) page. This enables you to:

- Associate a test with a [workflow](#) to ensure that the test is run when a review associated with that workflow is either created/updated or submitted.
- Associate a test with the [global workflow](#) to ensure that the test is run whenever a review is either created/updated or submitted. This ensures that the global tests are enforced for all changes even if they are not part of a project.

Integrating P4 Code Review with a test suite involves enabling **Automated Tests** in your project's configuration and providing a *trigger URL*. When the *trigger URL* is requested, P4 Code Review expects your test suite to be executed. When the tests complete, P4 Code Review expects an *update callback URL*, a *pass callback URL*, or a *fail callback URL* to be requested by your test suite.

1. Navigate to the **Project** page.
2. Click the project **Settings** tab to display the **Project Settings** page.
3. Ensure that paths in each named branch configured for the project do not overlap with paths in other named branches.
4. **Automated tests** checkbox: select **Enable** to display the configuration fields:

Automated Tests ☒ Enable

`http://test-server/build?change={change}`

A URL that will trigger automated tests to run when reviews are created or updated.

Some special [arguments](#) are supported. [See help for more details.](#)

POST Body

`foo=bar&baz=buzz`

URL Encoded ▼

Optional data to POST to the above URL. The special arguments supported for URLs can also be used here.

5. Provide a URL that triggers your test suite execution.  
Special arguments are available to inform your test suite of various details from P4 Code Review. P4 Code Review automatically replaces the arguments with the relevant P4 Code Review information when it calls the test:



**Note:**

**P4 for Jenkins 1.10.11 and later:** P4 Code Review must send the parameters for the build to Jenkins as a POST request. To do this, enter the parameters in the **Post Body** and select **URL Encoded**.

`{test}`

The name of the test

`{testRunId}`

The test run id

`{change}`

The change number

**Tip:**

If your CI system supports `change=now`, you can use this instead of the `change={change}` to make sure your review is always tested against the latest version of the submitted code.

`{status}`

Status of the change, *shelved* or *submitted*

`{review}`

The review's identifier

`{version}`

The version of the review

`{description}`

The change description of the change used to generate this update. `{description}` cannot be used in the **URL**, it can only be used in the **POST Body**.

`{project}`

The project's identifier

`{projectName}`

The project's name

`{branch}`

The branch identifier(s) impacted by the review, comma-separated

`{branchName}`

The branch name(s) impacted by the review, comma-separated

`{update}`

The update callback URL. You can include any or all of the following when calling the update url to update the test run: status, messages, and a url in the body that links to the CI system for that run. They should be formatted in JSON in the body of the POST request. **Status:** valid status values are running, pass, and fail. **Messages:** you can pass a maximum 10 messages, if you provide more than 10 messages only the first 10 are saved. Each message can contain a maximum of 80 characters, any messages with more than 80 characters will be automatically truncated. `{update}` is the preferred option for P4 Code Review 2019.3. For more details, see the note below.

`{pass}`

Tests pass callback URL. From P4 Code Review 2019.3, `{update}` is preferred. For more details, see the note below.

`{fail}`

Tests fail callback URL. From P4 Code Review 2019.3, {update} is preferred. For more details, see the note below.

**Note:**

- P4 Code Review 2019.3 and later still supports {pass} and {fail}, however {update} is preferred because you can also include a message with the test status.
- {update}, {pass}, and {fail} are composed automatically by P4 Code Review. They include P4 Code Review's own per-review authentication tokens.

6. **Optional:** specify any parameters that your automated tests require that must be sent via HTTP POST in the **POST Body** field. The POST parameters can include the special arguments listed above.

Select the format of the POST parameters, either **URL Encoded** or **JSON Encoded**.

- **URL Encoded:** POST parameters are parsed into name=value pairs.
- **JSON Encoded:** parameters are passed raw in the POST body.

## Configuring Jenkins for P4 Code Review integration

**Important:**

Your Jenkins host needs to be able to communicate with the P4 Code Review host, and the P4 Code Review host needs to be able to communicate with the Jenkins host. Ensure that the appropriate DNS/host configuration is in place, and that each server can reach the other via HTTP/HTTPS.

1. Install the p4-plugin for Jenkins:

For instructions on installing the p4-plugin, see in the [Installation](#) chapter of the [P4 for Jenkins Documentation](#).

2. Configure a Jenkins project:

**Note:**

**P4 for Jenkins 1.10.11 and later:** P4 Code Review must send the parameters for the build to Jenkins as a POST request. To do this, enter the parameters in the **Post Body** and select **URL Encoded**.

- a. Specify the job name so that it matches the project identifier used in the trigger URL, as defined [below](#).

For example, the computed value of {projectName}\_{branchName}.

Or, edit the trigger URL to use the Jenkins job name you specify.

- b. Make the build parameterized to accept these parameters (note that these are named to match up with the script that is called):

{test}

The name of the test

`{testRunId}`  
The test run id

`{status}`  
Whether the changelist to be tested is *shelved* or *submitted*

`{change}`  
Changelist # to run tests against

`{review}`  
The review's identifier

`{version}`  
The version of the review

`{branchName}`  
The branch name(s) impacted by the review, comma-separated

`{update}`  
The URL to POST to update the test run. You can include any or all of the following when calling the update url to update the test run: status, messages, and a url in the body that links to the CI system for that run. They should be formatted in JSON in the body of the POST request. **Status:** valid status values are `running`, `pass`, and `fail`. **Messages:** you can pass a maximum 10 messages, if you provide more than 10 messages only the first 10 are saved. Each message can contain a maximum of 80 characters, any messages with more than 80 characters will be automatically truncated. `{update}` is the recommended way of reporting test status.

**Important:**

If your test system cannot POST to P4 Code Review, you cannot use `update` and you must use `pass` and `fail` instead.

**Note:****When using {update}:**

- If your build script has access to any messages related to the test execution, pass the messages to P4 Code Review using the {update} URL. P4 Code Review uses the provided message(s) to add to the test results.
- If your build script has access to the results of test execution, include a POST parameter called url when calling the update URL. P4 Code Review uses the provided url to link reviews to the test results. Valid test status values are running, pass, and fail.
- The {update} callback url accepts a JSON body where you can specify any or all of the following: messages, url, and status.

```
{
 "messages" : ["My First Message", "My Second Message"],
 "url" : "http://jenkins_host:8080/link_to_run",
 "status": "pass"
}
```

**{pass}**

The URL to wget if the build succeeds. From P4 Code Review 2019.3, {update} is preferred. For more details, see the note below.

**{fail}**

The URL to wget if the build fails. From P4 Code Review 2019.3, {update} is preferred. For more details, see the note below.

**Note:**

**When using {pass} and {fail}:** if your build script has access to the results of test execution, include a GET or POST parameter called url when calling the pass or fail URLs. P4 Code Review uses the provided url to link reviews to the test results.

**For example:** ?url=https://jenkins.example.com/view/job/<jobname>/<jobid>/

**Tip:**

P4 Code Review 2019.3 and later still supports {pass} and {fail}, however {update} is preferred because you can also include a message with the test status.

- Select **Perforce Software** for the **Source Code Management** section.

**Important:**

You might see **Perforce** in the **Source Code Management** section. This represents an earlier community-provided Perforce plugin that does not include support for P4 Code Review.

- d. Set up credentials and workspace behavior as needed.

For instructions on configuring credentials and workspaces, see the [Credentials](#) and [Workspaces](#) sections of the [P4 for Jenkins Documentation](#).

**Important:**

- If your P4 Server is configured for P4 AS, the service user credentials used for automated testing must not use P4 AS.
- The client workspace configured in Jenkins must have a view that includes the paths defined for that branch in P4 Code Review.

3. Configure your P4 Code Review project to run automated tests with a URL and parameters like this:

**Tip:**

In this example, the build system is passed the changelist number of the latest version of the review in {change}. If the test is rerun at a later date, the build system will rerun the test against the same {change} number, and this will probably not be the latest version of the codebase. If you want the review to always be tested against the latest versions of submitted code, replace `change={change}` with `change=now`. Your CI system must support `change=now` to use this feature.

Automated Tests ☒ Enable

`http://jenkins_host:8080/job/{projectName}_{branchName}/review/build`

A URL that will trigger automated tests to run when reviews are created or updated.

Some special [arguments](#) are supported. [See help for more details.](#)

Optional data to POST to the above URL.

POST Body

`status={status}&review={review}&change={change}&update={update}`

URL Encoded ▼

**Important:**

For Jenkins, the job name needs to match the job identifier in the URL. In the example above, this is the computed value of `{projectName}_{branchName}`. If you prefer a different naming scheme in Jenkins, replace `{projectName}_{branchName}` in the URL above with the project name actually defined in Jenkins.

**Important:**

If security is enabled in Jenkins, the trigger URL needs to include credentials. Follow these steps:

- Create a Jenkins user that will trigger P4 Code Review builds. For example swarm.
- Log into Jenkins as the new user.
- Click on the user's username in the Jenkins toolbar.
- Scroll down to **API Token**.
- Click **Show API Token**.
- Incorporate the value of the **API Token** into the P4 Code Review trigger URL. For example, if the username is swarm and the API Token value is 832a5db7e5500c1288324c1441460610, the P4 Code Review trigger URL and parameters should be:

Automated Tests ☒ Enable

```
https://swarm:832a5db7e5500c1288324c1441460610@jenkins_host:8080/job/{projectName}_{branchName}/review/build
```

A URL that will trigger automated tests to run when reviews are created or updated.

Some special [arguments](#) are supported. [See help for more details.](#)

Optional data to POST to the above URL.

POST Body

```
status={status}&review={review}&change={change}&pass={pass}&fail={fail}
```

URL Encoded ▼

## Avatars

P4 Code Review uses *avatars*, images that represent users and groups responsible for events in activity streams, projects, reviews, etc.

Avatars are retrieved from an avatar provider; the default provider is [Gravatar](#). P4 Code Review sends an identifier to the avatar provider (for [gravatar.com](#), an MD5 hash of the user's or group's email address), and the provider returns the configured image (if one exists). If no avatar is defined with the provider or the requests fails for any reason, P4 Code Review selects an avatar from its internal collection.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the

new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

Configure the avatar lookups in the avatars configuration block in the [SWARM\\_ROOT/data/config.php](#) file. Here is an example:

```
<?php
// this block should be a peer of 'p4'
'avatars' => array(
 'http_url' => 'http://www.gravatar.com/avatar/{hash}?s={size}&d={default}',
 'https_url' => 'https://secure.gravatar.com/avatar/{hash}?s={size}&d={default}',
),
```

- `http_url`: specifies the http URL P4 Code Review uses to lookup avatars. If the URL is not defined, P4 Code Review uses the default `gravatar.com` URL.
- `https_url`: specifies the https URL P4 Code Review uses to lookup avatars. If the URL is not defined, P4 Code Review uses the default `gravatar.com` URL.

The avatars array URL that P4 Code Review uses depends on the settings of the P4 Code Review Apache server protocol and the ["external\\_url" on page 603](#) protocol, the securest of these protocols is used.

Several replacement values are available for inclusion in the URLs:

- `{user}`  
The current P4 Code Review userid, Perforce groupid, or empty string
- `{email}`  
The current P4 Code Review user's or group's email address, or empty string
- `{hash}`  
The MD5 hash of the P4 Code Review user's or group's email address, or 00000000000000000000000000000000 if no email address is configured
- `{default}`  
The value blank for a transparent GIF (allowing users or groups without avatars to fallback to P4 Code Review's internal avatars) or the value `mm` for a *mystery man* used in circumstances where no user or group identifier is known
- `{size}`  
the size P4 Code Review would like in pixels for both the width and height, without units, e.g. 64

The URL you specify must include one of `{user}`, `{email}`, or `{hash}` to properly select a user-specific or group-specific avatar. The URL should include `{size}` to assist P4 Code Review's presentation. `{default}` is not necessary, but helps provide a consistent avatar experience.

**Tip:**

You can adjust the appearance of avatars by using custom CSS, see ["Adjust the appearance of avatars" on page 779](#).

**Note:**

By default, gravatar.com serves only G-rated avatar images. If your P4 Code Review users and groups wish to use PG-, R-, or X-rated images, you need to configure the avatar lookup URLs with the appropriate rating flag. For example, to allow avatars with G or PG ratings, the configuration would look like:

```
<?php
// this block should be a peer of 'p4'
'avatars' => array(
 'http_url' => 'http://www.gravatar.com/avatar/{hash}?r=pg&s={size}&d={default}',
 'https_url' => 'https://secure.gravatar.com/avatar/{hash}?r=pg&s={size}&d=
{default}',
),
```

For more information on gravatar.com image requests, see [Profiles as a Service](#).

## Disable avatar lookups

If you wish to disable avatar lookups altogether and simply use P4 Code Review's internal avatars, set each URL to "" (empty string). For example:

```
<?php
// this block should be a peer of 'p4'
'avatars' => array(
 'http_url' => "",
 'https_url' => "",
),
```

## Backups

P4 Code Review stores all of the information it requires to operate within the P4 Server. This includes project definitions, code reviews, comments, followers, and more. Code reviews are largely built on top of P4 Server's shelving feature, and most other records are stored in custom counters called *keys*.

Therefore, the standard recommendations for backing up your P4 Server also apply when backing up your P4 Code Review data.

For more information about backup and recovery, see [Backup and recovery in P4 Server Administration Documentation](#).

In addition, your P4 Code Review configuration and any modifications you might make to the provided modules, templates, CSS, JavaScript, etc. also need to be backed up. The [SWARM\\_ROOT/data](#) directory contains the configuration, as well as temporary working files and browser session storage.

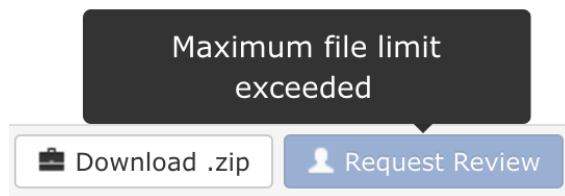


## Changelist files limit

Changelists or reviews with many files may take longer than usual to display in P4 Code Review. This is because of the time required to get the file listing from the P4 Server and then display the large number of files in the browser.

File listings with thousands of files, may cause P4 Code Review to run out of memory. Increasing the amount of memory for P4 Code Review can help, but the UI may work slower than usual or not at all depending on the amount of memory available in your browser.

With P4 Code Review 2025.1 and later, if the number of files in a changelist exceeds the `max_changelist_files` value, the **Request Review** button in the changelist page is disabled and you cannot request a review. This is to prevent P4 Code Review from running out of memory.



You can adjust the file limit by changing the `max_changelist_files` value in the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
 'p4' => array(
 'max_changelist_files' => 1000,
),
```

The default value of `1000` is sufficient for most P4 Code Review installations.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

Before changing the `max_changelist_files` value, consider the following effects of this change:

- A large number of files in a change list may cause P4 Code Review UI issues. There is no real advantage to displaying more than 10,000 files at a time.
- Smaller `max_changelist_files` values can interfere with changelists as the reduced file limit could prevent files from being displayed in P4 Code Review.

## Client integration

**Note:** P4 Code Review 2022.3 or later only works with P4 Visual Client (P4V) 2021.3 or later.

P4V and P4 for Visual Studio can integrate with P4 Code Review. To indicate how these applications should connect with P4 Code Review, P4 Code Review sets the `P4.Swarm.URL` property set in P4 Server. P4V and P4 for Visual Studio read this property, and if set, they connect to the specified URL to make P4 Code Review API calls. If the property is unset, P4 Code Review integration features are disabled.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

When `P4.Swarm.URL` is set, P4V provides the following integration features:

- **Request a review:** requests a review for pending or committed changelists.
- **Update a review:** updates a review from the current state of a pending changelist. This works for changelists that are already associated with a review, or for unassociated changelists.
- **Open review in P4 Code Review:** opens the review associated with the selected changelist in your system's default web browser.
- **Review Id and State columns:** adds **Review Id** and **Review State** columns to both the **Pending** and **Submitted** tabs.

By default, the first P4 Code Review worker auto-detects the URL it is running under and sets `P4.Swarm.URL` accordingly. P4 Code Review checks and updates this property every 10 minutes.

## Customized P4 Code Review installations

For customized P4 Code Review installations, the auto-detected URL might not use the correct protocol, hostname, or port. In these scenarios, you can disable the URL auto-detection by editing the `SWARM_ROOT/data/config.php` file and setting the `auto_register_url` item to `false` in the `p4` configuration block. For example:

```
<?php
 'p4' => array(
 'auto_register_url' => false,
),
```

If you disable this feature, you should:

- ["Manually set P4.Swarm.URL in the P4 Server"](#) below
- ["Set the external\\_url in P4 Code Review"](#) on the facing page

### Manually set P4.Swarm.URL in the P4 Server

If you disable `auto_register_url`, you should manually set the `P4.Swarm.URL` property in P4 Server to the URL for your P4 Code Review installation:

```
p4 property -a -n P4.Swarm.URL -v https://myswarm.url:port/
```

Replace `https://myswarm.url:port/` with the URL for your P4 Code Review installation.

To find out what the current `P4.Swarm.URL` value is for all users, run:

```
p4 property -AI -n P4.Swarm.URL
```

**Note:**

P4V uses an integration timeout, specified in the `P4.Swarm.Timeout` property, to limit delays in the P4V user interface. The default timeout is 10 seconds.

To change the integration timeout, run:

```
p4 property -a -n P4.Swarm.Timeout -v 10
```

Replace the `10` with the desired timeout in seconds. Increasing the timeout could cause notable delays in the P4V user interface, and decreasing the timeout could cause sporadic integration failures if P4 Code Review's API responses take longer than the specified timeout.

## Set the `external_url` in P4 Code Review

If you disable `auto_register_url`, you should usually set the `external_url` for P4 Code Review. However, you do not need to set the `external_url` if the `P4.Swarm.URL` is manually set. For information about setting the `external_url` for P4 Code Review, see "[external\\_url](#)" on page 603.

## Troubleshooting: Behavior issues when `hostname` does not match `P4.Swarm.URL`

If the host in `P4.Swarm.URL` does not match `hostname` in the `config.php` then the **Show full context** and **Show more lines** buttons found in the review page in P4 Code Review do not work.

This is because the auto register creates a second instance of `P4.Swarm.URL` with a higher sequence number.

For example, if the `hostname` in the `config.php` is initially `P4CR-server` and it is then updated to the IP address (both of which work in a browser) then the following two instances of `P4.Swarm.URL` are created:

- `P4.Swarm.URL = http://P4CR-server (any) #none`
- `P4.Swarm.URL = http://192.168.56.101 (any) #1`

These instances are picked up by P4V which then prevents the **Show full context** from working.

**Fix:** To fix this, delete the instance of the property that does not match the `hostname` in the `config.php`. Run the following to delete the instance:

```
p4 property -d -n P4.Swarm.URL -s1
```

## Comment attachments

P4 Code Review supports attaching arbitrary files to comments in code reviews and jobs.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the

new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the "Reload Configuration button" on page 720.

## Comment attachment storage

To store files attached to comments, P4 Code Review looks for a depot named `//.swarm`. As P4 Code Review does not create this depot, you need to create it, or specify another depot that the P4 Code Review *admin* user can write to.

The depot used to store P4 Code Review attachments must be:

- a [classic depot](#) (stream, unload, archive, and spec depots are not supported)
- a local depot (remote depots are not supported)
- a depot that the P4 Code Review *admin* user can write to but other users cannot access. For more information about permissions, see [Authorizing access](#) in the [P4 Server Administration Documentation](#).

### Tip:

It is best practice to use a depot directory that is not used to store any other files, this stops other users having access to the attachments stored in the depot. We recommend that you use `//.swarm`

## Create a directory for the comment attachments

We recommend that you use `//.swarm` because it is best practice to use a depot directory that is not used to store any other files. This prevents other users from having direct access to the comment attachments.

1. To create a `//.swarm` depot, run the following as a user with *admin*-level privileges:  

```
$ p4 depot .swarm
```
2. Ensure that the P4 Code Review *admin* user can write to the `//.swarm` depot and make sure that other users cannot access it.

For information on creating depots, see [Working with depots](#) in [P4 Server Administration Documentation](#).

## Specify the location for the comment attachments

By default P4 Code Review looks for a depot named `//.swarm`, to specify a different depot path for comment attachments, use the `depot_storage` configuration block in the `SWARM_ROOT/data/config.php` file.

It is mandatory to include the `depot_storage` configuration block in the P4 Code Review configuration. Failure to include the `depot_storage` configuration block will result in the option to add an attachment to a comment being unavailable in the P4 Code Review UI.

Replace `depot_name` with the depot where comment attachments are stored. The P4 Code Review *admin* needs to be able to write to this depot but make sure that other users cannot access it:

```
<?php
// this block should be a peer of 'p4'
'depot_storage' => array(
 'base_path' => '//depot_name',
),
```

**Tip:**

The `base_path` can be more than just the depot name, for example `//depot/perforce/.swarm` would work.

## Maximum comment attachment size

By default, the maximum file size of a single comment attachment is limited to:

- **Ubuntu:** 8Mb
- **RHEL 8 and later:** 2Mb

## Increasing the maximum attachment file size

You can increase the maximum attachment size if required.

### Ubuntu

Edit the `php_value upload_max_filesize` and `php_value post_max_size` configurables in the `.htaccess` file.

### RHEL 8 and later

Edit the `upload_max_filesize` and `post_max_size` configurables in the `/etc/php.ini` file.

## Comment mentions

This section describes how to customize how P4 Code Review treats user `@mentions` and group `@@mentions` in auto-complete drop-downs, comments, changelists, and review descriptions.

**Note:**

- `@mentions` for users is only available for P4 Code Review 2017.1 and later.
- `@@mentions` for groups is only available for P4 Code Review 2017.3 and later.
- If your P4 Server is configured to be case sensitive this will also apply to the names specified in your exclude lists.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the

new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

#### By default:

- When a user uses an *@mention* or *@@mention* in a comment, P4 Code Review will auto-complete for all P4 Code Review users or groups respectively.
- When a user edits the reviewers on a review, P4 Code Review will auto-complete for all P4 Code Review users and groups.
- When a user edits the author of a review, P4 Code Review will auto-complete for all P4 Code Review users.

This behavior can be fine tuned with the mentions configuration block in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
'mentions' => array(
 'mode' => 'global',
 'user_exclude_list' => array('super', 'swarm-admin'),
 'group_exclude_list' => array('testers', 'writers'),
),
```

- `mode` controls the scope of the auto-complete dropdown that is shown when the user starts typing an *@mention* or an *@@mention* in a review comment:
  - `global` displays all users/groups in P4 Code Review, this is the default value
  - `projects`: only displays project members (users and groups) in the auto-complete list
  - `disabled` disables the auto-complete drop down so that it is not displayed
- `user_exclude_list` users in the exclude list:
  - are not suggested when you *@mention* them in a comment, edit reviewers, or edit the author in P4 Code Review.
  - are not added to the review if you manually add them as an *@mention* in a P4 Code Review comment or a review description.
  - are not added to the review if they are *@mentioned* in a changelist description.
  - cannot be added to the review with the P4 Code Review API.

**Tip:**  
**When you add a user to the exclude list:**

- the user is not removed from any reviews that they are already on.
- the user will stop receiving email notifications for those reviews.

- `group_exclude_list` groups in the exclude list:

- are not suggested when you *@@mention* them in a comment or edit reviewers in P4 Code Review.
- are not added to the review if you manually add them as an *@@mention* in a P4 Code Review comment or review description.
- are not added to the review if they are *@@mentioned* in a changelist description.
- cannot be added to the review with the P4 Code Review API.

**Tip:**  
**When you add a group to the exclude list:**

- the group is not removed from any reviews that it is already on.
- the group members will stop receiving email notifications for those reviews.
- if a member of an excluded group is also on the review as a user or as a member of another group, that user will continue to receive emails for the review.

## Regular expressions in exclude lists

Regular expressions can be used in user and group exclude lists but they should be used with care to avoid accidentally excluding the wrong users or groups. P4 Code Review exclude lists can use any valid PHP regular expression but not all of them make sense for the exclude lists.

**Note:**

P4 Code Review adds a `^` character before your regular expression and a `$` character after it. For example: `^your-reg-ex-pattern$` You must take this into account when constructing your regular expressions.

**Tip:**

- You can test the results of your regular expressions before using them, see [regular expressions 101](#).
- Case sensitivity for regular expressions in the exclude lists is determined by the case sensitivity setting of the P4 Server.

## Regular expression examples

Useful regular expressions are shown below:

**Exact match only:**

'super' only users or groups named *super* are excluded.

**Beginning with:**

'admin.\*' users or groups beginning with *admin* are excluded.

**For example:** the following would be excluded:

*administrator*, *admin-group*, and *admin*

**Ending with:**

'.\*admin' users or groups ending with *admin* are excluded.

**For example:** the following would be excluded:

*test-admin*, *buildadmin*, and *admin*

**Ending with digits**

'.\*[0-9]+' users or groups ending with digits are excluded.

**For example:** the following would be excluded:

*test03*, *admin-10*, *tester\_123456*

**Containing:**

'.\*admin.\*' users or groups that contain *admin* are excluded.

**For example:** the following would be excluded:

*test-admin*, *buildadmin*, *administrator*, *admin-group*, and *admin*

## Comments

This section provides information on how to delay comment notifications, and configure comment threading.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## Comment notification delay

By default, comment notifications are delayed to allow reviewers to add or edit comments as they progress through a review without sending a notification for each individual comment on the review. Comment notifications are rolled up into a single notification and sent either manually by the reviewer, or automatically after the notification delay time has been exceeded.

The delay countdown is reset each time the reviewer adds or edits a comment on the review, by default the notification delay time is set to 30 minutes.

- If you are commenting on more than one review, each of the reviews that you are commenting on has its own notification delay countdown that only applies to the comments that *you* make on *that* review.



- If another reviewer is making comments on the same review as you, *that* reviewer has their own notification delay timer for *that* review.
- If you manually send a delayed comment notification, the notification will only contain the comments that *you* made on *that* review.

**Tip:**

The comment notification delay does not delay the posting of the comments, only the comment notification is delayed.

**Note:**

Comment notifications are only delayed for comments on reviews. Comments on commits or jobs produce notifications immediately.

To change the comment notification delay time, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the comments block:

```
<?php
'comments' => array(
 'notification_delay_time' => 1800, //Default to 30 minutes 1800 seconds
),
```

`notification_delay_time`: Specifies the comment notification delay time in seconds.

- Set to `0` to send the comment notification immediately the **Post** button is clicked.
- The default value if `notification_delay_time` is not specified is **1800** seconds (30 minutes).

## Comment threading

By default, you can reply to comments, replies are displayed in a thread below the parent comment. Comment thread depth is set to 4 by default, this means you can have up to 4 levels of replies for a parent comment. The **Reply** link is not displayed for replies at or above the maximum thread depth set for P4 Code Review. If the parent comment is archived, replies are archived with the parent, see "[Archiving comments](#)" on page 343. Comment threading depth is set by the `max_depth` configurable.

**Tip:**

If the thread depth is reduced by a P4 Code Review administrator, earlier replies at a deeper level will continue to be displayed but you cannot reply to them.

To configure comment threading, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the comments block:

```
<?php
'comments' => array(
 'threading' => array(
 'max_depth' => 4, // default depth 4, to disable comment threading set to 0
),
),
```

`max_depth`: Specifies maximum depth of a comment thread.

- Set to `0` to disable comment threading.
- The default value if `max_depth` is not specified is `4`.

## Commit credit

When you use P4 Code Review to commit a review, but you are not the review's author, P4 Code Review gives credit to the review author by default. Activity stream entries and email notifications include both the committer and review author's details.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

If you prefer P4 Code Review's original behavior, which was to give credit only to the committer, you can do so by editing the `$SWARM_ROOT/data/config.php` file and setting the `commit_credit_author` item to `false` in the `reviews` configuration block. For example:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'commit_credit_author' => false,
),
```

## Commit-edge deployment

P4 Code Review can connect to a P4 Server configured to use the *commit-edge architecture*, which is a specific replication configuration that employs a *commit* server and one or more *edge* servers. This configuration distributes the compute, storage, and network requirements for improved performance and geographic distribution.

For more information on P4 Server's commit-edge architecture, see the [Commit-edge](#) chapter in the [P4 Server Administration Documentation](#).

**Tip:**

P4V users may have problems when P4V connects to an edge server and your commit and edge tickets are different. For more information on P4V authentication for commit-edge deployment, see ["P4V Authentication"](#) on page 584.

## P4 Code Review commit-edge configuration

This section describes what configuration changes need to be made to P4 Code Review and the P4 Servers to run P4 Code Review with a commit-edge deployment. Before you begin making changes ensure that your P4 Server commit-edge deployment is working correctly, and that P4 Code Review has been installed and configured.

### Configure P4 Code Review for a commit-edge deployment:

1. Configure P4 Code Review to connect to the P4 Commit server, see ["P4 Code Review configuration" on page 179](#). This is called the **Commit Server P4 Code Review Instance**.

When P4 Code Review is connected to the **Commit Server P4 Code Review Instance**, the first worker detects this situation and sets a key in the P4 Server, `P4.Swarm.CommitURL`, to an auto-detected URL.

2. **Configure the P4 Code Review triggers:**

- a. Configure the P4 Code Review triggers on all of the Commit and Edge servers, see ["Installing triggers" on page 187](#).

#### Important:

- Save all of the P4 Code Review trigger scripts in exactly the same place on each of the Commit and Edge servers. This is important because they all share the same trigger table.
- You must install the trigger dependencies on all of the Commit and Edge servers, see ["Trigger dependencies" on page 100](#).

- b. Configure `SWARM_HOST` in the `swarm-trigger.conf` file of all of the Commit and Edge servers to point to the **Commit Server P4 Code Review Instance**, see ["Installing triggers" on page 187](#).
3. Configure all of the Commit and Edge servers to point to the **Commit Server P4 Code Review Instance URL**. This is done by setting the `P4.Swarm.URL` to the **Commit Server P4 Code Review Instance URL** on each of the Commit and Edge servers, see [Swarm integration properties](#) in the [P4 Server Administration Documentation](#).
4. Your P4 Code Review configuration is now complete but you must check that it is working correctly before using it in production, see ["Validate your P4 Code Review installation" on page 227](#).

### Configure edge servers to promote shelved changes

If your P4 Server is configured as a commit-edge deployment, and your normal connection is to an edge server, P4 Code Review refuses to start reviews for shelved changes that have not been promoted to the commit server.

Within P4 Code Review, this means that the **Request Review** button does not appear for unpromoted shelved changes. Outside of P4 Code Review, attempts to start reviews for unpromoted shelved changelists appear to do nothing. Ask your P4 Server administrator for assistance if you cannot start a review.

An administrator of the P4 Server can automatically promote shelved changes to the commit server by setting the configurable `dm.shelve.promote` to `1`.

## P4V Authentication

When using P4V's P4 Code Review integration in a commit-edge deployment, users may encounter authentication errors; such errors can result from incorrect configuration of login tickets in distributed environments. Essentially, the problem is that while P4V is connected to an edge server, P4 Code Review is connected to the commit server, and the login tickets do not match.

If P4V users see the `error Host requires authentication`, the solution we recommend is to forward login requests to the commit server. This can be achieved by executing the following commands as a user with *operator* or *super* privileges in the P4 Server.

For example, the following `p4 configure set` commands run against the commit server set `auth.id` globally and enable login forwarding from the `edge1` and `replica1` servers:

```
p4 configure set auth.id=myAuthName
p4 configure set edge1#rpl.forward.login=1
p4 configure set replica1#rpl.forward.login=1
```

Replace *myAuthName* with the authentication identifier for your P4 Server.

For more information, see our Knowledge Base article [Single Ticket Login in Distributed Environments](#), and the `p4 serverid` command in the [P4 CLI Reference](#).

## Troubleshooting: Azure Application Gateway issues with P4 Code Review and P4V

When running P4 Code Review behind an Azure Application Gateway (such as when using a load balancer with P4V), you may encounter authentication issues due to missing client IP information.

To resolve this, you need to configure the Azure Application Gateway to include an X-Forwarded-For header. Specifically, add a rewrite rule to insert or preserve the X-Forwarded-For header so that P4 Code Review can correctly identify the original client IP address. This can only be done by the Azure administrator managing your environment. It is not performed by P4 Remote Administration.

For more information, see [Configuring Azure Application Gateway for accessing Kibana](#).

## Commit timeout

When a code review contains many files, or large files, or both, committing the review within P4 Code Review can take some time. The default configuration, within the `SWARM_ROOT/data/config.php` file:

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
```

```
'commit_timeout' => 1800, // 30 minutes
),
```

The `commit_timeout` key is expressed in seconds. If a commit operation takes longer than this limit, it is terminated. It is likely that a terminated commit requires administrator intervention to complete the commit using another client.

## Configuration overview

This section provides an overview of all the possible configuration blocks in the [SWARM\\_ROOT/data/config.php](#) file.

**Important:** While the syntax of this example is correct, it includes configuration values that **cannot work**.

Ensure that you adjust the configuration appropriately for your P4 Code Review installation before using this example in testing or production.

If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

Click on any link in the configuration overview below to see a detailed description of that configurable.

```
<?php
return array(
 'activity' => array(
 'ignored_users' => array(
 'p4dtguser',
 'system',
),
),
 'ai_review' => array(
 // Please read Perforce's Generative AI policy before enabling this feature.
 // See https://www.perforce.com/generative-ai-policy
 // The AI configuration below can be used to integrate an OpenAI vendor with P4 Code
Review.
 // To integrate a generic AI model with P4 Code Review,
 // see "How to set up a generic AI model integration for P4 Code Review" on page 551.
 'enabled' => false, // set to true to enable the AI review feature
 'data_retention_lifetime' => '30 days', // Delete AI summary records that are older than
the
 // value provided. By default, this value is 30 days.
 // Enter a value in 'days' or 'months'.
 // For example, '30 days' or '2 months'.
 // A cron job is required to remove the
 // AI summaries on a schedule.
```

```

 // See "Set up a cron job to delete AI summaries" on page 212.
 'timeout' => 30, // Setting timeout for the request sent from p4 code review to AI vendor
 // Timeout set in seconds
 'ai_vendors' => array(
 'ai_model1' => array(
 'ai_vendor' => 'openAI', // name of the AI provider being used
 // See "AI for code analysis" on page 547 for how to configure other
 // AI vendors
 'ai_package_id' => '1', // Ensure this remains as '1'. Do not modify the ai_package_
 // id.
 'ai_package_key' => 'openaiwithexplaincodeongpt4', // This is intended for future
 // use when we
 // will support multiple models and have an
 // AI configuration page in the UI.
 'ai_package_value' => 'Open AI With Explain Code On GPT4', // This is used to
 // display the model
 // type in the summary of the AI
 // vendor's response.
 'ai_model' => 'gpt-4',
 'ai_package_type' => "Explain the following code:", // This is the prompt for the AI to
 // follow.
 // You can modify the prompt for purposes other
 // than explaining the code, or to request
 // output in a specific language.
 'api_key' => $SECRET_KEY, // Paste your copied OpenAI API key here
 'ai_min_char_limit' => 4, // Minimum number of characters required in the content to
 // be submitted
 // to the AI vendor for analysis. Defaults to 4 characters.
 'ai_max_char_limit' => 31000, // The maximum number of characters that can be
 // submitted to the
 // AI vendor for analysis. Defaults to 31000 characters.
),
),
),
 'archives' => array(
 'max_input_size' => 512 * 1024 * 1024, // 512M (in bytes)
 'archive_timeout' => 1800, // 30 minutes
 'compression_level' => 1, // 0-9
 'cache_lifetime' => 60 * 60 * 24, // 1 day
),
 'avatars' => array(
 'http_url' => 'http://www.gravatar.com/avatar/{hash}?s={size}&d={default}',
 'https_url' => 'https://secure.gravatar.com/avatar/{hash}?s={size}&d={default}',
),
 'comments' => array(
 'notification_delay_time' => 1800, // Default to 30 minutes 1800 seconds
 'threading' => array(
 'max_depth' => 4, // default depth 4, to disable comment threading set to 0
),
),

```

```

),
'depot_storage' => array(
 'base_path' => '//depot_name',
),
'diffs' => array(
 'max_diffs' => 1500, // maximum number of lines displayed in a diff
 // From 2025.2 onwards, this is only available in
 // the classic view of the review page.
 'max_total_diff_size' => 200000, // maximum size of the total diffs of a file in bytes,
defaults to 0.2 MB
 // From 2025.2 onwards, this is only available in
 // the new view of the review page.
),
'environment' => array(
 'mode' => 'production',
 'hostname' => 'myswarm.hostname',
 'external_url' => null,
 'base_url' => null,
 'logout_url' => null, // defaults to null
 'vendor' => array(
 'emoji_path' => 'vendor/gemoji/images',
),
),
'files' => array(
 'max_size' => 1000 * 1024, //1MB (in bytes),
 'download_timeout' => 1800,
 'allow_edits' => true, // default is true
),
'groups' => array(
 'super_only' => true, // default value is false
),
'http_client_options' => array(
 'timeout' => 10, // default value is 10 seconds
 'sslcapath' => "", // path to the SSL certificate directory
 'sslcert' => "", // the path to a PEM-encoded SSL certificate
 'sslpassphrase' => "", // the passphrase for the SSL certificate file
 'hosts' => array(), // optional, per-host overrides. Host as key, array options as
values
),
'jira' => array(
 'host' => "", // URL for your installed Jira web interface (start with https:// or http://)
 'api_host' => "", // URL for Jira API access, 'host' is used for Jira API access if 'api_
host' is not set
 'user' => "", // Jira Cloud: the username used to connect to your Atlassian account
 // Jira on-premises: the username required for Jira API access
 // Jira on-premises with Personal Access Tokens (PATs): the username must be
blank
 'password' => "", // Jira Cloud: a special API token, obtained from

```

```
https://id.atlassian.com/manage/api-tokens
 // Jira on-premises: the password required for Jira API access
 // Jira on-premises with Personal Access Tokens (PATs): the Personal Access Token
(PAT) obtained from Jira On-premises.
 'job_field' => "", // optional, if P4DTG is replicating Jira issue IDs to a job field, list that
field here
 'link_to_jobs' => false, // set to true to enable Perforce job links in Jira, P4DTG, and
job_field required
 'delay_job_links' => 60, // delay in seconds, defaults to 60 seconds
 'relationship' => "", // Jira subsection name links are added to defaults to empty, links
added to the "links to" subsection
 'ignored_users' => array(), // Reviews or changes by users specified here are not linked
to any Jira issue.
),
'linkify' => array(
 'word_length_limit' => 2048, // limit on the number of characters which a text to be
linkified can have
 'target' => '_self', // opens the URL in the same tab or a new tab, defaults to '_self'.
To open the URL in a new tab, set to '_blank'
 'markdown' => array(
 array(
 'id' => 'jobs',
 'regex' => "", // the regular expression used to match the job keyword, default is
empty
 'url' => "", // url that matching job numbers are appended to, default is empty
),
),
),
'log' => array(
 'priority' => 3, // 7 for max, defaults to 3
 'reference_id' => false // defaults to false
),
'mail' => array(
 // 'recipients' => array('user@my.domain'),
 'notify_self' => false,
 'transport' => array(
 'host' => 'my.mx.host',
),
),
'markdown' => array(
 'markdown' => 'safe', // default is 'safe' 'disabled' 'safe' 'unsafe'
),
'mentions' => array(
 'mode' => 'global',
 'user_exclude_list' => array('super', 'swarm-admin'),
 'group_exclude_list' => array('testers', 'writers'), // defaults to empty
),
'menu_helpers' => array(
```



```

 'MyMenu01' => array(// A short recognizable name for the menu item
 'id' => 'custom01', // A unique id for the menu item. If not included in the
array, parent array name is used.
 'enabled' => true, // When set to true, the menu item is visible. Defaults to
true if not included in the array.
 'target' => '/module/MyMenuItem/', // The URL or custom module route a menu click
takes you to.
 // If not included in array, id is used. If id not included, parent array
name is used.
 'cssClass' => 'custom_menu', // The custom CSS class name added to the menu
item, appended to h2.menu- in P4 Code Review CSS
 'title' => 'MyMenuItem', // The text that will be shown on the button.
 // If not included in array, id is used. If id not included, parent array
name is used.
 'class' => "", // If not included in array or empty, the menu item is added to
the main menu.
 // To add the menu item to the project menu for all of the projects,
set to '\Projects\Menu\Helper\ProjectContextMenuHelper'
 'priority' => 155, // The position the menu item is displayed at in the menu.
 // If not included in the array, the menu item is placed at the bottom
of the menu.
 'roles' => null, // null|'authenticated'|'admin'|'super'
 // If not included in the array, null is the default.
 // Specifies the minimum level of Perforce user that can see the
menu item.
 // 'authenticated' = any authorized user, null = unauthenticated
users
),
),
'notifications' => array(
 'honor_p4_reviews' => false,
 'opt_in_review_path' => '//depot/swarm',
 'disable_change_emails' => false,
),
'p4' => array(
 'port' => 'my-helix-core-server:1666',
 'user' => 'admin_userid',
 'password' => 'admin user ticket or password',
 'sso' => 'disabled', // 'disabled'|'optional'|'enabled'
 // default value is 'disabled'
 'proxy_mode' => true, // defaults to true
 'slow_command_logging' => array(
 3,
 10 => array('print', 'shelve', 'submit', 'sync', 'unshelve'),
),
 'max_changelist_files' => 1000,
 'auto_register_url' => true,
),
'projects' => array(

```

```

 'mainlines' => array(
 'stable', 'release', // 'main', 'mainline', 'master', and 'trunk' are hardcoded, there is no
 need to add them to the array
),
 'add_admin_only' => false,
 'add_groups_only' => array(),
 'edit_name_admin_only' => false,
 'edit_branches_admin_only' => false,
 'permission_check' => false,
 'readme_mode' => 'enabled',
 'fetch' => array('maximum' => 0), // defaults to 0 (disabled)
 'allow_view_settings' => true, // defaults to true
),
'queue' => array(
 'workers' => 3, // defaults to 3
 'worker_lifetime' => 595, // defaults to 10 minutes (less 5 seconds)
 'worker_task_timeout' => 1800, // defaults to 30 minutes
 'worker_memory_limit' => '1G', // defaults to 1 gigabyte
),
'redis' => array(
 'options' => array(
 'password' => null, // Defaults to null
 'namespace' => 'Swarm',
 'server' => array(
 'host' => 'localhost', // Defaults to 'localhost' or enter your Redis server hostname
 'port' => '7379', // Defaults to '7379' or enter your Redis server port
),
),
),
'items_batch_size' => 100000,
'check_integrity' => '03:00', // Defaults to '03:00' Use one of the following options:
 // 'HH:ii' (24 hour format with leading zeros), the time the integrity check
starts each day
 // positive integer, the time between integrity checks in seconds. '0' =
integrity check disabled
 'population_lock_timeout' => 300, // Timeout for initial cache population. Defaults to 300
seconds.
),
'reviews' => array(
 'patterns' => array(
 'octothorpe' => array(// #review or #review-1234 with surrounding whitespace/eol
 'regex' => '/(?:P<pre>(?:\s|^)\(?)(?P<keyword>review|append|replace)(?:-(?P<id>[0-9]+))?(?P<post>[.,!?:;])*(?=\s|$))/i',
 'spec' => '%pre%#%keyword%-%id%%post%',
 'insert' => "%description%\n\n#review-%id%",
 'strip' => '/^\s*#(review|append|replace)(-[0-9]+)?(\s+|$)(\s+|^)\s*#(review|append|replace)(-[0-9]+)?\s*$/i',
),
 'leading-square' => array(// [review] or [review-1234] at start
 'regex' => '/^(?P<pre>\s*)(?P<keyword>review|append|replace)(?:-(?P<id>[0-9]+))?(?P<post>[.,!?:;])*(?=\s|$)/i',
),
),
),

```

```

9])?)\](?P<post>\s*)/i',
 'spec' => '%pre%[%keyword%-%id%]%post%',
),
 'trailing-square' => array(// [review] or [review-1234] at end
 'regex' => '/(?P<pre>\s*)\[(?P<keyword>review|append|replace)(?:-(?P<id>[0-9]+))?)\](?P<post>\s*)?$/i',
 'spec' => '%pre%[%keyword%-%id%]%post%',
),
),
'filters' => array(
 'filter-max' => 15,
 'result_sorting' => true,
 'date_field' => 'updated', // 'created' displays and sorts by created date, 'updated'
displays and sorts by last updated
),
'cleanup' => array(
 'mode' => 'user', // auto - follow default, user - present checkbox(with default)
 'default' => false, // clean up pending changelists on commit
 'reopenFiles' => false, // re-open any opened files into the default changelist
),
'statistics' => array(
 'complexity' => array(
 'calculation' => 'default', // 'default|disabled'
 'high' => 300,
 'low' => 30
),
),
),
'allow_author_change' => true,
'allow_author_obliterate' => false,
'commit_credit_author' => true,
'commit_timeout' => 1800, // 30 minutes
'disable_approve_when_tasks_open' => false,
'disable_commit' => true,
'disable_self_approve' => false,
'end_states' => array('archived', 'rejected', 'approved:commit'),
'expand_all_file_limit' => 10,
'expand_group_reviewers' => false,
'ignored_users' => array(),
'max_secondary_navigation_items' => 6, // defaults to 6
'max_files' => 0, // defaults to 0
'auto_resolve' => false // defaults to false
'moderator_approval' => 'any', // 'any|each'
'more_context_lines' => 10, // defaults to 10 lines
'process_shelf_delete_when' => array(),
'sync_descriptions' => true,
'unapprove_modified' => true,
),
'search' => array(
 'maxlocktime' => 5000, // 5 seconds, in milliseconds

```

```

 'p4_search_host' => "", // optional URL to Helix Search Tool
),
 'security' => array(
 'disable_system_info' => false,
 'email_restricted_changes' => false,
 'emulate_ip_protections' => false, // defaults to false
 'https_port' => null,
 'https_strict' => false,
 'https_strict_redirect' => true,
 'x_frame_options' => 'SAMEORIGIN', // X-Frame-Options header to send - set false
 to disable
 'x_xss_protection' => '1; mode=block', // X-XSS-Protection header to send - set 0 to
 disable
 'x_content_type_options' => 'nosniff', // X-Content-Type-Options header to send
 'referrer_policy' => 'strict-origin-when-cross-origin', // Referrer-Policy header to send
 'strict_transport_security_policy' => 'max-age=15724800', // Strict-Transport-Security
 header to send
 'subresource_integrity' => "", // Subresource-Integrity header to send - pattern: script-src
 `hashValue`
 'content_security_policy' => "", // Content-Security-Policy header to send
 'permissions_policy' => "", // Permissions-Policy header to send - pattern: geolocation=
 (self 'https://ex.com'), camera=(), microphone=()
 'forwarded_address' => false // X-Forwarded-For header used for systems that are
 running load balancers or proxies in front of Helix P4 Code Review
 'auto_create_user' => "", // By default, no value is set, which means no auto-creation
 of users will happen.
 // To allow auto-creation of users, set to true.
 'require_login' => true,
 'prevent_login' => array(
 'service_user1',
 'service_user2',
),
),
 'session' => array(
 'cookie_lifetime' => 0, // lifetime in seconds, default value is 0=expire when
 browser closed
 'remembered_cookie_lifetime' => 60*60*24*30, // lifetime in seconds, default value is 30
 days
 'cookie_samesite' => 'lax' // SameSite attribute to prevent the web browser from
 sending the cookie in cross-site requests
 // default is 'lax' 'strict'|'lax'|'none'
 'user_login_status_cache' => 10, // Set in seconds, default value is 10 seconds.
 // Set to 0 to disable the cache and make P4 Code Review
 // check the user login status for every call to Helix Server.
 'gc_maxlifetime' => 60*60*24*30, // lifetime in seconds, default value is 30 days
 'gc_divisor' => 100, // 100 user requests
),
 'short_links' => array(
 'hostname' => 'myho.st',

```

```

 'external_url' => 'https://myho.st:port/sub-folder',
),
 'slack' => array(
 'token' => 'TOKEN',
 'project_channels' => array(
 'myproject' => array('myproject-channel'), // The P4 Code Review
project ID must be in lower case letters.
// For project 'myproject' the slack notification
// will go into the Slack channel 'myproject-
channel'.
),
 'summary_file_names' => false, // Attaches the file to the original notification
message sent to a Slack channel.
 'bypass_restricted_changelist' => false, // Allows P4 Code Review to post notification
messages to a Slack channel when a change is committed
// or a review is created for a restricted changelist, default value is
false.
 'summary_file_limit' => 10, // Limits the number of files shown in the original
notification message sent to a Slack channel, default value is 10.
 'user' => array(
 'enabled' => true, // Forces the Swarm app to use the custom username,
overrides the P4 Code Review app details.
 'name' => 'Helix Swarm', // This is the username shown in the Slack
channel when a notification is posted.
 'icon' => 'URL', // This is the avatar icon shown in the Slack channel
// when a notification is posted, overrides the avatar set in the P4
Code Review app.
 'force_user_header' => false, // The Slack notification shows the username and
avatar only for the first post by a user, default value is false.
),
),
 'tag_processor' => array(
 'tags' => array(
 'wip' => '/(^|\s)+#wip($|\s)+/i'
),
),
 'test_definitions' => array(
 'project_and_branch_separator' => ':',
),
 'translator' => array(
 'detect_locale' => true,
 'locale' => array("en_GB", "en_US"),
 'translation_file_patterns' => array(),
 'non_utf8_encodings' => array('sjis', 'euc-jp', 'windows-1252'),
 'utf8_convert' => true,
),
 'upgrade' => array(
 'status_refresh_interval' => 10, // Refresh page every 10 seconds
 'batch_size' => 1000, // Fetch 1000 reviews to lower memory usage
)
);

```

```

),
'users' => array(
 'dashboard_refresh_interval' => 300000, // Default 300000 milliseconds
 'display_fullname' => true,
 'settings' => array(
 'review_preferences' => array(
 'show_comments_in_files' => true,
 'view_diffs_side_by_side' => true,
 'show_space_and_new_line_characters' => false,
 'ignore_whitespace' => false,
),
 'time_preferences' => array(
 'display' => 'Timeago', // Default to 'Timeago' but can be set to 'Timestamp'
),
),
),
'workflow' => array(
 'enabled' => true, // Switches the workflow feature on. Default is true
),
'xhprof' => array(
 'slow_time' => 3,
 'ignored_routes' => array('download', 'imagick', 'libreoffice', 'worker'),
),
'saml' => array(...), // This configurable will be removed from P4 Code Review in a later
release.
);

```

**Important:** The P4 Code Review configuration file does not include PHP's standard closing tag (`?>`).

This is intentional as it prevents unintentional whitespace from being introduced into P4 Code Review's output, which would interfere with P4 Code Review's ability to control HTTP headers. Debugging problems that result from unintentional whitespace can be challenging, since the resulting behavior and error messages often appear to be misleading.

## Diff configuration

P4 Code Review's [Diffs](#) feature can be customized via the following config item.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## max\_diffs

This is the maximum number of lines in a diff that are displayed in a review. If the number of lines in a diff is greater than the configured maximum, the amount of lines displayed is truncated by default. An option is provided to display the whole diff for a file.

```
'diffs' => array(
 'max_diffs' => 1500,
),
```

**Important:** The max\_diffs configuration only works on the classic view of the Review page. On preview view, the Review page displays the whole diff, regardless of the max\_diffs configuration.

To limit the number of diffs lines in the preview view, use the max\_size configuration. For more information, see ["max\\_size" on page 613](#).

## max\_total\_diff\_size

If the maximum size of the total diffs of a file exceeds the value set for max\_total\_diff\_size, the first unread file will remain closed. By default, max\_total\_diff\_size is set to 0.5 MB.

```
'diffs' => array(
 'max_total_diff_size' => 200000, // maximum size of the total diffs of a file in bytes, defaults
 to 0.2 MB
),
```

**Important:** From P4 Code Review 2025.2 and onwards, the max\_total\_diff\_size configuration only works on the new view of the Review page.

## max\_size\_chunk

This is the maximum size allowed for a diff block in a file. If the size of a diff block is greater than 500 KB then the file is truncated by default. An option is provided to view all the diffs.

```
'diffs' => array(
 'max_size_chunk' => 1024 * 512, //0.5 MB is the maximum size of the diff block and the
 value must be specified in bytes
),
```

## Email configuration

P4 Code Review's email delivery is controlled by the mail configuration block in the [SWARM\\_ROOT/data/config.php](#) file.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

Example:

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 // 'sender' => 'swarm@my.domain', // defaults to 'notifications@hostname'
 'transport' => array(
 'name' => 'localhost', // name of SMTP host
 'host' => '127.0.0.1', // host/IP of SMTP host
 'port' => 587, // SMTP host listening port
 'connection_class' => 'plain', // 'smtp', 'plain', 'login', 'crammd5'
 'connection_config' => array(// include when auth required to send
 'username' => 'user', // user on SMTP host
 'password' => 'pass', // password for user on SMTP host
 'ssl' => 'tls', // empty, 'tls', or 'ssl'
),
 // override email deliveries and store all messages in this path
 // for more information about available options when using the path parameter, see
 // "Transport" on the facing page section
 // 'path' => '/var/spool/swarm',
),

 // override regular recipients; send email only to these addresses
 // 'recipients' => array(
 // 'user1@my.domain',
 // 'user2@my.domain',
 //),

 // send notifications of comments to comment authors?
 'notify_self' => false,

 // blind carbon-copy recipients
 // 'use_bcc' => true,

 // suppress reply-to header
 // 'use_replyto' => false,

 // change the email subject prefix, the default prefix is '[Swarm]'
 // 'subject_prefix' => '[Swarm]',

 // Control email thread indexing
```



```
// 'index-conversations' => true, // ['true'|'false'] default is true, email thread indexing is
turned on

// Filter emails by required or optional reviewer using the To/Cc line of an email.
// 'use_cc_on_review_emails' => true // ['true'|'false'] default is true
// for more information on the parameter behavior, see "Filter review related emails" on
page 601 section
),
```

**Note:**

Without any mail configuration in the [SWARM\\_ROOT/data/config.php](#) file, P4 Code Review attempts to send email according to PHP's configuration, found in the `php.ini` file. By default, the configuration in `php.ini` relies on SendMail being installed.

**Important:**

Email delivery for events related to restricted changes is disabled by default. See ["Restricted Changes" on page 695](#) for details on how to enable restricted change notifications.

## Sender

The `sender` item within the `mail` block specifies the sender email address that should be used for all notification email messages. The default value is:

```
notifications@hostname
```

`hostname` is the name of the host running P4 Code Review, or when specified with the ["Environment" on page 602](#) configuration.

## Transport

The `transport` block within the `mail` block defines which mail server P4 Code Review should use to send email notifications. Most of the items in this block can be omitted, or included as needed.

**Important:**

When setting the `path` parameter, ensure that you do not use any other transport configuration options. See the Laminas Framework's [File Transport Options](#) and [SMTP Transport Options](#) for more information about the supported options.

**Note:**

P4 Code Review supports the Laminas component versions in the `LICENSE.txt` file, features and functions in the Laminas documentation that were introduced in later versions of Laminas will not work with P4 Code Review. The `LICENSE.txt` file is in the `readme` folder of your P4 Code Review installation.

P4 Code Review uses the `custom_path` item to direct all email messages to a directory instead of attempting delivery via SMTP. For details, see ["Save all messages to disk" on page 599](#).

## Recipients

The `recipients` item within the `mail` block allows you to specify a list of recipients that should receive email notifications, overriding the normal recipients. This is useful if you need to debug mail deliveries.

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 'recipients' => array(
 'user1@my.domain',
 'user2@my.domain',
),
),
```

Any number of recipients can be defined. If the array is empty, email notifications are delivered to the original recipients.

## notify\_self

The `notify_self` item within the `mail` block specifies whether comment authors should receive an email for their comments. The default value is `false`. When set to `true`, comment authors receive an email notification for their own comments.

When a group mailing list is enabled it overrides the group member's preference set for the `notify_self` configurable in the `SWARM_ROOT/data/config.php` file. So even when the `notify_self` configurable is set to `false` the group member will receive an email notification.

There is a caveat that if a user is a group member and the same user is added individually to the review and has set the `notify_self` configurable to `true`, the user receives two email notifications.

## Use BCC

The `use_bcc` item within the `mail` block allows you to address recipients using the Bcc email field. Setting the value to `true` causes P4 Code Review to use the `Bcc:` field in notifications instead of the `To:` field, concealing the email addresses of all recipients.

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 'use_bcc' => true,
),
```

## Use Reply-To

The `use_replyto` item within the `mail` block allows you to suppress populating the Reply-To: email field. Setting the value to `false` causes P4 Code Review to omit the `Reply-To:` field in notifications; by default, it is populated with the author's name and email address. When this field is `true`, users receiving an email notification can simply reply to the email and their response will be addressed to the author.

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 'use_replyto' => false,
),
```

## Email subject prefix

The `subject_prefix` item within the `mail` block sets the prefix for the subject line of emails sent by P4 Code Review. By default this is set to **[Swarm]**.

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 'subject_prefix' => '[Swarm]',
),
```

## Save all messages to disk

For testing purposes, you may want to send all email to disk without attempting to send it to recipients. Use the following configuration block to accomplish this:

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 'transport' => array('path' => MAIL_PATH),
),
```

`<MAIL_PATH>` should be replaced with the absolute path where email messages should be written, the mail transport will create a new file for each email message and write it to the path directory. This path must already exist and be writable by the web server user.

### Note:

Use of the `path` item causes P4 Code Review to ignore **all** other configuration within the `transport` block. This is why `path` is commented out in the main example.

## Email headers

P4 Code Review sends out notification emails that contain custom headers which email programs can use for automatic filtering. Emails also contain headers to ensure they are correctly threaded in email clients which support doing so.

All P4 Code Review emails contain the following headers, which can be used to identify which P4 Code Review server they came from:

```
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2019.1/1798019 (2019/05/09)
```

The exact values may differ according to the version of P4 Code Review you are running, and its configuration.

If a notification is applicable to one or more projects, then each project will be listed in the **X-Swarm-Project** header, which contains a list of one or more project names. Reviews may span multiple projects, so in this case a single email is sent out with each project listed.

```
X-Swarm-Project: gemini, apollo
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2019.1/1798019 (2019/05/09)
```

If one or more of the applicable projects is private, then two or more emails may be sent. In order for the existence of the private project to be hidden from non-members, any email sent to them will not contain references to the private project. Members of each private project will receive an email tailored to them which contains references to that private project. The email to the non-private projects will not contain references to any of the private projects in the **X-Swarm-Project** header.

For example, if a review spans three projects, called **Gemini**, **Apollo**, and **Ultra**, where **Ultra** is a private project, then members of projects **Gemini** and **Apollo** will receive an email with the following headers:

```
X-Swarm-Project: gemini, apollo
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2019.1/1798019 (2019/05/09)
```

Members of the **Ultra** project will receive an email with the following header:

```
X-Swarm-Project: ultra
X-Swarm-Host: swarm.perforce.com
X-Swarm-Version: SWARM/2019.1/1798019 (2019/05/09)
```

This can result in users receiving two notification emails (if they are members of **Ultra** and one of the other two projects), but privacy for the private project is preserved.

## Email thread indexing

By default, P4 Code Review indexes emails so that email conversations are threaded correctly in your email clients. If you find that your email clients are not displaying P4 Code Review email threads correctly, disable thread indexing. To disable indexing, use the following configuration block and set to false:

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 'index-conversations' => false, // ['true'|'false'] default is 'true'
),
```

## Filter review related emails

By default, P4 Code Review filters users based on their reviewer status and assigns them to either the **To:** or **Cc:** email field. This allows you to filter out only the emails that are important to you in your email client.

When `use_cc_on_review_emails` is set to `true`:

- Required reviewers are added to the **To:** field.
- Optional reviewers are added to the **Cc:** field.

If a review has no required reviewers, all reviewer email addresses are added to the **Cc:** field. This configuration option applies to all of your P4 Code Review instance.

To disable this feature and add all the reviewers to the **To:** field of an email, set `use_cc_on_review_emails` to `false`.

```
<?php
// this block should be a peer of 'p4'
'mail' => array(
 'use_cc_on_review_emails' => false // ['true'|'false'] default is true
),
```

## Emoji

By default, P4 Code Review uses a font to provide Emoji images. P4 Code Review can make use of Emoji images from the [Gemoji project](#). Gemoji provides support for more Emojis and works on more browsers and platforms than the font P4 Code Review normally uses.

### Note:

By default, Gemoji is only supplied with a small number of custom emojis. For instructions on adding more emojis to Gemoji, see the documentation for the [Gemoji project](#).

Due to licensing issues, Gemoji cannot be distributed with P4 Code Review. You can use the Gemoji images after following these steps:

1. Download the latest release (currently 3.0.0) of the [Gemoji project](#) from their [releases page](#).
2. Unpack the release archive into `SWARM_ROOT/public/vendor`.

After unpacking, you should see a new folder: `SWARM_ROOT/public/vendor/gemoji-3.0.0`.

3.0.0 represents the version of Gemoji, which may differ if you downloaded a different or newer release.

### Tip:

By default, Gemoji is stored in the `SWARM_ROOT/public/vendor/gemoji` directory.

If the `"emoji_path"` on page 605 configurable has been changed: unpack the Gemoji release archive into the directory specified in the `emoji_path` configurable. Modify the commands in the following steps to match that directory.

3. Rename the new folder, default location shown in the example below:

```
$ cd SWARM_ROOT/public/vendor
$ mv gemoji-3.0.0 gemoji
```

Replace `3.0.0` in the above command if you have downloaded a different or newer release of Gemoji.

4. Ensure that the new images are readable.

```
$ cd SWARM_ROOT/public/vendor
$ chmod -R +r gemoji
```

P4 Code Review detects and uses Gemoji images automatically.

## Environment

This section describes the *environment* configuration items available for P4 Code Review:

- **mode**: whether P4 Code Review operates in *production* or *development* mode.
- **hostname**: specifies the canonical hostname P4 Code Review should use, such as in links to P4 Code Review in email notifications.
- **external\_url**: specifies the canonical URL P4 Code Review should use, such as in links to P4 Code Review in email notifications and in test callback URLs. P4 Code Review can often auto-detect the correct URL, but use of `external_url` might be necessary in complex web hosting environments.
- **base\_url**: specifies the folder name P4 Code Review is installed within when P4 Code Review is not installed in the web server's document root.
- **logout\_url**: specifies the URL that users are redirected to when they logout of P4 Code Review.
- **emoji\_path**: specifies the location the Gemoji images are stored in.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

Add the following configuration block to the `SWARM_ROOT/data/config.php` file:

```
<?php
// this block should be a peer of 'p4'
```

```
'environment' => array(
 'mode' => 'development', // defaults to 'production'
 'hostname' => 'myswarm.hostname', // defaults to requested hostname
 'external_url' => null, // defaults to null
 'base_url' => null, // defaults to null
 'logout_url' => null, // defaults to null
 'vendor' => array(
 'emoji_path' => 'vendor/gemoji/images',
),
),
```

## mode

By default, P4 Code Review operates in *production* mode. When **mode** is set to **development**, P4 Code Review displays greater error detail in the browser. Also, P4 Code Review switches from including aggregated and minified JavaScript and CSS to requesting each JavaScript and CSS resource for all active modules. The default value is **production**. Any value other than **development** is assumed to mean **production**.

**development** mode makes it easier to discover problems and to identify their source, but also incurs additional browser overhead due to many more JavaScript and CSS requests for larger files. We recommend that you do not use development mode in production environments, unless directed to do so by Perforce technical support.

## hostname

The **hostname** item allows you to specify P4 Code Review's hostname. This could be useful if you have multiple virtual hosts deployed for a single P4 Code Review instance; P4 Code Review uses the hostname you configure when generating its web pages and email notifications.

### Note:

The value specified for the **hostname** item should be just the hostname. It should not include a scheme (e.g. "http://"), nor should it include a port (e.g. ":80").

## external\_url

The **external\_url** item allows you to specify P4 Code Review's canonical URL. This is useful if your P4 Code Review instance is proxied behind another web service, such as a load balancer, caching proxy, etc., because P4 Code Review's auto-detection of the current hostname or port could otherwise result in incorrect self-referencing URLs.

When specified, P4 Code Review uses the **external\_url** item as the prefix for any URLs it creates that link to itself in its web pages, email notifications, and test callback URLs.

**Note:**

- Any path components included in external\_url are ignored. If you specify `https://myswarm.url:8080/a/b/c`, P4 Code Review only uses `https://myswarm.url:8080/` when composing URLs.
- If you set the external\_url for P4 Code Review, disable the auto\_register\_url feature in P4 Code Review. For information about disabling the auto\_register\_url feature for P4 Code Review, see ["Client integration" on page 573](#).

**Important:****Multiple P4 Code Review instances connected to a single P4 Server instance:**

If you specify [base\\_url](#) along with external\_url, ensure that all P4 Code Review instances specify the same base\_url. Varying base\_url amongst cooperating P4 Code Review instances is not supported.

## base\_url

The `base_url` item allows you to specify P4 Code Review's folder within the web server's document root. This is useful if you cannot configure P4 Code Review to operate within its own virtual host, such as when you have an existing web service and P4 Code Review must exist alongside other applications or content. For information on configuring P4 Code Review to run within a sub-folder of an existing website, see ["Run P4 Code Review in a sub-folder of an existing web site" on page 219](#).

**Tip:**

The swarm-trigger.conf file must be updated to include the base\_url in the SWARM\_HOST path, see ["P4 Code Review trigger configuration" on page 221](#)

By default, `base_url` is null, which is equivalent to specifying `/`. If you specify a folder, include the leading `/`. For example, `/swarm`.

**Warning:****Multiple P4 Server instances on a single P4 Code Review instance:**

**Issue:** P4 Code Review will lose connection to all of the P4 Servers if you edit the [base\\_url](#) configurable value in the environment block of `<swarm_root>/data/config.php`. This will stop your system working.

**Fix:** Remove the base\_url configurable from the environment block of `<swarm_root>/data/config.php`.

**Important:****Multiple P4 Code Review instances connected to a single P4 Server instance:**

If you specify [external\\_url](#) along with base\_url, ensure that all P4 Code Review instances specify the same base\_url. Varying base\_url amongst cooperating P4 Code Review instances is not supported.



## logout\_url

The `logout_url` configurable is used to automatically redirect users to a custom URL after they have logged out of P4 Code Review.

When the `logout_url` item is set and you log out of P4 Code Review:

- Using **Log out** from P4 Code Review: you are logged out by P4 Code Review and then redirected to the custom URL.
- Using **Log out** from a single P4 Code Review instance on the global dashboard: you are logged out of the P4 Server instance by P4 Code Review. You are only redirected to the custom URL if you are not logged in to any other P4 Server instances on the global dashboard.
- Using **Log out from all instances** on the global dashboard: you are logged out of all of the P4 Server instances by P4 Code Review and then redirected to the custom URL.

To redirect users to a custom URL after they have logged out of P4 Code Review, edit the [SWARM\\_ROOT/data/config.php](#) file and set the `logout_url` configurable to the URL you want to redirect users to:

```
<?php
// this block should be a peer of 'p4'
'environment' => array(
 'logout_url' => <http[s]://your/redirect/logout/url>, // defaults to null
),
```

The default value is `null`, users are not redirected when they log out of P4 Code Review.

## emoji\_path

The `emoji_path` configurable defines the location of the Gemoji images in [SWARM\\_ROOT/public/](#).

To change the location of the Gemoji images, edit the [SWARM\\_ROOT/data/config.php](#) file and set the `emoji_path` configurable to the location the Gemoji images are stored in:

```
<?php
// this block should be a peer of 'p4'
'environment' => array(
 'vendor' => array(
 'emoji_path' => 'vendor/gemoji/images',
),
),
```

The default location is `vendor/gemoji/images`, this means that by default images are stored in [SWARM\\_ROOT/public/vendor/gemoji/images](#).

## Further information

For instructions on downloading Gemoji and configuring P4 Code Review to use the images, see [Emoji](#).

## Excluding Users from Activity Streams

Larger P4 Server installations often have one or more *service* users that perform automated tasks, such as build systems, continuous integration test servers, or integrations with 3rd-party databases via P4 DTG.

As P4 Code Review reports the activity of users, and these service users can generate significant volumes of activity entries, P4 Code Review provides a mechanism to ignore activity from specified users.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

Update the `SWARM_ROOT/data/config.php` file to include the following configuration block:

```
<?php
// this block should be a peer of 'p4'
'activity' => array(
 'ignored_users' => array(
 'p4dtguser',
 'system',
),
),
```

After `SWARM_ROOT/data/config.php` is updated, P4 Code Review no longer records activity for any of the listed userids. Any previously recorded activity is included in activity streams.

## Extensions Management

**Note:**

You must be logged in as a user with *super* user privileges to configure the P4 Server Extensions.

This section details how to manage the P4 Server Extensions. You can use the `swarm-extctl.sh` script file located in `/opt/perforce/swarm/sbin` directory to perform different actions on the P4 Server Extensions.

You can use the following commands to manage the P4 Server Extensions:

- ["List" on the facing page](#)
- ["Delete " on page 608](#)
- ["Install" on page 608](#)
- ["Config" on page 609](#)

- ["Tokens" on page 611](#)
- ["Disable" on page 611](#)
- ["Enable" on page 612](#)
- ["Save" on page 612](#)
- ["Load" on page 612](#)
- ["Ping" on page 613](#)

## List

### Description

Lists all the installed Extensions and their configurations.

### Commands run

Runs the `list` command.

`swarm-extctl list`

The `swarm-extctl.sh` script responds with:

```
... config swarm
... extension Perforce::helix-swarm
... uuid 4532BC59-7BC8-478F-ADF6-0A563C42563D
... revision 1
... owner super
... type change-commit
... arg //...
... debugStatus none

... config swarm
... extension Perforce::helix-swarm
... uuid 4532BC59-7BC8-478F-ADF6-0A563C42563D
... revision 1
... owner super
... type change-content
... arg //...
... debugStatus none

... config swarm
... extension Perforce::helix-swarm
... uuid 4532BC59-7BC8-478F-ADF6-0A563C42563D
... revision 1
... owner super
... type change-submit
... arg //...
... debugStatus none
```

## Delete

### Description

Deletes the P4 Server extension and their configurations.

### Commands run

Runs the delete command.

```
swarm-extctl delete
```

The swarm-extctl.sh script responds with:

```
Checking to see if you are a super user... super
Deleting the configuration
Are you really sure (y/n)?
y
Extension 'Perforce::helix-swarm and its configurations' successfully deleted.
```

## Install

### Description

Installs the P4 Server extension. The helix-swarm.p4-extension file located in "[swarm\\_root](#)" on [page 712](#)/p4-bin/extensions directory is used to install Extensions.

For more information about installing P4 Server extension, see "[Installing the P4 Code Review extension for P4 Server \(recommended\)](#)" on [page 182](#).

After installation, the saved configurations are automatically applied to the extension. For more information about the saved configurations, see "[Save](#)" on [page 612](#) command.

### Commands run

Runs the install command.

```
swarm-extctl install
```

The swarm-extctl.sh script responds with:

```
Checking to see if you are a super user... super
Installing the configuration
Are you really sure (y/n)?
y
Extension 'Perforce::helix-swarm' version '2023.1/2414875' installed successfully.
Perform the following steps to turn on the Extension:

Create a global configuration if one doesn't already exist.
p4 extension --configure Perforce::helix-swarm
```

```
Create an instance configuration to enable the Extension.
p4 extension --configure Perforce::helix-swarm --name swarm
```

For more information, visit:

<https://www.perforce.com/manuals/v23.1/extensions/Content/Extensions/Home-extensions.html>

Reading configuration from /opt/perforce/etc/swarm-extension.conf

Extension config helix-swarm saved.

Reading configuration from /opt/perforce/etc/swarm-extension.conf.inst

Extension config swarm saved.

## Config

### Description

Lists the global configuration and instance configuration for the installed Extensions.

The global configuration is retrieved from /opt/perforce/etc/swarm-extension.conf file and the instance configuration is retrieved from /opt/perforce/etc/swarm-extension.conf.inst file. If no configuration files are available, the extension specification file opens in your text editor. You can edit the specification file and save it in your text editor.

### Commands run

Runs the config command.

```
swarm-extctl config
```

The swarm-extctl.sh script responds with:

The global configuration and instance configuration opened in your text editor.  
Here is a sample global configuration specification for P4 Server extension:

```
A Perforce Extension Specification.
#
ExtName: The name of the Extension being configured.
ExtDescription: The description of the Extension being configured.
ExtVersion: The version of the Extension being configured.
ExtUUID: The UUID/key of the Extension being configured.
ExtRev: The revision of the Extension being configured.
ExtMaxScriptTime: Maximum seconds the Extension may be run.
ExtMaxScriptMem: Maximum megabytes the Extension may use.
ExtAllowedGroups: Groups whose members may configure the Extension.
ExtEnabled: Enable/Disable this Extension.
ExtP4USER: Perforce user account for the Extension to use.
Name: The name of this Extension config.
```

# Owner: The user who created this Extension config.  
# Update: Update time for the Extension config spec.  
# Description: The description of this Extension config.  
# ExtConfig: Extension-supplied configuration fields.  
#  
# See 'p4 help extension' for detailed information.

ExtName: helix-swarm

ExtDescription:  
Helix Swarm Extension

ExtVersion: 2023.1/2414875

ExtUUID: 4532BC59-7BC8-478F-ADF6-0A563C42563D

ExtRev: 1

ExtMaxScriptTime: unset

ExtMaxScriptMem: unset

ExtAllowedGroups:

"/tmp/tmp.28273.140308554466304.89" 63 lines, 1847 characters

ExtEnabled: true

ExtP4USER: super

Name: helix-swarm

Owner: super

Update: 2023/03/07 15:35:18

Description:  
This is the Helix Swarm extension, which replaces the old Perl trigger that was used previously. If this is installed and configured, then the Swarm triggers MUST be removed.  
This file configures the connection to the Swarm server. Only one Swarm instance can manage a single Helix server.

ExtConfig:  
Swarm-URL: http://localhost/

Swarm-Token: 785D3DD6-304D-5FCA-AF47-751578849960

```
Swarm-Secure: true
```

```
Debug: 3
```

```
E163: There is only one file to edit
```

After you edit the specification file and save it in your text editor.

```
Extension config helix-swarm saved.
```

```
Extension config swarm saved.
```

## Tokens

### Description

Lists the trigger tokens available for a P4 Server including the multiple P4 Server (P4D) instances. The tokens are saved in `/opt/perforce/swarm/data/queue/tokens` directory. The Swarm installation folder is `/opt/perforce/swarm/`.

### Commands run

Runs the `tokens` command.

```
swarm-extctl tokens
```

The `swarm-extctl.sh` script responds with:

```
Checking to see if you are a super user... super
```

```
Available tokens:
```

```
785D3DD6-304D-5FCA-AF47-751578849960
```

## Disable

### Description

Disables the P4 Server extension configuration.

### Commands run

Runs the `disable` command.

```
swarm-extctl disable
```

The `swarm-extctl.sh` script responds with:

```
Checking to see if you are a super user... super
```

```
Disabled
```

## Enable

### Description

Enables the P4 Server extension configuration.

### Commands run

Runs the `enable` command.

```
swarm-extctl enable
```

The `swarm-extctl.sh` script responds with:

```
Checking to see if you are a super user... super
Enabled
```

## Save

### Description

Saves the global configuration to `/opt/perforce/etc/swarm-extension.conf` file and the instance configuration to `/opt/perforce/etc/swarm-extension.conf.inst` file.

### Commands run

Runs the `save` command.

```
swarm-extctl save
```

The `swarm-extctl.sh` script responds with:

```
Checking to see if you are a super user... super
Extension config helix-swarm saved.
Extension config swarm saved.
```

## Load

### Description

Loads the global configuration from `/opt/perforce/etc/swarm-extension.conf` file and the instance configuration from `/opt/perforce/etc/swarm-extension.conf.inst` file.

### Commands run

Runs the **P4 Code Review** load command.

```
swarm-extctl load
```

The `swarm-extctl.sh` script responds with:



```
Checking to see if you are a super user... super
Extension config helix-swarm saved.
Extension config swarm saved.
```

## Ping

### Description

Lists the version and status of the P4 Server extension.

### Commands run

Runs the ping command.

```
swarm-extctl ping
```

The `swarm-extctl.sh` script responds with:

```
Checking to see if you are a super user... super
OK
Swarm Version: SWARM/2023.1/2414875 (2023/03/07)
Extension Version: 2023.1/2414875
```

## Files configuration

P4 Code Review can be customized using the following config items:

- `"max_size"` below
- `"download_timeout"` on page 615
- `"allow_edits"` on page 615

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

### max\_size

P4 Code Review limits the size of files displayed in both reviews and standard file views. Add or deleted files larger than 100 KB are paginated. The pagination limit cannot be changed.

**Important:** The truncation pagination feature was added to P4 Code Review 2025.3. This feature changes how the `max_size` configuration behaves.

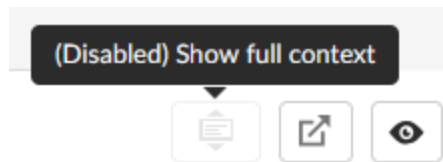
For P4 Code Review 2025.2 and earlier, see [File configuration](#).

How the files are paginated depends on the type of review being displayed.

- When viewing added or deleted files, the entire file is treated as a single diff, with large files paginated according to the 100 KB limit. The `max_size` does not affect pagination.
- When viewing edited files, `max_size` is still relevant but works differently alongside pagination. This is because diffs can be small even though they are in a large file.
  - **Diff size over limit:** If a diff exceeds 500 lines, it is paginated. The 500 line limit cannot be changed. The `max_size` configuration is not used for truncating diffs.
  - **Full file size over limit:** The `max_size` configuration is used to limit the display of edited files with large file sizes. If the file is larger than the `max_size` value:
    - The buttons to show more lines are disabled.



- The **Show full context** button is disabled.



- When using the API, the `fetchContent` response flag is set to false.
- **Within file size limit:** If the file size is within the limit defined by `max_size`, these buttons are enabled and `fetchContent` is true.

**Important:** The API response flag `isCut` is now obsolete.

## Known limitations with pagination

### Truncated diffs may hide unchanged lines and additional diffs

When viewing the changes of a large file that has been truncated, if a diff with more than 500 lines is followed by fewer than 10 unchanged lines along with another diff, only the first (truncated) diff is shown initially. The 10 unchanged lines and the second diff are hidden until the first diff is fully expanded using the **Show next** button.

**Workaround:** Reviewers should fully expand any truncated diffs to ensure all subsequent changes and unchanged lines are visible.

### Comment links may fail to navigate to truncated diff sections

Clicking a comment from the **Comments** tab does not navigate you to its location in the **Files** tab if the comment was made in a section of the diff that has been truncated. The comment remains hidden until the diff is manually expanded.

**Workaround:** If you do not see the in-line comment, click **Show next** to expand the truncated diff manually in the **Files** tab to view the comment in context.

### Complexity column displays incorrect line count for large added or deleted files

When a review includes added or deleted files with a large number of lines, the Complexity column on the Review page may display an incorrect line count. Instead of showing the full number of lines in the file (for example, 10,000), it reflects only the number of lines in the first truncated diff chunk (this is usually 3,800). This issue does not affect edited files.

#### In summary:

When viewing a large file that has been added or deleted, the file contents are paginated with a fixed 100 KB system limit.

When viewing an edited file:

- The file diffs are controlled by a fixed 500 line limit.
- The file size is controlled by the configurable `max_size` value.

Below is an example of the `max_size` configurable.

```
<?php
// this block should be a peer of 'p4'
'files' => array(
 'max_size' => 1000 * 1024, //1MB (in bytes)
),
```

## download\_timeout

When downloading files, there is a timeout which defaults to 30 minutes. This can be changed by setting the `download_timeout` configurable to a value in seconds. Alternatively, setting it to zero removes the timeout.

```
<?php
// this block should be a peer of 'p4'
'files' => array(
 'download_timeout' => 1800, // seconds (30 minutes)
),
```

## allow\_edits

By default, you can edit a file from the files page and shelve or commit it.

When set to false, files cannot be edited from the P4 Code Review files page.

```
<?php
// this block should be a peer of 'p4'
'files' => array(
 'allow_edits' => false, // default is true
),
```

## Groups configuration

By default, P4 Code Review allows all users to see the **Groups** menu item and use it to view the list of user groups. Users can view the members of a group, as well as their activities.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

You configure the visibility of this menu item with the `groups` configuration block in the [SWARM\\_ROOT/data/config.php](#) file, as in the following example:

```
<?php
// this block should be a peer of 'p4'
'groups' => array(
 'super_only' => true, // default value is false
),
```

Setting the `super_only` option to `true` hides the groups tab from non-super users.

**Tip:**

If you need more fine grained control over the visibility of the **Groups** menu item, create a `groups` block in the [menu\\_helpers](#) configuration block. This enables you to restrict the visibility to only authenticated users, only users with admin privileges, or only users with super privileges. It is important to note that visibility of the **Groups** menu is controlled by a combination of the `super_only` setting and the role setting for groups in the `menu_helper` configuration block. P4 Code Review uses the most restrictive of these settings to control the visibility of the **Groups** menu item.

## High Availability

P4 Code Review offers an active-passive form of High Availability, where the active node is a P4 Code Review instance and the passive node is a duplicate of this P4 Code Review instance that is left in standby.

Both of these P4 Code Review instances must be connected to the same P4 Server to ensure a seamless failover transition if the active node fails.

Most of your P4 Code Review data is backed up on the P4 Server. However, certain files will have to be backed up locally. These include modifications made to config, module, templates, and CSS files. If your P4 Code Review instance fails, during the failover the majority of your data is retrieved from the P4 Server and used to get the passive P4 Code Review instance up and running. You will then need to add in the modified files you have backed up locally.

## Set up active-passive in P4 Code Review

1. Set up a Redis cluster (or an equivalent standalone system). This is so P4 Code Review can connect to a system which stores the cache.
2. Set up a P4 Code Review instance to be the active node (referred to as node A) and complete the following steps:
  - a. Connect this P4 Code Review instance to the Redis cluster (or standalone system) that stores the cache.
  - b. Configure the `config.php` file with any custom modification.
  - c. If you have any Extensions installed on the P4 Server, point them towards the node A (the current active node).
3. Set up a P4 Code Review instance to be the passive node (referred to as node B) and complete the following steps:
  - a. For the passive node, you do not need to run the `configure-swarm.sh` script. For more information, see ["Why the passive node does not require the setup script" on the next page](#).
  - b. To prevent jobs from being duplicated or missed, turn off workers on node B.
  - c. Ensure the P4 Code Review data directory is shared via NFS or another shared filesystem. The data directory contains files such as `config.php`, tokens, logs, and other attachments.
  - d. Copy the queue token from the active node to the passive node. You can find the queue token in the following location:  
`/<SWARM>/data/queue/tokens/<TOKENID>`
  - e. (Optional) If you have custom modules in your active node (node A), copy them over to the passive node (node B).
  - f. (Optional) If you have public files in your active node, copy these over to the passive node.

## Pre-disaster preparations

Complete the following tasks to ensure a successful failover if a P4 Code Review instance fails.

1. Local backups
  - a. Regularly back up any local modifications, such as configuration, module, templates, and CSS files, to a backup location of your choice.
  - b. Ensure that your backup process includes checks for both the current version and historical changes of these modified files.
2. Training and documentation:
  - a. Keep detailed documentation of where your local files are stored and the steps required to retrieve them.

- b. Ensure your operations team is well-trained on the manual failover process and is aware of how to retrieve the local backups and implement the files on the passive node when it becomes active.
3. Health Checks and monitoring:
  - a. Implement automated monitoring for both the active and passive nodes.
  - b. Create alerts that inform the operations team about potential issues. This is so they can prepare for a manual failover.
  - c. If you are using HTTPS servers, make sure all certificates are trusted by the P4 Servers.
4. P4 Server High Availability setup:
  - a. Confirm the P4 Server that contains your review data is configured for High Availability.
  - b. Regularly test the High Availability failover procedures according to the [P4 Server High Availability documentation](#).

## Volume sharing between P4 Code Review instances

As an optional step, you can configure both the active and passive P4 Code Review instances to share the same data volume found in `/<SWARM>/data/`. This means the node B could easily take over should node A fail.

However, there is an edge case where some of the data may be lost during the failover period. For example, if node A failed at 14:00 and node B becomes active at 14:10, any new activity on the server may be missed in the 10 minute period.

## Failover from active node to passive node

For more in depth steps on how to move from one P4 Code Review instance to another, see "[Moving your P4 Code Review instance](#)" on page 278.

As a brief overview, to manually move activity from the active node to the passive node, follow these steps:

1. Turn on workers in node B.
2. Change the DNS record so it points to node B instead of node A. This is assuming that the correct extensions, triggers and tokens have been set up on node B.
3. Create a new P4 Code Review instance to be the passive node (referred to as node C).

The new active node is node B and requires a new passive node, which will be node C. Follow Step 3 in the [Set up active-passive in P4 Code Review](#) section above to create the passive node C.

## Why the passive node does not require the setup script

The `configure-swarm.sh` script is run when creating a P4 Code Review environment from scratch. It performs many setup tasks such as creating P4 Code Review users in P4, installing server-side extensions and modifying the `config.php` file.

After you have set up the active node, the `configure-swarm.sh` script has already configured the P4 Code Review environment for the passive node. This means the script does not need to be run again, as it would recreate users and re-edit the `config.php` file.

## Ignored users for reviews

Automated test environments may inadvertently participate in code reviews if they copy user-generated change descriptions. For example, if an automated system copied a change description containing `#review` and subsequently shelved or committed files, a new review would be started. Similarly copying a description with `#review-123` could inadvertently update an existing review. As test environments may involve thousands or millions of tests, such interactions can potentially generate far too many P4 Code Review notifications.

To mitigate this problem, P4 Code Review can be configured to ignore specified users for the purposes of starting or updating a review. Edit the `SWARM_ROOT/data/config.php` file, and provide the list of users to ignore in the `ignored_users` item in the reviews configuration block.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

Edit the `SWARM_ROOT/data/config.php` file, and provide the list of users to ignore in the `ignored_users` item in the reviews configuration block. For example:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'ignored_users' => array('build_user1', 'build_user2'),
),
```

## Job keywords

By default, if you add `jobnnnnn` or `@jobnnnnn` to a job description, changelist description, review description, or comment, P4 Code Review automatically creates a link to the P4 Code Review job page if a match is found. The `jobnnnnn` or `@jobnnnnn` keywords are hardcoded in P4 Code Review and always present.

If your job numbers are not formatted as `jobnnnnn`, you can configure P4 Code Review to recognize other keywords in addition to `job` and `@job`. For example, if your job numbers are formatted as `defectnnnnnn`, P4 Code Review can be configured to recognize them and link to the correct job page.

When you create a new job keyword, this is in addition to the default job keywords. Multiple job keywords can be specified. The first successful match is used, starting with the default job keywords. Once P4 Code Review finds a match, none of the other keywords are evaluated.

The job keyword is configured with a regular expression so that almost any keyword can be used. When you add a new job keyword to P4 Code Review, take care to choose syntax and terminology that is unlikely to occur naturally in a description to avoid unexpected links being created.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## Adding a new job keyword

**Tip:**

- You can test the results of your regular expression before using it, see [regular expressions 101](#)
- Regular expressions are case insensitive.

**Job parameters:**

- `'word_length_limit'`: The limit on the number of characters which a text to be linkified can have.
- `'target'`: Opens the URL in the same tab or a new tab, defaults to `'_self'`. To open the URL in a new tab, set to `'_blank'`.
- `'regex'`: The regular expression used to match the job keyword.
- `'url'`: The URL to append the regex match to. This URL plus the job number matched by the regex is used to display the job page. This URL can reference a [P4 Code Review page](#) or an [external defect tracking system](#).

**Optional:** Specify the regex capture group to append to the URL. For example, adding `{2}` after the url parameter tells P4 Code Review to append the second regex capture group result to the url. For more information on capture groups, see [Regex Tutorial - Parentheses for Grouping and Capturing](#).

## P4 Code Review page link example

**Tip:**

In this example, only the second regex capture group is appended to the url parameter. This means that the `@` character is only used to help identify the job and is not appended to the URL.

For example, to add "defect" as a jobs keyword so that both `@defectnnnnn` and `defectnnnnn` mentions are linked to a job page in P4 Code Review, add the following block to the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
// this block should be a peer of 'p4'
```



```
'linkify' => array(
 'word_length_limit' => 2048,
 'target' => '_blank',
 'markdown' => array(
 array(
 'id' => 'jobs',
 'regex' => '(@?)(defect[0-9]+)', // the regular expression used to match the job
keyword, default is empty
 'url' => '{baseurl}/{2}', // url that matching job numbers are appended to, default is
empty
),
),
),
```

## External page link example

### Tip:

In this example, the `{0}` character string at the end of the url is important because it helps the external link to get built dynamically.

For example, to add "issue" as a jobs keyword so that mentions of `issuennnnn` are linked to jobs in your defect tracking system, add the following block to the `SWARM_ROOT/data/config.php` file:

```
<?php
// this block should be a peer of 'p4'
'linkify' => array(
 'word_length_limit' => 2048,
 'target' => '_blank',
 'markdown' => array(
 array(
 'id' => 'jobs',
 'regex' => '(issue[0-9]+)', // the regular expression used to match the job
keyword, default is empty
 'url' => 'example.defect-tracker-url/{0}', // url that matching job numbers are
appended to, default is empty
),
),
),
```

## Jira issue link example

### Tip:

In this example, the `{0}` character string at the end of the url is important because it helps the external link to get built dynamically.

For example, to add your Jira issue format so that mentions of `xxx-nnnnn` are linked to Jira issues in your Jira system, add the following block to the `SWARM_ROOT/data/config.php` file:

```

<?php
// this block should be a peer of 'p4'
'linkify' => array(
 'word_length_limit' => 2048,
 'target' => '_blank',
 'markdown' => array(
 array(
 'id' => 'jira',
 'regex' => '[A-Z1-9]{2,}-\d+', // the regular expression used to match the job
keyword, default is empty
 'url' => 'example.jira-system-url/browse/{0}', // url that matching numbers are
 appended to, default is empty
),
),
),
),
),

```

## License

P4 Code Review is free to use with any P4 Server. You do not need to purchase a P4 Code Review license.

P4 Code Review can be used with an unlicensed P4 Server. An unlicensed P4 Server provides unlimited use for up to 5 users and 20 workspaces. When the user or workspace limit is crossed, P4 Server imposes a maximum of 1,000 files. P4 Code Review honors the restrictions of P4 Server. For more information about an unlicensed P4 Server, see [Limits for unlicensed use](#).

## Localization & UTF8 character display

### Important:

To ensure that all characters, including Unicode characters, are displayed and handled correctly by P4 Code Review, configure your P4 Server in Unicode mode. For information on configuring your P4 Server in Unicode mode, see [Set up and manage Unicode installations](#) in the [P4 Server Administration Documentation](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

P4 Code Review supports localization and the display of some non-UTF8 characters.

## Localization

### Important:

If you are using a web browser that uses the en-US locale, P4 Code Review will automatically redirect to en to get locale files.

P4 Code Review is fully localized; with an appropriate language pack installation, P4 Code Review can support users in multiple languages.

A language pack consists of `gettext-style default.po` and `default.mo` files, placed in a folder named for the locale they represent, within the language folder in the P4 Code Review root directory. In addition, language packs contain two JavaScript files to provide translation strings for the in-browser UI, `locale.js` and `locale.jsgz`, which both appear in the `SWARM_ROOT/public/build/language` folder.

The following example illustrates the directory layout of a language pack:

```
SWARM_ROOT/
 language/
 locale/
 default.mo
 default.po
 public/
 build/
 language/
 locale.js
 locale.jsgz
```

You can configure certain localization behaviors with the `translator` configuration block in the `SWARM_ROOT/data/config.php` file. Here is an example:

```
<?php
// this block should be a peer of 'p4'
'translator' => array(
 'detect_locale' => true,
 'locale' => "",
 'translation_file_patterns' => array(),
),
```

The `detect_locale` key determines whether P4 Code Review attempts to detect the browser's locale. The default value is `true`. Set the value to `false` to disable browser locale detection.

The `locale` key is a string specifying the default locale for P4 Code Review. Alternately, an array of 2 strings can be used to specify the default locale, and a fallback locale. For example:

```
<?php
// this block should be a peer of 'p4'
'translator' => array(
 'locale' => array("en_GB", "en_US"),
),
```

The `translation_file_patterns` key allows you to customize the Laminas translation infrastructure, which you might do if you are developing your own language pack. For details, see [Laminas\l18n](#).

**Note:**

P4 Code Review supports the Laminas component versions in the LICENSE.txt file, features and functions in the Laminas documentation that were introduced in later versions of Laminas will not work with P4 Code Review. The LICENSE.txt file is in the readme folder of your P4 Code Review installation.

## Non-UTF8 character display

P4 Code Review supports the display of non-UTF8 characters in file content when all of the following are true:

- The P4 Server is Unicode enabled.
- The character encoding is supported by P4 Code Review.
- The character encoding is listed in the `translator` array.

Multiple character encodings can be specified in the array, if P4 Code Review fails to find a matching encoding in the array it will fall back to windows-1252. Unsupported and invalid character encodings in the array are ignored.

**Tip:**

For information on character encodings supported by P4 Code Review, see [Supported Character Encodings](#).

Configure character encodings by adding the following block to the `SWARM_ROOT/data/config.php` file. For example:

```
<?php
// this block should be a peer of 'p4'
'translator' => array(
 'non_utf8_encodings' => array('sjis', 'euc-jp', 'windows-1252'),
),
```

**Important:**

windows-1252 is the default character encoding, it must always be the last entry in the array.

## UTF8 conversion

If P4 Code Review is connected to a non-unicode enabled P4 Server, P4 Code Review converts special characters to UTF8 by default. For ANSI and ASCII files, this can result in some special characters being displayed in the diff view incorrectly or not at all. To stop P4 Code Review converting special characters to UTF8 and to display special characters as unknown ( ), disable UTF8 conversion by setting `utf8_convert` to false.

Disable UTF8 character conversion by adding the following block to the `SWARM_ROOT/data/config.php` file. For example:

```
<?php
// this block should be a peer of 'p4'
'translator' => array(
 'utf8_convert' => false,
),
```

By default, the value of this configurable is `true`.

## Logging

P4 Code Review is a web application, so there are several types of logging involved during the course of P4 Code Review's normal operations.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## Web server logging

All accesses to P4 Code Review may be logged by the web server hosting P4 Code Review. As web server log configuration is web server specific, refer to your web server's documentation. Since we recommend the use of Apache, more information regarding log configuration in Apache can be found here: [Apache HTTP server project](#).

## P4 Server logs

P4 Code Review communicates with the associated P4 Server for every page request. Review the P4 Code Review-generated requests on your P4 Server by enabling logging.

For more information, see [Logging](#) and [Audit log file](#) in the [P4 Server Administration Documentation](#).

## P4 Code Review logs

Depending on the log configuration you provide to P4 Code Review, P4 Code Review can log its own operations. P4 Code Review's logs are much more helpful if you encounter problems.

P4 Code Review stores its log data in the `SWARM_ROOT/data/log` file. Each request in the P4 Code Review logs contains the following:

- URI (Uniform Resource Identifier)
- Process ID (PID) from the system
- IP address of the system that created the request

The worker event requests contain additional details such as:

- Worker ID
- Task information

The volume of entries recorded in the log depends on the configuration stored in the `SWARM_ROOT/data/config.php` file. Here is an example:

```
<?php
// this block should be a peer of 'p4'
'log' => array(
 'priority' => 3, // 7 for max, defaults to 3
),
```

The log `priority` specifies the importance of the messages to be logged. When `priority` is set to `0` (the lowest value), only the most important messages are logged. When `priority` is set to `7` (the highest value), all messages, including the least important messages, are logged. The default priority is `3`.

The different priority levels are:

Priority	Description
0	Emergency: a message about a system instability
1	Alert: a message about a required immediate action
2	Critical: a message about a critical condition
3	Error: a message about an error
4	Warning: a message about a warning condition
5	Notice: a message about a normal but significant condition
6	Info: an informational message
7	Debug: a debugging message

**Note:**

Setting priority to a value lower than `0` does not result in reduced logging.

## Logrotate

From P4 Code Review 2021.1, P4 Code Review package installations and upgrades install logrotate to manage your P4 Code Review log rotation. The logrotate configuration supports P4 Code Review installations with single and multiple P4 Server instances.

Logrotate is on by default and is configured to:

- Rotate your P4 Code Review logs daily.
- Keep P4 Code Review logs for 14 days.

The helix-swarm logrotate configuration file can be modified or disabled and is located in `/etc/logrotate.d`. Any changes you make will be preserved when you upgrade P4 Code Review in the future.

## Reference ID

When `reference_id` is set to `true`, every log message is appended with `referenceId:<id>` where `<id>` is a hash value based on the web request id that generated the log message. The `<id>` is appended to all of the log messages related to that web request for the life of the web request allowing you to follow the request in the log. This can help to diagnose which request was being processed when the log message was created. It is particularly useful when you have lots of requests updating the log because it helps you to see which request each log message relates to.

**Example:** when a P4 Code Review web request starts a worker the `<id>` that is generated is attached to all of the log entries related to that worker for the lifetime of the worker. This enables you to see all of the events the worker processes in its lifetime. For this example the web request reference id is `"066ae44103ced2ab05c28578a36b6854"`:

```
2019-02-27T09:07:21+00:00 INFO (6): Worker 1 startup.
{"referenceId":"066ae44103ced2ab05c28578a36b6854"}
2019-02-27T09:07:21+00:00 INFO (6): Worker 1 event: task.commit(3276)
{"referenceId":"066ae44103ced2ab05c28578a36b6854"}
2019-02-27T09:07:21+00:00 DEBUG (7): P4 (0000000055794826000000002e2ec483) start
command: help {"referenceId":"066ae44103ced2ab05c28578a36b6854"}
.
.
.
2019-02-27T09:17:16+00:00 INFO (6): Worker 1 shutdown.
{"referenceId":"066ae44103ced2ab05c28578a36b6854"}
```

To append the `referenceId:<id>` to P4 Code Review log messages, edit the [SWARM\\_ROOT/data/config.php](#) file to include the `reference_id` configurable and set it to `true`:

```
<?php
// this block should be a peer of 'p4'
'log' => array(
 'reference_id' => true, // defaults to false
),
```

The default value is `false`.

## Trigger Token Errors

If the trigger tokens are missing or invalid, the web server error log contains a suitable error:

queue/add attempted with invalid/missing token: *token used*

A token is *invalid* when it is not formed from the characters A through Z (in upper or lowercase), 0 through 9, or a dash (-).

A token is *missing* when a file using the token as its name does not exist in the [SWARM\\_ROOT](#)/data/queue/tokens directory.

## Performance logging

P4 Code Review logs warnings whenever commands issued to the P4 Server take longer than expected to complete. These warnings can be used to diagnose performance problems in the P4 Server.

### Note:

By default, warnings are not captured in the P4 Code Review log. To capture warnings, the [log priority](#) must be set to 4 or higher.

The default configuration is:

```
<?php
// this block should be placed within the 'p4' block
'slow_command_logging' => array(
 3, // commands without a specific rule have a 3-second limit
 10 => array('print', 'shelve', 'submit', 'sync', 'unshelve'),
),
```

In this configuration block, the numeric key specifies the time threshold in seconds, and the value (if present) is an array of P4 Server commands that should use the threshold. Should a command be associated with multiple thresholds, the largest is used for that command.

In addition, P4 Code Review automatically detects when the PHP extension **xhprof** is installed and collects profiling data for requests that take longer than expected. The profiling data could be helpful in diagnosing performance issues within P4 Code Review itself, particularly when analyzed in combination with the slow command logging described above. When collected, profiling data is stored in the [SWARM\\_ROOT](#)/data/xhprof folder.

The default configuration is:

```
<?php
// this block should be a peer of 'p4'
'xhprof' => array(
 'slow_time' => 3, // the threshold in seconds
 'ignored_routes' => array('download', 'imagick', 'libreoffice', 'worker'),
),
```



`slow_time` specifies the threshold, in seconds, that should be used to determine when a P4 Code Review request is slow. `ignored_routes` is an array that specifies a list of Laminas Framework `route` names that should not be profiled. For example, P4 Code Review's `Files` module specifies that the download route should be ignored from profiling as downloads could take significantly longer than the default threshold.

**Note:**

Depending on the performance of the server hosting P4 Code Review, and particularly the performance of the associated P4 Server, you may want to monitor the [SWARM\\_ROOT/data/xhprof](#) folder for disk usage. Each request that exceeds the time threshold causes 200-600KB of data to be written.

## Mainline branch identification

When viewing a project's files, the initial view is the list of the project's branches. The branches appear in alphabetical order, but the branch identified as the main branch appears first and in bold. By default, a project overview page is displayed when there is a README Markdown file in the project mainline, this can be disabled by your P4 Code Review administrator, see ["Project readme" on page 660](#).

P4 Code Review uses a list of names to identify which of a project branches should be considered as the main branch and which Markdown file is used for the project overview page. The `mainlines` array supports regular expressions, see ["Regular expressions" on the next page](#). The default names are `main`, `mainline`, `master`, and `trunk`. For the steps P4 Code Review goes through to find the mainline branch and the Markdown file for a project, see ["Project mainline and README check" on page 631](#) below.

**Tip:**

The default `main`, `mainline`, `master`, and `trunk` names are hardcoded in P4 Code Review, there is no need to add them to your array. P4 Code Review will always try to find exact matches to them from your branch names.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

You can adjust the configuration to match your local branch naming convention . Here is an example of the configuration block:

```
<?php
// this block should be a peer of 'p4'
'projects' => array(
 'mainlines' => array(
 'stable', 'release', // 'main', 'mainline', 'master', and 'trunk' are hardcoded, there is no
 need to add them to the array
```

),  
)

## Regular expressions

The `mainlines` array supports any valid PHP regular expression, this can be useful if you have a large number of similar branch names across different projects that you are using as your main branches. Regular expressions must be used with care to avoid accidentally including a branch that is not a mainline branch.

### Note:

P4 Code Review adds a `^` character before your regular expression and a `$` character after it. For example: `^your-reg-ex-pattern$` You must take this into account when constructing your regular expressions.

### Tip:

- You can test the results of your regular expressions before using them, see [regular expressions 101](#).
- Regular expressions are case insensitive.

## Regular expression examples

Useful regular expressions are shown below:

### Exact match only:

`'mymainbranch'` branches named `mymainbranch` are treated as mainline branches.

### Only match default `mainlines` array branch names:

`"` (an empty `mainlines` array) only branches that exactly match the default names of `'main'`, `'mainline'`, `'master'`, or `'trunk'` are treated as mainline branches.

### Beginning with:

`'stable.*'` branch names beginning with `stable` are treated as mainline branches.

**For example:** the following are treated as mainline branches.

`stablecode-01`, `stable2019_2`, and `stable_branchxyz`

### Ending with:

`'.*mastprj'` branch names ending with `mastprj` are treated as mainline branches.

**For example:** the following are treated as mainline branches.

`productxy-mastprj`, `assets_mastprj`, and `project07mastprj`

### Ending with digits

'.\*[0-9]+' branch names ending with digits are treated as mainline branches.

**For example:** the following are treated as mainline branches.

*branch03, product-10, stable22, main\_123456*

### Containing:

'.\*main.\*' branch names that contain main are treated as mainline branches.

**For example:** the following are treated as mainline branches.

*swarm-main-branch, branchmain\_22, main05, and assetmain-brnch*

## Project mainline and README check

### Tip:

P4 Code Review checks the branches of a project in alphabetical order.

P4 Code Review uses the following process to find the mainline branch and README Markdown file for a project:

1. P4 Code Review searches the project branches in alphabetical order for a branch that matches a name in the mainlines array. The search includes the default names.
2. When P4 Code Review finds a matching branch, that branch is used as the mainline when displaying the branches on the project page.
  - If the branch contains a README Markdown file, that file is used for the project overview page.
  - If the branch does not contain a README Markdown file, see step 3.
3. P4 Code Review checks the remaining branches for a branch that matches a name in the mainlines array.
  - If P4 Code Review finds a README file, that file is used for the project overview page and swarm stops checking the branches.
  - If a README file is not found, this step is repeated until a README file is found in a branch that matches a name in the mainlines array or all of the project branches have been checked.
  - If no README Markdown page is found, the project overview page is not displayed.

## Markdown

The markdown configurable defines what can be rendered in Markdown by P4 Code Review for project overview pages and files stored in the P4 Server. By default, markdown is set to safe, Markdown text is displayed, but Markdown support is limited to prevent execution of raw HTML and JavaScript content.

**Tip:**

- Valid Markdown file extensions are: md, markdown, mdown, mkdn, mkd, mdwn, mdtxt, mdtext.
- By default, project overview pages are displayed when there is a README Markdown file in the project mainline. This can be disabled by your P4 Code Review administrator, see ["Project readme" on page 660](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

Add or update the following configuration block to the [SWARM\\_ROOT/data/config.php](#) file, at the same level as the p4 entry:

```
<?php
// this block should be a peer of 'p4'
'markdown' => array(
 'markdown' => 'safe', // default is 'safe'
),
```

- safe:** Markdown content is displayed, but Markdown support is limited to prevent execution of raw HTML and JavaScript content. This is the default.
- unsafe:** Markdown support is unrestricted, allowing full HTML and JavaScript to be used. This is insecure as any person with access to P4 Code Review can add script to the Markdown which would execute as the currently logged in user.
- disabled:** Markdown text is not rendered and is only displayed as plain text. This is the most secure setting.

**Note:**

Markdown content is displayed in comments and review descriptions, but Markdown support is limited to prevent execution of raw HTML and JavaScript content. This is the equivalent of safe mode and cannot be changed.

## Menu helpers

The `menu_helpers` configuration block is used to control the visibility of P4 Code Review menu items and to create new menu items for URLs and P4 Code Review modules.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## Add a menu item to a menu

Add P4 Code Review menu items in the menu\_helpers configuration block in the [SWARM\\_ROOT/data/config.php](#) file, as in the following example:

```
<?php
// this block should be a peer of 'p4'
'menu_helpers' => array(
 'MyMenu01' => array(// A short recognizable name for the menu item
 'id' => 'custom01', // A unique id for the menu item. If not included in the array,
parent array name is used.
 'enabled' => true, // When set to true, the menu item is visible. Defaults to true
if not included in the array.
 'target' => '<https://MyWebite>', // The URL or custom module route a menu click
takes you to.
// If not included in array, id is used. If id not included, parent array
name is used.
 'cssClass' => 'custom_menu', // The custom CSS class name added to the menu
item.
 'title' => 'MyMenuitem', // The text that will be shown on the button.
// If not included in array, id is used. If id not included, parent array
name is used.
 'class' => "", // If not included in array or empty, the menu item is added to the
main menu.
// To add the menu item to the project menu for all of the projects, set
to '\Projects\Menu\Helper\ProjectContextMenuHelper'
 'priority' => 155, // The position the menu item is displayed at in the menu.
// If not included in the array, the menu item is placed at the bottom
of the menu.
 'roles' => null, // null|'authenticated'|'admin'|'super'
// If not included in the array, null is the default
// Specifies the minimum level of Perforce user that can see the
menu item.
// 'authenticated' = any authorized user, null = unauthenticated users
),
),
```

### Configurables

#### Tip:

You do not need to enter values for all of the configurables in a menu item array, configurables not included in the array will be set to their default values. For an example of this, see ["Create a menu item using only the custom module name" on page 636](#).

- `id` is a unique id used by P4 Code Review for the menu item, this can be anything. If `id` is not included in the array, P4 Code Review uses the name of the parent array.
- `enabled`
  - `true` the menu item is visible subject to restrictions imposed by the `roles` value. This is the default value if `enabled` is not included in the array.
  - `false` the menu item is not in use
- `target` the URL or [P4 Code Review module](#) route used when the menu item is clicked. If `target` is not included in the array, P4 Code Review uses the `id` for the route. If `id` is not included in the array, P4 Code Review uses the parent array name for the route.
- `cssClass` specifies the CSS class for the menu item, used to replace the default icon with a custom icon for the menu item, see ["Add a custom icon to a menu item" on the facing page](#).
- `title` specifies the menu item name displayed in the menu. If `title` is not included in the array, P4 Code Review uses the `id` for the menu title. If `id` is not included in the array, P4 Code Review uses the parent array name for the menu title.
- `class` specifies whether the menu item is displayed in the main menu or the project menu.
  - `"` or not included in the array, adds the menu item to the main menu.
  - `\Projects\Menu\Helper\ProjectContextMenuHelper` adds the menu item to the project menu of all of your projects.

**Tip:**

You can restrict the menu item to a specific project by extending the `ProjectAwareMenuHelper` module, see ["Add a menu item to the project menu of a specific project" on page 636](#).

- `priority` specifies the menu item position in the menu, lower numbers are displayed higher up the menu structure. If `priority` is not included in the array, the menu item is placed at the bottom of the P4 Code Review menu. If you use the same `priority` value as an existing menu item (not recommended), the menu items with the same `priority` are displayed in alphabetical order.

**Tip:**

Enter `<yourSwarmURL>/api/v10/menus` in your browser address bar to return a list of the `priority` values for the P4 Code Review menu items.

- `roles` restrict the display of the menu item based on the permissions of the logged in user:
  - `null` menu item visible to any user even if they are not authenticated. if `roles` is empty or not included in the array, `null` is used as the default.
  - `authenticated` menu item is visible to any authenticated user
  - `admin` menu item is visible to any user with Perforce *admin* permissions or higher
  - `super` menu item is visible to any user with Perforce *super user* permissions

## Add a custom icon to a menu item

Add some custom CSS to P4 Code Review to replace the default icon with a custom icon for the menu item. For example, the CSS below replaces the default icon with the `double-speech-bubbles.svg` icon for the `custom01` menu item by modifying the `svgIcon.custom01MenuIcon` and `.menuItem-custom01` classes:

```
.swarmMenu .menuItem.menuItem-tests .svgIcon{
 display:none;
}.swarmMenu .menuItem.menuItem-tests a::before{
 content: "";
 background-image: url('/swarm/img/icons/line/double-speech-bubbles.svg');
 background-size: 16px;
 background-repeat: no-repeat;
 padding-left: 20px;
 padding-right: 4px;
 margin-left: 20px;
}
```

Save these lines in a file, perhaps `menu_custom01.css` in a folder called `custom_menus`, in the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, immediately using your custom menus for subsequent page views. If you are using your own custom images, we recommend you store them in the same folder as your custom css.

## Make Groups menu item visible to admin-user and above only

### Note:

The visibility of the P4 Code Review **Groups** menu is controlled by a combination of the role setting here and the `super_only` setting in the groups configuration block, see ["Groups configuration" on page 616](#). P4 Code Review uses the most restrictive of these two settings to control the visibility of the **Groups** menu item.

If you need fine grained control over the visibility of the **Groups** menu item, create a groups block in the `menu_helpers` configuration block and set the roles as required. In this example roles is set to `'admin'` so that only users with *admin-user* access and higher will be able to see the **Groups** menu item:

```
<?php
// this block should be a peer of 'p4'
'menu_helpers' => array(
 'groups' => array(
 'roles' => 'admin',
),
),
```

**Tip:**

Enter `<yourSwarmURL>/api/v10/menus` in your browser address bar to return a list of roles for the P4 Code Review menu items.

## Create a menu item using only the custom module name

If you have created a P4 Code Review custom module you can create a menu item for it that uses the custom module route and you do not need to add any other configurables for it in the `menu_helpers` configuration block.

For example, if you have a custom module with a route of `'codeAnalytics'`, the configuration block only needs to be:

```
<?php
// this block should be a peer of 'p4'
'menu_helpers' => array(
 'codeAnalytics' // links the menu item to a module called 'codeAnalytics'
);
```

All of the other configurable settings for the `codeAnalysis` menu item will be set to their default values because the `codeAnalysis` array is empty:

- `id`: parent array name used (`codeAnalysis`)
- `enabled` = `true`, menu item is enabled
- `target`: parent array name used (`codeAnalysis`), this must match the custom module name
- `cssClass`: no class is set and P4 Code Review uses the default chevron `>` icon for the menu item
- `class`: no class is set, the menu item is displayed in the main menu
- `priority`: no priority is set, the menu item is displayed at the bottom of the P4 Code Review menu
- `roles`: `null`, visible to all users even if they are not authenticated

## Add a menu item to the project menu of a specific project

To add a menu item in the project menu of a specific project, extend the `ProjectAwareMenuHelper` and set the menu item `class` to use the extended helper.

The example in this section only displays the **Jenkins Build** menu item in the **Jam** project menu.

### Extend the ProjectAwareMenuHelper

Extend the `ProjectAwareMenuHelper` so that the **Jenkins Build** menu item is only displayed in the **Jam** project menu. This helper can be called anything and saved anywhere but it should have a descriptive name and be in a logical place, in this example it is `/modules/Project/src/Menu/Helper/JamOnlyMenuItem.php`:



```

<?php
/**
 * Perforce Swarm
 * @copyright 2020 Perforce Software. All rights reserved.
 * @license Please see LICENSE.txt in top-level folder of this distribution.
 * @version <release>/<patch>
 */

namespace Projects\Menu\Helper;

/**
 * Only show the menu item using this class for the specified Project.
 * In this example it is for the project with the id of Jam.
 * Class JamOnlyMenuItem
 * @package Projects\Menu\Helper
 */
class JamOnlyMenuItem extends ProjectAwareMenuHelper
{
 /**
 * Modifies an item's target if the context is for a project, the project supports the item and
 * the item already has a target. Otherwise, nullify the item
 * @return array|null
 */
 public function getMenu()
 {
 if ($this->project && $this->project->getId() === 'jam') {
 $item = parent::buildMenu();
 // Allow project characteristics to determine menu item availability
 return !empty($this->project) && $this->isDisabled() === false ? $item : null;
 }
 return null;
 }
}

```

## Define your menu item

The following block defines the **Jenkins Build** menu item you are adding to the **Jam** project menu. The important part is that class is set to `\Projects\Menu\Helper\JamOnlyMenuItem` so that it is only displayed in the **Jam** project.

```

<?php
// this block should be a peer of 'p4'
'menu_helpers' => array(
 'jenkins_jam' => array(// A short recognizable name for the menu item
 'id' => 'jenkins_jam',
 'enabled' => true,
 'target' => 'http://my-jenkins.instance.jam.com',
),
)

```

```

 'cssClass' => 'custom_menu',
 'title' => 'Jenkins Build',
 'class' => '\Projects\Menu\Helper\JamOnlyMenuItem',
 'priority' => 160,
 'roles' => ['authenticated'],
),
),

```

## Delete P4 Code Review config cache to enable the menu item

Your changes will not be used by P4 Code Review until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration. You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## Multiple P4 Server instances

A single P4 Code Review server can be connected one or more P4 Server (P4D) instances. This section describes how to configure P4 Code Review to connect to multiple P4 Servers.

### Warning:


**Issue:** P4 Code Review will lose connection to all of the P4 Servers if you edit the [base\\_url](#) configurable value in the environment block of <swarm\_root>/data/config.php. This will stop your system working.

**Fix:** Remove the `base_url` configurable from the environment block of <swarm\_root>/data/config.php.

### Note:

**Known limitations, only applies if you want to use "Global Dashboard" on page 291:**

- **Issue:** If P4 AS is enabled for P4 Code Review and the **Try to login to all available servers with these credentials** checkbox or the **All available servers** option is selected in a login dialog, P4 Code Review will not try to log in to any of the other P4 Server instances that are configured for P4 AS.

**Workaround:** Log in to them individually using the instance **Log in** button  in the sidebar or by including the server instance name in the URL, for example: <https://swarm.company.com/serverA>.

- P4 Code Review must be installed in its own virtual host.

Complete the following steps to connect P4 Code Review to multiple P4 Server instances:

- ["Set up the P4 Code Review configuration file for the P4 Servers" on the facing page](#)
- ["Set the P4 Code Review trigger token and P4 Code Review host variable for each P4 Server" on page 647](#)
- ["Configure a cron job for each P4 Server instance" on page 648](#)

## Set up the P4 Code Review configuration file for the P4 Servers

The connection details between P4 Code Review and P4 Servers are configured using the `p4` item in the `<swarm_root>/data/config.php` configuration file.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

When P4 Code Review is configured for a single P4 Server instance the `p4` item looks like this:

```
<?php
return array(
 'p4' => array(
 'port' => 'my-helix-core-server:1666',
 'user' => 'admin_userid',
 'password' => 'admin user ticket or password',
 'sso' => 'disabled', // ['disabled'|'optional'|'enabled'] default value is 'disabled'
),
);
```

Where:

- port is the P4PORT for your P4 Server.
- user is a *userid* that has *admin* privileges for the P4 Server.
- password is the ticket or password for the P4 Server user.
- sso if the P4 Server is configured for the P4 AS, set to one of the following:
  - enabled all users must use P4 AS to log in to P4 Code Review.
  - optional P4 AS is available for users to log in to P4 Code Review but is not enforced.
  - disabled P4 AS is not available to P4 Code Review. This is the default value.

### Important:

From P4 Code Review 2021.1, the `sso_enabled` configurable is deprecated but remains supported. It is replaced with the more flexible `sso` configurable. If the `sso_enabled` configurable and `sso` configurable are both present in the `p4` configuration block, P4 Code Review uses the `sso` configurable value.

Multiple P4 Server instances can be added to the `p4` item in `config.php` but each P4 Server instance must have a label to identify the server instance. This enables P4 Code Review to connect to the P4 Servers in the `p4` item.

**Tip:**

The P4 Server instances do not have to all be at the same version but they must all be versions that P4 Code Review supports. For a list of P4 Server versions supported by P4 Code Review, see [P4 Server requirements](#).

The first P4 Server in the `p4` item is the primary P4 Server, this instance renders the global dashboard. If P4 Code Review fails to connect to the primary P4 Server, P4 Code Review cannot display any of the P4 Server instances in the global dashboard. You can still connect directly to the other P4D instances by including the P4 Server label in the URL for each instance, for example:

`https://swarm.company.com/serverA`.

**Configure the P4 Code Review configuration file for multiple P4 Server instances:**

Enter a label for each P4 Server instance and enter the server connection details under the server label.

For example:

```
<?php
return array(
 'p4' => array(
 'serverA' => array(
 'port' => 'p4d-A:1666',
 'user' => 'admin_userid_serverA',
 'password' => 'admin user ticket or password for serverA',
 'sso' => 'disabled', // ['disabled'|'optional'|'enabled'] default value is 'disabled'
),
 'serverB' => array(
 'port' => 'p4d-B:1666',
 'user' => 'admin_userid_serverB',
 'password' => 'admin user ticket or password for serverB',
 'sso' => 'disabled', // ['disabled'|'optional'|'enabled'] default value is 'disabled'
),
);
```

Where:

- `serverA` and `serverB` are labels that identify the individual P4 Servers.
  - The labels must be URL-friendly labels.
  - The labels must not contain any spaces or any characters that require URL encoding, and
  - The total length of the server label and [Redis](#) namespace combined is limited to  $\leq 127$  characters. The default namespace is `Swarm`.
- The `port`, `user`, `password`, and `sso` items are specific to the P4 Server instance they are nested under.

## Configure a specific P4 Server

The root element in the `<swarm_root>/data/config.php` configuration file contains default values for all P4 Server instances. You can override the default values by using a server specific configuration file. For example, to override the global value of jira configuration for a specific server, you can provide a different value for jira configuration in the server specific configuration file.

To configure a specific P4 Server instance, add a `config.php` file in the `<swarm_root>/data/servers/<serverid>` directory. For the new P4 Server configuration to take effect, the configuration cache must be deleted and reloaded. For more information on how to delete and reload cache, see ["Cache Info" on page 719](#).

### Important:

- You must include a global `config.php` file to specify the multiple P4 Server instances.
- You must ensure that a `p4` item is not included in the P4 Server specific configuration. If the `p4` item is included, no error message is displayed and the `p4` item is ignored by the P4 Server.

### Note:

P4 Server specific configuration overrides the global configuration.

## Configure event notification for each P4 Server

### Important:

You must use either P4 Server Extensions (recommended) or P4 Server Triggers to notify P4 Code Review about events on the P4 Servers. You cannot use a mixture of the two methods.

### Tip:

Triggers are still supported, but we recommend you use P4 Server Extensions. P4 Server Extensions are easier to install and maintain than Triggers.

P4 Code Review needs to know about a number of P4 Server events to operate correctly, this can be done by using P4 Server Extensions (recommended) or P4 Server Triggers. P4 Code Review installs include the P4 Server extension file and trigger scripts required for P4 Code Review to get the events it needs from your P4 Server.

You must install and configure the P4 Server extension or triggers to complete the P4 Code Review configuration.

**Do one of the following so that P4 Code Review is notified about events on the P4 Servers it is connected to:**

- Install and configure the P4 Server extension for each P4 Server instance, see ["Install and configure the P4 Server extension \(recommended\)" on the next page](#)
- Set the P4 Code Review trigger token and host variable for each P4 Server instance, see ["Set the P4 Code Review trigger token and P4 Code Review host variable for each P4 Server" on page 647](#)

## Install and configure the P4 Server extension (recommended)

### Important:

- If you are using the P4 Code Review extension to connect to the P4 Server, do not install P4 Code Review triggers.
- The following commands must be run on each P4 Server instance P4 Code Review is connected to.
- You must be a P4 Server *super* user to install and configure P4 Server Extensions.

### Prerequisites

#### To install the P4 Server extension you need:

A compatible version of P4 Server for the extension:

- **Linux:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.
- **Windows:** P4 Server 2021.2 and later. If you are using an earlier version of P4 Server, you must use triggers.

#### You will also need:

- Extensions installed and configured on your P4 Server.
- A user with *super* permissions to install and configure the P4 Server extension.

### P4 Server extension installation

The P4 Server extension file is included for all of the P4 Code Review installation types. The `helix-swarm.p4-extension` file is located in "`swarm_root`" on [page 712](#)/p4-bin/extensions

On each P4 Server you are connecting P4 Code Review to, run the following commands as a P4 Server user with *super* permissions:

1. Install the P4 Server extension on the P4 Server with:  
`p4 extension --yes --install helix-swarm.p4-extension`

#### Example response:

Extension 'Perforce::helix-swarm' version '2022.1.20221215' installed successfully.  
Perform the following steps to turn on the Extension:

# Create a global configuration if one doesn't already exist.

```
p4 extension --configure Perforce::helix-swarm
```

# Create an instance configuration to enable the Extension.

```
p4 extension --configure Perforce::helix-swarm --name Perforce::helix-swarm-
instanceName
```

For more information, visit:

<https://www.perforce.com/manuals/extensions/Content/Extensions/Home-extensions.html>

2. Create a global configuration if one doesn't already exist:

```
p4 extension --configure Perforce::helix-swarm
```

The spec file opens in your text editor, for example:

```
ExtName: helix-swarm
```

```
ExtDescription:
```

```
 Helix Swarm Extension
```

```
ExtVersion: 2022.1.DEV.20220120
```

```
ExtUUID: 4532BC59-7BC8-478F-ADF6-0A563C42563D
```

```
ExtRev: 1
```

```
ExtMaxScriptTime: unset
```

```
ExtMaxScriptMem: unset
```

```
ExtAllowedGroups:
```

```
ExtEnabled: true
```

```
ExtP4USER: sampleExtensionsUser
```

```
Name: helix-swarm
```

```
Owner: super
```

```
Update: 2022/01/21 10:43:46
```

```
Description:
```

```
 The description of your config.
```

```
ExtConfig:
 Debug:
 2
 Swarm-Secure:
 true
 Swarm-Token:
 SWARM-TOKEN
 Swarm-URL:
 http://localhost/
```

3. Edit the spec file to match your system:

- **ExtP4USER:** Set to an existing user with *super* permissions.
- **ExtConfig:** Check and edit the values in this block:
  - **Debug:**

Debug levels 0 to 3 control the amount of debug information sent to the P4 Server Extensions log.

**Important:**

A debug level of 10 and higher sends all debug information to the client. This is useful for debugging, but should not be run in a production environment.

- **SSL-CA-File:** (optional) Set the absolute path of the [Certificate Authority \(CA\) Privacy-Enhanced Mail \(PEM\)](#) file to validate the P4 Server's certificate for P4 Code Review. Ensure that this file is accessible to the P4 Server process owner.

If a suitable certificate is not found in the local SSL certificate store maintained by the operating system or if there are verification issues from using a self-signed certificate then the default CA file is used.

By default, the local certificate stored in the default CA file is used if the absolute path for the [Certificate Authority \(CA\)](#) is not specified.

- **Swarm-Secure:**
  - **true** will only accept secure SSL certificates.
  - **false** will accept insecure SSL certificates.
- **Swarm-Token:** Set to the P4 Code Review trigger token value.

**Important:**

To set multiple P4 Server token settings for Extensions, repeat the following steps for each P4 Server. The process to obtain an extension token of your P4 Code Review instance is similar to obtaining a trigger token of your P4 Code Review instance.



To obtain the trigger token of your P4 Code Review instance:

- a. Log in to P4 Code Review as a *super* user.
    - b. Click your *userid*, found at the right of the main toolbar.
    - c. Select **About P4 Code Review**.
    - d. The **About P4 Code Review** dialog is displayed and P4 Code Review generates an API token if it doesn't already exist.
    - e. Copy the trigger token value displayed at the bottom of the dialog and paste the token into the spec file.
  - **Swarm-URL**: Set to the URL of your P4 Code Review instance.
4. When you have finished editing the spec file, save it in your text editor.
  5. Create an instance configuration to enable the extension:  
`p4 extension --configure Perforce::helix-swarm --name swarm`  
The spec file opens in your text editor, for example:

ExtName: helix-swarm

ExtDescription:  
Helix Swarm Extension

ExtVersion: 2022.1.DEV.20220120

ExtUUID: 4532BC59-7BC8-478F-ADF6-0A563C42563D

ExtRev: 1

ExtMaxScriptTime: unset

ExtMaxScriptMem: unset

ExtEnabled: true

ExtDebug: none

Name: swarm

Owner: super

Update: 2022/01/21 10:58:41

Description:  
The description of your config.

ExtConfig:  
depot-path:  
//...  
enableStrict:  
true  
enableWorkflow:  
true  
httpTimeout:  
30  
ignoreErrors:  
false

We recommend that these settings are left at their default values, but the configuration needs to be run to set the default values.

- `enableStrict`: Enable strict checking of reviews for workflow. Verifies that the file content in a commit matches the file content of its associated approved review. If one or more files in a commit do not match the content of the file in its associated review, the commit is rejected.

- `enableWorkflow`: Enable workflow.
  - `httpTimeout`: Timeout (in seconds) when communicating with the P4 Code Review server.
  - `ignoreErrors`: If the server returns an error (timeout, not there), then default to allowing a request. It means workflow rules can be skipped, but if the P4 Code Review server is down it will not block submits.
6. Save the spec file in your text editor.
  7. Confirm your P4 Server extension is installed and configured by running the following command on the P4 Server:
 

```
p4 extension --run swarm ping
```

    - If the P4 Server extension is working, the P4 Server responds with `OK`
    - If the P4 Server extension is not working, the P4 Server responds with an error message. For example, `BAD (Cannot reach https://swarm-example.com/)`

For more information about P4 Server Extensions, see:

    - [Extension overview](#) in the [P4 Server Administration Documentation](#).
    - [p4 extension](#) in the [P4 CLI Reference](#).
  8. Repeat these steps for each P4 Server.
  9. When you have configured all of the P4 Servers, configure a cron job to start P4 Code Review workers for each P4 Server instance, see "[Configure a cron job for each P4 Server instance](#)" on the next page.

## Set the P4 Code Review trigger token and P4 Code Review host variable for each P4 Server

### Important:

Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.

P4 Server uses a P4 Code Review trigger token to confirm that trigger requests from P4 Code Review are valid. Each P4 Server instance must have a valid P4 Code Review trigger token in its `swarm-trigger.conf` file. The `swarm-trigger.conf` file also contains the P4 Code Review host URL for the P4 Server instance.

### Set the P4 Code Review trigger token and P4 Code Review host variable for each P4 Server instance:

1. Navigate to the P4 Code Review URL for the instance.  
For example: `https://swarm.company.com/serverA`
2. Log in to P4 Code Review as a *super* user.
3. Click your *userid*, in the main toolbar and select **About P4 Code Review**.  
The **About P4 Code Review** dialog is displayed with a **Trigger Token**. P4 Code Review generates the trigger token if it doesn't already exist.
4. Copy the trigger token value from the dialog.

5. Open the `swarm-trigger.conf` file for the P4 Server instance.

**Tip:**

For more information about the location of the `swarm-trigger.conf` file, see ["Installing triggers" on page 187](#).

6. In the `swarm-trigger.conf` file:
  - a. Set the `SWARM_HOST` URL.
  - b. Set the `SWARM_TOKEN` by pasting the P4 Code Review trigger token you copied in the steps above.

For example:

```
SWARM_HOST (required)
Hostname of your Swarm instance, with leading "http://" or "https://".
SWARM_HOST="https://swarm.company.com/serverA"

SWARM_TOKEN (required)
The token used when talking to Swarm to offer some security. To obtain the
value, log in to Swarm as a super user and select 'About Swarm' to see the
token value.
SWARM_TOKEN="TRIGGER-TOKEN-FOR-SERVERA"
```

7. Save the `swarm-trigger.conf` file.
8. Repeat these steps for each P4 Server.
9. When you have configured all of the P4 Servers, configure a cron job to start P4 Code Review workers for each P4 Server instance, see ["Configure a cron job for each P4 Server instance" below](#).

**Tip:**

Alternatively, you can simply touch a file in each `<Swarm root>/data/servers/<serverid>/tokens` folder. The file content is ignored, the filename itself is the token.

This allows you to specify a common token that is used by all of the P4 Server instances.

However, we recommend that you use a separate token for each P4 Server instance. This makes it easier to invalidate a token for a specific P4 Server by deleting the file in the tokens folder for that server if you need to.

## Configure a cron job for each P4 Server instance

To automatically spawn workers for each of the servers connected to P4 Code Review, you must manually configure a cron job for each of the servers.

### Swarm package installations only

The `swarm-cron-hosts.conf` file specifies the connection type (HTTP or HTTPS), hostname, port number and, server label for P4 Code Review cron jobs.

If you have installed P4 Code Review via packages, you must specify all of the P4 Server URLs within `/opt/perforce/etc/swarm-cron-hosts.conf`.

1. Edit the `swarm-cron-hosts.conf` file so that it contains the actual P4 Code Review hostname, ports, and server labels you have configured for P4 Code Review.

The following format is used with one P4 Server on each line:

```
[http[s]://]<swarm-host>[:<port>][/<base-url>]
```

**Default if value not specified:**

- `[http[s]://]` http
- `<swarm-host>` must be specified
- `[:<port>]` 80
- `[/<base-url>]` server label, must be specified

For example, for `serverA` and `serverB` configured earlier on a P4 Code Review host of `https://swarm.company.com` with a default port value of 80. The entries in the `swarm-cron-hosts.conf` file would be:

```
https://swarm.company.com/serverA
https://swarm.company.com/serverB
```

2. Save the `swarm-cron-hosts.conf` file.
3. P4 Code Review is now configured to connect to multiple P4 Server instances.

**Tip:**

To check or modify P4 Code Review worker configuration, see ["Worker configuration" on page 741](#).

## P4 Code Review Tarball installation only

If you have installed P4 Code Review from a tarball or configured cron manually, you need create a file called `helix-swarm` and add all of the P4 Server instances connected to P4 Code Review.

1. Create a file named `helix-swarm` in `/etc/cron.d` if it does not already exist.
2. Edit the `helix-swarm` file so that it has the following content:

```
#
Cron job to start Swarm workers every minute
#
***** nobody [-x /opt/perforce/swarm/p4-bin/scripts/swarm-cron.sh] &&
/opt/perforce/swarm/p4-bin/scripts/swarm-cron.sh
```

3. Save the `helix-swarm` file.
4. Edit the `swarm-cron-hosts.conf` file so that it contains the actual P4 Code Review hostname, ports, and server labels you have configured for P4 Code Review.

The following format is used with one P4 Server on each line:

```
[http[s]://]<swarm-host>[:<port>][/<base-url>]
```

**Default if value not specified:**

- [http[s]://] http
- <swarm-host> must be specified
- :<port> 80
- [/<base-url>] server label, must be specified

For example, for serverA and serverB configured earlier on a P4 Code Review host of *https://swarm.company.com* with a default port value of 80. The entries in the `swarm-cron-hosts.conf` file would be:

```
https://swarm.company.com/serverA
https://swarm.company.com/serverB
```

5. Save the `swarm-cron-hosts.conf` file.
6. P4 Code Review is now configured to connect to multiple P4 Server instances.

**Tip:**

To check or modify P4 Code Review worker configuration, see ["Worker configuration" on page 741](#).

## Further information

### Users

- To visit the Global Dashboard, enter the basic P4 Code Review URL without a P4 Server instance name, for example: `https://swarm.company.com`.
- To visit a specific P4 Server instance in P4 Code Review without going via the global dashboard, include the server name in the URL, for example:  
`https://swarm.company.com/serverA`.

Once you are viewing the P4 Server in P4 Code Review, P4 Code Review works as a standard single P4 ServerP4 Code Review system.

**Tip:**

- To browse jobs on **serverA**, navigate to: `https://swarm.company.com/serverA/jobs`
- To browse reviews on **serverB**, navigate to:  
`https://swarm.company.com/serverB/reviews`
- To view the dashboard for **serverB**, navigate to  
`https://swarm.company.com/serverB/#actionable-reviews`

- If you don't include the server label in the URL you will be taken to the specified page for the primary P4 Server.

For example: navigating to <https://swarm.company.com/reviews> will redirect you to <https://swarm.company.com/serverA/reviews> because **serverA** is the [Primary P4 Server](#) in the p4 configuration item.

## Administrators

On the web server hosting P4 Code Review, P4 Code Review automatically creates a data folder for each P4 Server instance in `<Swarm_root>/data/servers`. This is because each P4 Server request is a field.

For example:

```
ls -la /opt/perforce/swarm/data/servers/serverA
total 362296
drwx----- 6 apache apache 4096 Sep 8 17:15 .
drwx----- 6 apache apache 4096 Sep 6 15:41 ..
drwx----- 2 apache apache 4096 Sep 20 18:08 cache
drwx----- 3 apache apache 4096 Sep 7 13:25 clients
-rw-r--r-- 1 apache apache 3709543 Sep 26 14:19 log
-r----- 1 apache apache 84 Sep 6 15:41 p4trust
drwx----- 4 apache apache 4096 Sep 20 14:27 queue
drwx----- 2 apache apache 4096 Sep 20 11:18 sessions
```

### Tip:

It is important to understand that there will be a P4 Code Review log file for each P4 Server instance.

## Developers

To get a list of projects via the P4 Code Review API for *serverA*, run bash:

- If you are using `wget`:

```
$ wget -u "apiuser:password" https://swarm.company.com/serverA/api/v10/projects
```

- If you are using `curl`:

```
$ curl -u "apiuser:password" https://swarm.company.com/serverA/api/v10/projects
```

### Tip:

If you don't include the server label in the URL you will be taken to the projects page for the primary P4 Server.

For example: navigating to <https://swarm.company.com/api/v10/projects> will redirect you to <https://swarm.company.com/api/v10/serverA/projects> because **serverA** is the [Primary P4 Server](#) in the p4 configuration item.

## Notifications

This section describes the configurables that can be set for notifications.

### Note:

For more information on how to delay comment notifications, and configure comment threading, see ["Comments" on page 580](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

P4 Code Review can be configured to provide generic notifications of committed changes in P4 Server, taking the role of a review daemon.

The review daemon ignores protections and notifications are triggered for review paths regardless of the user rights on the paths.

Notifications configuration is expressed with a `notifications` block in the [SWARM\\_ROOT/data/config.php](#) file, similar to the following example (defaults shown):

```
<?php
// this block should be a peer of 'p4'
'notifications' => array(
 'honor_p4_reviews' => false, // defaults to false
 'opt_in_review_path' => "", // required if honor_p4_reviews is true; defaults to ""
 'disable_change_emails' => false, // optional; defaults to false
),
```

If `honor_p4_reviews` is set to `true`, then `opt_in_review_path` must be set to a path somewhere in the depot. This path does not need to point to an actual file that exists, but it must be accessible by all users who want to make use of this functionality. For example:

```
'notifications' => array(
 'honor_p4_reviews' => true,
 'opt_in_review_path' => '//depot/swarmReviews',
),
```

If these two values are set, then users can make use of the Perforce review functionality by subscribing to the `opt_in_review_path` in their user spec. Any user subscribed to that file, will receive notifications for all the other paths they are subscribed to.

We recommend that `opt_in_review_path` point to a file that does not exist. Ideally, it points to a file that no user is likely to want to create. It must however be in a valid depot.

For example, if a user has the following review paths set in their user spec:



```
$ p4 user -o asmith
```

```
User: asmith
```

```
Email: asmith@example.com
```

```
FullName: Alice Smith
```

```
Reviews:
```

```
 //depot/swarmReviews
```

```
 //depot/main/acme/...
```

```
 //depot/main/orion/...
```

```
 //depot/dev/asmith/...
```

```
 //streams/main
```

The `//depot/swarmReviews` means that this user is subscribed to the path set in `opt_in_review_path`, and therefore will receive notifications.

The subscription lines ending in `/...`, define which paths in the depot the user is interested in. For example, this user will receive a notification for a change made to `//depot/main/acme/foo.txt`, but not a change made to `//depot/dev/acme/foo.txt`.

To receive notifications for stream spec changes in a path, leave the `/...` off of the end of the subscription line. For example, the `//streams/main` subscription line means the user will receive a notification for a change made to a stream spec in `//streams/main`.

To see which users are subscribed to receive notifications you can run `p4 reviews <path>` against the `opt_in_review_path` value:

```
$ p4 reviews //depot/swarmReviews
```

```
asmith <asmith@example.com> (Alice Smith)
```

```
bbrown <bbrown@example.com> (Bob Brown)
```

```
erogers <erogers@example.com> (Eve Rogers)
```

To see which users are subscribed to files in a particular changelist, you can run `p4 reviews -c <changelist>`. Swarm will notify the users who subscribe to both the review of this changelist and the review path, `opt_in_review_path`.

- **honor\_p4\_reviews**: When set to `true`, P4 Code Review sends notification emails for every committed change to all users where the change matches one of their **Reviews**: paths.
- **opt\_in\_review\_path**: Optional item, required only if **honor\_p4\_reviews** is set. This item specifies a special depot path, which typically would not exist in the P4 Server machine. When a path is specified, users must include this path (or a path that contains it) in the **Reviews**: field of their user spec to cause P4 Code Review to send the user a notification for every committed change that matches one of their **Reviews**: paths.

For example, if the `opt_in_review_path` is set to `//depot/swarmReviews`, users can opt-in to review notifications by adding that path, or a path such as `//depot/...`, to the **Reviews**: field in their user spec.

- **disable\_change\_emails**: Optional item. When set to `true`, notifications for committed changes, based on the **Reviews**: field and the users and projects you follow, are disabled. Notifications for reviews and comments will still be sent.

**Note:**

If your P4 Server machine already has a review daemon in operation, users receive two notifications for `Reviews: paths`. You may want to deprecate the review daemon in favor of P4 Code Review's change notifications.

**Note:**

"Groups" on page 352 have per-group notification settings. See "Add a group" on page 471 for details.

## Global settings

There are many situations that can result in email notifications being sent out to users and groups. Whilst it is possible for a user and group owner to configure their own settings, it is also possible for the system owner to configure the defaults for all users and groups by modifying the settings in the `config.php`.

**Note:**

- **If a group owner is not specified for the group:** only users with *super* privileges can configure group notification settings.
- **If one or more group owners are specified for the group:** only group owners and users with *super* privileges can configure group notification settings.

Each notification consists of an `Event` and a `Role`. The `Event` is what happened (for example, a new review was created, a file was submitted) and the `Role` is the role of the user or group who could receive a notification. A user or group can belong to multiple roles, in which case if any of them are set to send a notification, then the user or group will receive a notification.

For example, when a review is voted on (`review_vote`), there are a number of roles of users, and groups that could be notified:

- **Users:** the user that voted on the review (`is_self`), the author of the review (`is_author`), a user who is a moderator of the project branch the review is in (`is_moderator`), and a user who is a reviewer of the review (`is_reviewer`).
- **Groups:** members of a moderator group for the project branch the review is in (`is_moderator`), and members of a reviewer group for the review (`is_reviewer`).

## Configuration options

By default, all notifications are enabled for all roles. The system-wide defaults can be changed by adding the following options into the notifications block of the `SWARM_ROOT/data/config.php`. These options are in addition to those described above.

```
<?php
// this block should be a peer of 'p4'
'notifications' => array(
 'review_new' => array(
 'is_author' => 'Enabled',
 'is_member' => 'Enabled',
```

```
),
'review_files' => array(
 'is_self' => 'Enabled',
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_vote' => array(
 'is_self' => 'Enabled',
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_required_vote' => array(
 'is_self' => 'Enabled',
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_optional_vote' => array(
 'is_self' => 'Enabled',
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_state' => array(
 'is_self' => 'Disabled',
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_tests' => array(
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_changelist_commit' => array(
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_member' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_comment_new' => array(
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
),
'review_comment_update' => array(
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
```

```
),
'review_comment_liked' => array(
 'is_commenter' => 'Enabled',
),
'review_opened_issue' => array(
 'is_self' => 'Enabled',
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
'review_join_leave' => array (
 'is_self' => 'Enabled',
 'is_author' => 'Enabled',
 'is_reviewer' => 'Enabled',
 'is_moderator' => 'Enabled',
),
)
```

Each setting can have one of four possible values to either enable or disable notifications of that type. If multiple settings apply to a given event, then a user will receive a notification if *any* of them are enabled.

- **Enabled:** Notifications for this event and role are enabled, and a user will receive emails by default.
- **Disabled:** Notifications for this event and role are disabled, and a user will not receive emails by default.
- **ForcedEnabled:** Same as for Enabled, but this is forced enabled for all users. An individual user is not able to disable this notification type on their settings page.
- **ForcedDisabled:** As for Disabled, but this is forced disabled for all users. An individual user will not be able to enable this notification type on their settings page.

Unless one of the forced options is used, system wide options can be overridden by individual users and group owners, who can configure which notifications they receive.

## Notification Roles

The various roles are as follows:

- **is\_self:** This role is the user who changed the state of the review.
- **is\_author:** This role is the user who was the author of the review.
- **is\_reviewer:** This role includes all users and groups who are listed as being a reviewer on the review.
- **is\_member:** This role includes all users who are a member of the project in which the event happened.
- **is\_moderator:** This role includes all users and groups who are listed as being a moderator of the project branch in which the event happened.

- **is\_follower:** This role includes all users who are followers of the project in which the event happened.
- **is\_commenter:** This role is the user who was the author of a comment.

## Notification events

An event is the action that causes the notification:

- **review\_new:** A new review has been created.
- **review\_files:** Files have been added to a review.
- **review\_vote:** A user has voted on a review.
- **review\_state:** The status of a review has been changed.
- **review\_tests:** Automated tests have failed for a review. The first time automated tests pass for a review after a test failure for that review.
- **review\_changelist\_commit:** Files on a review have been committed.
- **review\_comment\_new:** A comment has been added to a review.
- **review\_comment\_update:** A comment on a review has been updated.
- **review\_comment\_liked:** A user has liked a comment.
- **review\_join\_leave:** A user or group has joined or left a review.

## Obliterate Review

### Note:

- By default, you must be a user with *admin* or *super* user rights to obliterate a review.
- **Optional:** P4 Code Review can be configured to allow users to obliterate reviews that they have authored. This can be configured by your P4 Code Review administrator. See ["Allow author obliterate review" on page 675](#).

Obliterate is used to permanently delete reviews that have been created by mistake. For instance, if a review is associated with the wrong changelist, or a review contains sensitive information that should not be openly available.

## When you obliterate a review

- **All reviews:**
  - Metadata associated with the review is permanently deleted.
  - Events associated with the review are permanently deleted from the activity feed.
  - The P4 Code Review 404 page is displayed if a user navigates to the review from a P4 Code Review notification email link, or by using the URL.


- **Review with a shelved changelist:**
  - The review changelist, and any associated shelved files are permanently deleted.
  - Files shelved in the user's associated changelist are left intact.
- **Review with a committed changelist:**
  - The review changelist is permanently deleted.
  - The committed changelist is left intact.

## Obliterate a review

### Important:

Obliterate must be used with care, the review and all of its associated metadata are permanently deleted. An obliterated review cannot be reinstated, not even by Perforce Support.

### To obliterate a review:

1. Navigate to the review.
2. Click the **Review actions**  button and select **Obliterate Review**.
3. Click **Yes** on the confirmation dialog to complete the obliterate action.
4. The review is obliterated.

## P4TRUST

When P4 Code Review is configured to connect to a P4 Server (P4D) using an SSL connection, P4 Code Review automatically executes the `p4 trust` command, which accepts the SSL fingerprint and creates a `p4trust` file containing a list of trusted servers and their fingerprints.

The location the `p4trust` file is saved to depends on whether P4 Code Review is connected to a single P4 Server or to multiple P4 Servers.

- **Single P4 Server:** saved as `SWARM_ROOT/data/p4trust`
- **Multiple P4 Servers:** saved as a separate file for each server. For example, for `serverA`, `serverB`, and `serverC` they are saved as:
  - `SWARM_ROOT/data/serverA/p4trust`
  - `SWARM_ROOT/data/serverB/p4trust`
  - `SWARM_ROOT/data/serverC/p4trust`

## If a certificate changes

If a certificate for a P4 Server is changed for any reason then P4 Code Review connections to that server will fail after that server is restarted.

The solution is to delete the `p4trust` file for that P4 Server from the location described above. P4 Code Review will automatically run `p4 trust` on the next request if the `p4trust` file is not found.

## Projects

This section describes the configurables that can be set for projects.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## Restrict project name and branch definition editing to administrators

By default, once a project has been created, any member of the project can edit or delete the project's settings.

Instead of allowing any changes, or preventing all changes, you may want to prevent project members from making specific changes, such as to the project's name (and associated identifier), or adjusting the branch definition(s). This is useful when build infrastructure or other tooling treats these details as operational configuration, but you still want members to be able to adjust other aspects of the project configuration.

To do so, edit the `SWARM_ROOT/data/config.php` file, and set the following two items, similar to the following example:

```
'projects' => array(
 'edit_name_admin_only' => true,
 'edit_branches_admin_only' => true,
),
```

- `edit_name_admin_only`: when set to true, only users with *admin* privileges in the P4 Server can modify a project's name.
- `edit_branches_admin_only`: when set to true, only users with *admin* privileges in the P4 Server can modify a project's branch definition(s).

Both items default to `false`.

## Limit adding projects to administrators

By default, any authenticated user can add new projects. P4 Code Review can restrict project creation to users with *admin*-level privileges or higher. Once restricted, P4 Code Review prevents non-administrators from adding projects, and does not display the + icon to add a project to non-administrators.

Add or update the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
```

```
'projects' => array(
 'add_admin_only' => true,
),
```

**Important:**

If `add_admin_only` is enabled and `add_groups_only` has one or more groups configured, project creation is only available to users with administrator privileges and who are members of the specified groups.

## Limit adding projects to members of specific groups

P4 Code Review can restrict project creation to members of specific groups. The groups and membership need to be defined in the P4 Server.

Add or update the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
'projects' => array(
 'add_groups_only' => array('wizards', 'slayers', 'phbs'),
),
```

**Important:**

If `add_admin_only` is also enabled, project creation is only available to users with administrator privileges **and** who are members of the specified groups.

## Project readme

By default, projects can have a `README` markdown file associated with them that is automatically displayed on the project overview page. This file is read from the root of the project's mainline if it is available, and displayed on the project page. See "[Mainline branch identification](#)" on [page 629](#) for details on configuring the project mainline.

**Tip:**

- Valid Markdown file extensions are: `md`, `markdown`, `mdown`, `mkdn`, `mkd`, `mdwn`, `mdtxt`, `mdtext`.

If more than one `README` file is found, P4 Code Review displays the first one it finds based on the order above.

- By default, Markdown support is limited to prevent execution of raw HTML and JavaScript content. The level of Markdown support can be configured by your P4 Code Review administrator to disabled, safe, and unsafe, see "[Markdown](#)" on [page 631](#).

Add or update the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:



```
<?php
// this block should be a peer of 'p4'
'projects' => array(
 'readme_mode' => 'enabled',
),
```

- **enabled:** the use of README markdown files is enabled for project overview pages, text content is displayed for project overview pages with a README markdown file. This is the default.
- **disabled:** the use of README markdown files is disabled for project overview pages, project overview pages are not displayed.

#### Note:

In P4 Code Review 2018.3 and earlier, `readme_mode` could be set to disabled, restricted, or unrestricted. If your `readme_mode` was originally set to restricted or unrestricted and you have upgraded to P4 Code Review 2019.1 or later, P4 Code Review treats `readme_mode` as if it is set to enabled. P4 Code Review uses the markdown configurable to determine the level of Markdown support (default is safe), see ["Markdown" on page 631](#).

## Projects tab initial fetch

By default, all of the projects are fetched for the **Projects** tab with a single call. If your P4 Code Review system has a large number of projects, it can take some time to populate the **Projects** tab. The `fetch` configurable is used to tell P4 Code Review to fetch and display the first x projects and then load the rest from cache in the background. This speeds up the initial display of projects in the **Projects** tab.

Add or update the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
'projects' => array(
 'fetch' => array('maximum' => 50), // defaults to 0 (disabled)
),
```

The default value is 0, all projects are fetched in a single call.

## Allow project members to view project settings

When `allow_view_settings` is set to true, project members, owners, and administrators can view the project **Settings** tab. The project **Automated Tests** and **Automated Deployment** details are hidden from the project members unless they are an owner or an administrator. This enables project members to check the project settings but not change them.

Add or update the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
```

```
'projects' => array(
 'allow_view_settings' => true, // defaults to true
),
```

The default value is true.

## Prevent users from creating a branch that includes paths to which they do not have permission

By default, when creating a branch, P4 Code Review does not check if the user has the necessary permissions for the specified path.

When `permission_check` is set to true, P4 Code Review checks if the user has permissions to the specified path. If the user does not have the necessary permissions for a specified path, the creation of the branch is rejected. Only a super user, administrator, owner, or a user with write permissions is allowed to create a branch for a specific path. P4 Code Review uses the `p4 protects -M` command with the `ztg` option to determine if a user has necessary permissions for a specified path.

If a specific branch has multiple branch paths, P4 Code Review checks all branch paths and if the user lacks permission for a single branch, they are not allowed to delete the branch. If a user attempts to modify a project-level workflow, P4 Code Review checks all branch paths and if the user lacks permission for a single branch, they are not allowed to edit the project-level workflow.

```
<?php
// this block should be a peer of 'p4'
'projects' => array(
 'permission_check' => true, // defaults to false
),
```

The default value is false, P4 Code Review allows a user to create a branch with a specific path without checking if the user has access to it.

## Redis server

P4 Code Review requires Redis to manage its caches and by default P4 Code Review uses its own Redis server on the P4 Code Review machine. P4 Code Review caches data from the P4 Server to improve the performance of common searches in P4 Code Review and to reduce the load on the P4 Server.

### Redis configuration:

- "P4 Code Review connection to Redis " on the facing page: [SWARM\\_ROOT/data/config.php](#).
- "Redis server configuration file" on page 665: `/opt/perforce/etc/redis-server.conf`.

#### Tip:

- If required, you can use your own Redis server instead of the P4 Code Review Redis server. For instructions on how to configure P4 Code Review to use your Redis

server, see ["Use your own Redis server" on page 224](#).

- If users and groups are added, edited, or deleted in the P4 Server while the P4 Code Review server is shutdown, the P4 Code Review user and group Redis caches will be out of sync. After the P4 Code Review server has restarted, run a curl request to verify that the Redis caches are in sync with the P4 Server. For the Redis cache verification steps, see ["Manually verify the Redis caches" on page 665](#).
- When P4 Code Review starts it verifies the Redis cache, during this time you cannot log in to P4 Code Review. The time taken to verify the Redis cache depends on the number of users, groups, and projects P4 Code Review has. Start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves in the redis-server.conf file, see ["Redis server configuration file" on page 665](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## P4 Code Review connection to Redis

The redis block of the [SWARM\\_ROOT/data/config.php](#) file contains the password, namespace, host and port details of the P4 Code Review Redis server:

```
<?php
// this block should be a peer of 'p4'
'redis' => array(
 'options' => array(
 'password' => null, // Defaults to null
 'namespace' => 'Swarm',
 'server' => array(
 'host' => 'localhost', // Defaults to 'localhost' or enter your Redis server hostname
 'port' => '7379', // Defaults to '7379' or enter your Redis server port
),
),
 'items_batch_size' => 100000,
 'check_integrity' => '03:00', // Defaults to '03:00' Use one of the following two formats:
 // 1) The time of day that the integrity check starts each day. Set in 24 hour
format with leading zeros and a : separator
 // 2) The number of seconds between each integrity check. Set as a
positive integer. Specify '0' to disable the integrity check.
 'population_lock_timeout' => 300, // Timeout for initial cache population. Defaults to 300
seconds.
),
```

**Configurables:**

- **password:** Redis server password. Defaults to null and should be left at default if using the P4 Code Review Redis server.
- **namespace:** the prefix used for key values in the Redis cache. Defaults to `Swarm` and should be left at default if using the P4 Code Review Redis server.

**Note:**

If you have multiple P4 Code Review instances running against a single Redis server, each P4 Code Review server must use a different Redis namespace. This enables the cache data for the individual P4 Code Review instances to be identified. The namespace is limited to  $\leq 128$  characters.

If one or more of your P4 Code Review instances is connected to multiple P4 Servers, the Redis namespace includes the server label and the character limit is reduced to  $\leq 127$  characters, see ["Multiple P4 Server instances" on page 638](#).

- **host:** Redis server hostname. Defaults to `localhost` and should be left at default if using the P4 Code Review Redis server.
- **port:** Redis server port number. Defaults to `7379`. P4 Code Review uses port `7379` as its default to avoid clashing with other Redis servers that might be on your network. It should be left at default if using the P4 Code Review Redis server. The default port for a non-P4 Code Review Redis server is `6379`.
- **items\_batch\_size:** Maximum number of key/value pairs allowed in an mset call to Redis. Sets exceeding this will be batched according to this maximum for efficiency. Defaults to `100000`.

**Note:**

The default value of `100000` was chosen to strike a balance between efficiency and project data complexity. This value should not normally need to be changed, contact support before making a change to this value.

- **check\_integrity:** In some circumstances, such as when changes are made in the P4 Server when P4 Code Review is down or if errors occur during updates, the Redis cache can get out of sync with the P4 Server. P4 Code Review can run a regular integrity check to make sure that the Redis caches and P4 Server are in sync. If an integrity check finds an out of sync cache file, P4 Code Review automatically updates the data in that cache.

The `check_integrity` configurable specifies when the Redis cache integrity check is run. Set as a specific time of day (24 hour format with leading zeros) or a number of seconds (positive integer) between checks. Disable the integrity check with `'0'`. Defaults to `'03:00'`.

- **population\_lock\_timeout:** specifies the timeout, in seconds, for initial cache population. If you have a large P4 Code Review system, increase this time if the initial cache population times out. Defaults to `300` seconds.

## Redis server configuration file

### Tip:

To fine-tune your Redis server settings, make your changes in the Laminas Redis adapter. For information about the Laminas Redis adapter, see the [Laminas Redis Adapter documentation](#).

The Redis server configuration for P4 Code Review is defined in `/opt/perforce/etc/redis-server.conf` and defaults to:

```
bind 127.0.0.1
port 7379
supervised auto
save ""
dir /opt/perforce/swarm/redis
```

### Default values:

### Tip:

- The default settings are shown below, the `redis-server.conf` file contains more detailed information about the Redis configuration for P4 Code Review.
- On P4 Code Review systems with a large number of users, groups, and projects, start-up time can be improved by persisting the memory cache. You can persist the memory cache by disabling background saves and enabling append saves, see the `redis-server.conf` file comments for detailed information.

- `bind 127.0.0.1` - Redis server IP address (loopback interface)
- `port 7379` - Redis server port number
- `supervised auto` - detects the use of upstart or systemd automatically to signal that the process is ready to use the supervisors
- `save ""` - background saves disabled, recommended.
- `dir /opt/perforce/swarm/redis` - the directory the Redis cache database is stored in.

## Manually verify the Redis caches

### Important:

- Only *admin* and *super* users can verify the Redis cache.
- On P4 Code Review systems with a large number of users, groups, and projects verification of the caches can take some time and can impact performance. In normal operation, the Redis cache and P4 Server data should stay in sync and there is no need to verify the caches. If you have a large number of users, groups, and projects, we recommend that you verify the individual user and group caches rather than using **Verify All**.

In normal operation the Redis cache and P4 Server data should stay in sync and there is no need to verify the caches. However, in some circumstances, such as when changes are made in the P4 Server when P4 Code Review is down or if errors occur during updates, the Redis cache can get out of sync with the P4 Server.

If you suspect that you have a cache issue, you can verify the cache manually from the System Information page.

**For example, to verify the User cache:**

1. Navigate to the **User id** dropdown menu.
2. Select **System Information**.
3. Click the **Cache Info** tab.
4. Click the **Verify User** button.

P4 Code Review responds with a message to say that it has successfully verified the user cache.

If verification finds the cache is out of sync with the P4 Server, P4 Code Review automatically updates the data in that cache.

For more information, see ["Verify buttons" on page 720](#).

## Check progress of the verification request

**To check the progress of the verification request:**

1. From the **Cache Info** tab.
2. Click the **Refresh Status** button.

## Review cleanup

This section describes the configurables that can be set for review cleanup.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

When a review is created in P4 Code Review, it creates its own version of the changelist, leaving the user's own changelist untouched so as not to interfere with the user's ongoing work. Each time new work is submitted to the review, P4 Code Review creates a new changelist so that there is a versioned history of the review.

When the review is finally committed it is P4 Code Review's own changelist that is committed. P4 Code Review's changelists are generally hidden from the users, but the user's changelist will remain open. By default it is necessary for the user to tidy up and remove this changelist themselves after the review has been completed.

There is an option to do this automatically when the review is committed. Configuration for this is expressed with a reviews block in the [SWARM\\_ROOT/data/config.php](#) file, similar to the following example:

```
<?php
'reviews' => array(
 'cleanup' => array(
 'mode' => 'user', // auto - follow default, user - present checkbox(with default)
 'default' => false, // clean up pending changelists on commit
 'reopenFiles' => false, // re-open any opened files into the default changelist
),
),
```

By default, this option is enabled but defaults to no clean up (so a user can select the option if they want to when they commit a review).

If the P4 Server user that P4 Code Review is configured to use is a super user, then the user can clean up all user changelists associated with a review. If this is not the case, then the user who commits a review can only clean up changelists that are in their name.

By default, P4 Code Review only cleans up changelists that are owned by the committing user. In the case where a user is committing a review that has been contributed to by other users, their changelists will not be cleaned up.

If you want to configure P4 Code Review to clean up all changelists contributing to a review, regardless of whether they are owned by the committing user, you can do this by granting the P4 Server user that P4 Code Review is configured as 'super' permissions/privileges (rather than just the 'admin' permissions/privileges that P4 Code Review requires for its other operations).

#### Note:

There is an API option that allows full cleanup to be executed with *super user* permissions using an external script. This removes the need for the user to have super privileges. See [Pending review cleanup example](#) for details.

#### ■ mode

If the mode is `user` then a checkbox is displayed when a review is committed, and the user has the option as to whether to clean up changelists or not.

If the mode is `auto`, then no checkbox is displayed, and all committed reviews will either always be tidied up, or never will be, depending on the value of `default`.

#### ■ default

If set to `true`, changelists will always be cleaned up.

If set to `false`, changelists will never be cleaned up.

#### ■ reopenFiles

If a changelist has files checked out (not shelved), then it cannot be deleted. Setting `reopenFiles` to `true` will mean that when a changelist is cleaned up, any opened files will first be moved to the default changelist, allowing the changelist to be removed.

If set to `false`, then a changelist with checked out files will not be cleaned up.

If users normally revert their files after shelving them, then this option may not be needed. If set to `true` then it may result in files appearing in the user's default changelist unexpectedly.

The review cleanup feature is designed to help users keep their workspaces clean, and prevent the proliferation of unwanted changelists after reviews have been approved and committed.

The following caveats should be kept in mind when using this feature:

- The changelists created by P4 Code Review itself will not be touched by this process. There will always be a record of the review history that is kept.
- If the user who commits a review is normally different to the user that did the work, then unless P4 Code Review runs as a super user then many changelists will not be cleaned up.
- If the committer created some of the changelists, then those changelists will be removed. However, changelists created by other users will not be removed.
- If the user has shelved files into changelists without reverting them, then they will still remain in the user's local workspace, and will need to be cleaned up manually.

## Review keyword

This section describes how to configure keywords, and how to use keywords in a changelist description to create a review, and add a changelist to a review.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## Review keyword configuration

The keyword can be configured with a regular expression so that most any keyword syntax can be used. If you choose to customize the review keyword, take care to choose syntax and terminology that is unlikely to occur in a changelist description, to avoid unexpected P4 Code Review activity.

To configure the review keyword, add the following block to the `SWARM_ROOT/data/config.php` file:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'patterns' => array(
 'octothorpe' => array(// #review or #review-1234 with surrounding whitespace/eol
 'regex' => '/(?P<pre>(?:\s|^)\(?\)#(?P<keyword>review|append|replace)(?:-(?P<id>[0-9]+))?(?P<post>[.,!?:;])*(?=\s|$))/i',
 'spec' => '%pre%##keyword%- %id%%post%',
 'insert' => "%description%\n\n#review-%id%",
 'strip' => '/^\s*#(review|append|replace)(-[0-9]+)?(\s+|$)((\s+|^)\s*)#'
```



```
(review|append|replace)(-[0-9]+)?\s*$/i',
),

'leading-square' => array(// [review] or [review-1234] at start
 'regex' => '/^(?P<pre>\s*)\[(?P<keyword>review|append|replace)(?:-(?P<id>[0-9]+))?\](?P<post>\s*)/i',
 'spec' => '%pre%[%keyword%-%id%]%post%',
),

'trailing-square' => array(// [review] or [review-1234] at end
 'regex' => '/(?:P<pre>\s*)\[(?P<keyword>review|append|replace)(?:-(?P<id>[0-9]+))?\](?P<post>\s*)?$/i',
 'spec' => '%pre%[%keyword%-%id%]%post%',
),
),
),
```

Multiple patterns can be specified; the first successful match is used and none of the other patterns are evaluated.

The keyword types are grouped under their identifiers. In each group, the `regex` item specifies the regular expression to be used to identify the review keyword in the changelist description. The `spec` item is used when the review keyword needs to be updated.

#### Note:

The use of named capture groups in the `regex`, for example `(?<pre>\s*)`. The values captured during regex matching are used to replace any identically named placeholder values in the `spec` item that are surrounded by percent `%` characters. In the example configuration above, the `pre` and `post` capture groups and placeholders maintain any whitespace surrounding the review keyword.

For `octothorpe` (or "hashtag") review keywords, these can appear anywhere in the changelist description. The `strip` item is used to ensure that the keyword is removed from the review description if it appears at the start or end of the changelist description. The `insert` item is currently not used; it is included here to prevent future upgrade issues. The intended use case is when a review is started and the changelist does not already contain a review keyword, the `insert` item would be used to add the review keyword to the changelist description.

For more information on named capture groups in PHP, see [Named Capturing Groups and Backreferences](#).

## Create a review

By default, including the keyword `#review` within a changelist description (separated from other text with whitespace, or on a separate line) tells P4 Code Review that a review should begin when the changelist is shelved or committed.

Once a review has begun, P4 Code Review adds the review identifier to the `#review` keyword, for example `#review-1234`. This tells P4 Code Review which review should be updated whenever the original changelist is re-shelved or committed.

**Note:**

- If you create a pre-commit review and a user appends a changelist to your review, the default add mode of your review is changed from replace to append, see ["Add a changelist to a review"](#) below for details.
- P4 Code Review acts on the first valid keyword it finds in the changelist description, P4 Code Review then ignores any further valid keywords it finds in the description.
- P4 Code Review can also accept `[review]` at the start or end of the changelist description, but this form of review keyword is now deprecated and is likely to be removed in a future version of P4 Code Review.

## Add a changelist to a review

Once a review has been started you can add a changelist to the review. It can be useful to add changelists to an existing review. For example, if follow up changes are made to files in a review or if you need to group a number of changelists under a single review.

**Note:**

The changelist must not be part of another review, if it is P4 Code Review will reject it.

**Note:**

P4 Code Review acts on the first valid keyword it finds in the changelist description, P4 Code Review then ignores any further valid keywords it finds in the description.

By default, the `#append-`, `#replace-`, and `#review-` keywords along with the *review identifier* (separated from other text with whitespace, or on a separate line) can be used in a changelist description to add changelists to an existing review. The options available depend on whether the review is pre-commit or post-commit:

- **Pre-commit reviews:**

- `#append- + review identifier`. When you add a changelist to a review with the append option, the files in the changelist are appended to the existing files in the review. This changes the default add mode of the review to append.
- `#replace-+ review identifier`. When you add a changelist to a review with the replace option, all of the files in the review are replaced with the files in the changelist you are adding to the review. This changes the default add mode of the review to replace.

**Note:**

If you replace a pre-commit review with a committed changelist, the new version of the review will be a post-commit review.

- `#review-+ review identifier`. When you add a changelist to a review with this option, the review's default add method is used. The default add method can be either replace or append and it is set by the most recent add method used on the review:

- When a review is first created the default add mode for the review is replace.
- If a changelist is added to the review using append, the default add mode for the review is set to append.
- If a changelist is added to the review using replace, the default add mode for the review is set to replace.
- If you do not know what the current default add mode of the review is, delete the `#review-` keyword and specify the add mode you want by using either `#append- + review identifier`, or `#replace- + review identifier`.
- **Post-commit reviews:**
  - `#review-` or `#replace- + review identifier`. When you add a changelist to a review with the replace changelist option, all of the files in the review are replaced with the files in the changelist you are adding to the review.

**Note:**

If you replace a post-commit review with a pending changelist, the new version of the review will be a pre-commit review.

**Tip:**

When the content of a review is changed, P4 Code Review checks to see which branches are in the new version of the review:

- **If a new branch was added to the review:**
  - Default reviewers on the new branch are added to the review.
  - Moderators from the added branch become moderators for the review alongside the existing moderators.
  - **Only if workflow is enabled:** if the new branch is associated with a workflow, the [workflow is merged](#) with the existing workflow. The most restrictive workflow is used for the review.
- **If a branch is no longer part of the review:**
  - Reviewers for the review are not changed.
  - Moderators from the removed branch no longer moderate the review.
  - **Only if workflow is enabled:** if the branch was associated with a workflow, the branch workflow is removed from the review.

**Example:** Appending a changelist to review 1234:

Add `#append-1234` to the changelist description.

- Original review contains the following files: A#1, B#2, C#1, D#1, and E#4
- Changelist contains the following files: A#2, C#3, E#4 (marked for delete), and F#1
- Appending the changelist to the original review generates a new version of the review that contains the following files: A#2, B#2, C#3, D#1, E#4 (marked for delete), and F#1

### Important:

**Committing a review with P4 Code Review (recommended):** P4 Code Review automatically commits the files in the approved version of the review.

### Committing a review outside of P4 Code Review:

Before you commit the review:

1. Unshelve the review into the pending changelist associated with the review.
2. Reshelve the files in the pending changelist.
3. Commit the pending changelist.
4. This process ensures that all of the files in the approved version of the review are committed.

### Workflow feature [enabled \(default\):](#)

P4 Code Review can be configured to automatically check that the files being committed match the files in the approved version of the review by using the **On commit with a review** workflow rule. For information about the **On commit with a review** rule, see "[Workflow rules](#)" on page 510.

### Workflow feature [disabled:](#)

P4 Code Review can be configured to automatically check that the files being committed match the files in the approved version of the review by using the [strict](#) trigger option.

**Example:** Replacing the files in a review 1234 with the files in a changelist:

Add `#replace-1234` to the changelist description.

- Original review contains the following files: A#1, B#2, C#1, D#1, and E#4
- Changelist contains the following files: A#2, C#3, E#4 (marked for delete), and F#1
- Replacing the original review with the changelist generates a new version of the review that contains the following files: A#2, C#3, E#4 (marked for delete), and F#1

### Tip:

When you replace a review with a changelist, the base revisions of the files in the new version of the review are the base revisions of the files in the replacement changelist.

## Keyword workflow

For workflow examples using keywords, see "[Review creation and modification outside of P4 Code Review](#)" on page 462.

## Remove mentions from review descriptions

This example shows how to use regular expressions in the `patterns` block to remove `@mentions` in changelist descriptions so they don't get added to the review description.

**Tip:**

This method can be adapted to remove anything you want from the review description.

Add the following block to the `SWARM_ROOT/data/config.php` file:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'patterns' => array(
 'mentions' => array(
 'regex' => '/(?:P<mention>\s@\S+$|@\S+($|\s))/',
 'spec' => '%mention%'
),
),
),
```

## Example

Using the above pattern on the following changelist description:

```
Updated copyright date to 2021.
#review
@bruno
@mei
```

If the changelist is shelved, it becomes:

```
Updated copyright date to 2021.
#review-12345
@bruno
@mei
```

The review description would be:

```
Updated copyright date to 2021.
```

## Reviews filter

This section describes the configurables that can be set for the reviews filter.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

How the reviews are sorted on the ["Reviews list" on page 406](#) page is customized by using the following configuration block in the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'filters' => array(
 'filter-max' => 15,
 'result_sorting' => true,
 'date_field' => 'updated', // 'created' displays and sorts by created date, 'updated' displays
and sorts by last updated
),
),
```

## filter-max

When you visit the ["Reviews list" on page 406](#) page, reviews are displayed incrementally **x** reviews at a time until the white space on the page is filled with reviews or there are no more reviews to display. This process continues as you scroll down the page. The filter-max configurable sets this number.

Default value is 15.

## result\_sorting

Controls whether the **Result order** button is displayed on the ["Reviews list" on page 406](#) page.

- Set `result_sorting` to `true` to display the **Result order** button on the ["Reviews list" on page 406](#) page, this is the default setting.
- Set `result_sorting` to `false` to remove the **Result order** button from the ["Reviews list" on page 406](#) page, reviews are sorted based on the `date_field` setting.

## date\_field

If `result_sorting` is set to `true` (default), the `date_field` sets the initial setting for the **Result order** button on the ["Reviews list" on page 406](#) page.

If `result_sorting` is set to `false`, the `date_field` sets the order reviews are displayed in on the ["Reviews list" on page 406](#) page.

- `created`: reviews are sorted by when they were created. This is the default setting.
- `updated`: reviews are sorted by when they were last updated.

When a user starts a new session the **Result order** button is set to the initial sort order set in `date_field`. The review sort order can be changed by the user from the **Result order** button. This setting is remembered even if the user navigates away from the ["Reviews list" on page 406](#) page. When the user starts a new session, the **Result order** button is reset back to the setting in `date_field`.

## Reviews

This section describes the configurables that can be set for reviews.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## Allow author change

When enabled, the author of a review can be changed. This is useful in the case where the original author is no longer available, and someone else needs to take over ownership.

To allow the author of a review to be changed, update the [SWARM\\_ROOT/data/config.php](#) file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'allow_author_change' => true,
),
```

The default value is `false`.

## Allow author obliterate review

When enabled, users can obliterate reviews that they have authored. For details about how to obliterate a review, see ["Obliterate Review" on page 657](#).

### Important:

By default, only users with *admin* and *super* user rights can obliterate a review. Perforce recommends that the `allow_author_obliterate` configurable is kept at its default value of `false`. An obliterated review cannot be reinstated, not even by Perforce Support.

To allow users to obliterate reviews that they have authored, update the [SWARM\\_ROOT/data/config.php](#) file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'allow_author_obliterate' => true,
),
```

The default value is `false`.

## Disable approve for reviews with open tasks

When enabled, P4 Code Review will not allow you to **Approve**, or **Approve and Commit** a review that has open tasks. To approve, or approve and commit a review with open tasks, you must address the tasks first and then set them to **Task Addressed**, or **Not a Task**, see ["Set a task to Task addressed or Not a task" on page 333](#) for details.

### Important:

If the open tasks on the review are archived, the review can then be approved, or approved and committed.

To prevent reviews being approved, or approved and committed when the review has open tasks, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'disable_approve_when_tasks_open' => true,
),
```

The default value is `false`, in this case a warning is displayed for the open tasks but the action is allowed.

## Disable self-approval of reviews by authors

The P4 Code Review 2015.2 release provides the ability to disable review approval by authors, even if they are moderators or administrators. This is useful for development workflows where review by others is of paramount importance.

To disable review approval by authors, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'disable_self_approve' => true,
),
```

The default value is `false`.

## Disable tests on approve and commit

### Important:

- From P4 Code Review 2020.1, the `disable_tests_on_approve_commit` configurable has been removed from P4 Code Review. This functionality has been moved to workflows and is set on a per workflow per test basis. A test can be set to run **On**



**Update** or **On Submit** for a specific [workflow](#) or globally by setting the test on the [global workflow](#).

- If the `disable_tests_on_approve_commit` configurable is in your `"swarm_root"` on [page 712/data/config.php](#) file, it is ignored.

## Protected end states

By default, the content of a review can be updated no matter what state the review is in. Reviews can be protected from automatic content change if they are in a specified state by using the `end_states` configurable.

### Tip:

When a review is in a protected end state, it can still be updated manually by a user from the P4 Code Review UI.

### Workflow:

- **If workflow is enabled** (default): the `end_states` configurable sets the protected review states. A combination of the [On update of a review in an end state rule](#) and [workflows](#) determines which projects and branches have protected reviews.
- **If workflow is disabled**:: when the content of a review is updated, the `end_states` array is checked to see if the review can be updated. If the review state matches a state in the `end_states` array the submit is rejected.

To set the protected review states, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the reviews block. In this example the **Archived**, **Rejected**, and **Approved : Committed** states are specified:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'end_states' => array('archived','rejected','approved:commit'),
),
```

### Valid review states for the array:

- `archived`: the review state is **Archived**.
- `rejected`: the review state is **Rejected**.
- `approved:commit`: the review has been approved and committed, the review state is **Approved : Commit**.
- `approved`: the review state is **Approved**.

By default the array is empty, review content can be updated no matter what state the review is in.

## Expand all file limit

The review page has an **Expand All** button that opens all the files within that review. If the number of files is large, clicking the button might affect performance.

The `expand_all_file_limit` disables the button if the number of files in the review exceeds the given value. If the value is set to zero, the button is always enabled and can therefore open all the files.

To change the expand all file limit, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'expand_all_file_limit' => 10,
),
```

The default value if the option is not specified is `10`.

## Expand group reviewers

By default, reviewer group members are not displayed in the *Individuals area* of the reviews page when they interact with a review (vote, comment, update, commit, archive, etc.). This avoids overloading the *Individuals* area with individual avatars if you have large reviewer groups.

**Note:**

An exception to this behavior is when a member of a reviewer group is also an individual *required reviewer*, in this case their avatar will be displayed in the *Individuals* area.

When `expand_group_reviewers` is set to `true`, reviewer group members are added to the *Individuals* area of the review page when they interact with the review (vote, comment, update, commit, archive, etc.). If you have large reviewer groups, this might affect performance.

To expand group reviewers, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'expand_group_reviewers' => true,
),
```

The default value is `false`.

## Maximum initial reviewers and reviewer groups in secondary navigation bar

By default, a maximum of 6 individual reviewers and 6 reviewer groups are initially displayed in the secondary navigation pane on the Review page. If there are more individual reviewers or reviewer groups on the review, the + n more link is displayed as required so that you can display all of the reviewers and reviewer groups.

**For example, with the default value of 6:**

- **8 individual reviewers on the review:** 6 displayed in the secondary navigation bar and the remainder are available on the + 2 more link
- **9 reviewer groups on the review:** 6 reviewer groups are displayed in the secondary navigation bar and the remainder are available on the + 3 more link

To edit the maximum number of reviewers and reviewer groups initially displayed in the secondary navigation pane, update the [SWARM\\_ROOT/data/config.php](#) file to include the following configuration item within the reviews block. For example, to set the maximum number of reviewers and reviewer groups to 10:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'max_secondary_navigation_items' => 10, // defaults to 6 reviewers
),
```

The default value if the option is not specified is 6.

## Moderator behavior when a review spans multiple branches

By default, when a review spans multiple project branches that have different moderators, only one moderator from any one of the branches needs to approve the review. Set the configurable to **each** to require one moderator from each branch approve the review. If a moderator belongs to more than one of the branches spanned by the review, their approval will count for each of the branches they belong to.

To require one moderator from each branch to approve a review, edit the [SWARM\\_ROOT/data/config.php](#) file and set the `moderator_approval` configurable to **each** in the `reviews` section of the codeblock:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'moderator_approval' => 'each', // 'any|each'
),
```

The default value is **any**.

## More context lines

The [Review display](#) and [Changelist display](#) pages have **Show More Lines above** and **Show More Lines Below** buttons on the [file diff view](#) when a file is expanded. The buttons are used to display more lines of context for the file being viewed. By default 10 extra lines are displayed.

To change the number of lines displayed when the buttons are clicked, update the [SWARM\\_ROOT/data/config.php](#) file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'more_context_lines' => 15, // defaults to 10 lines
),
```

The default value if the option is not specified is 10.

## Process shelf file delete when

By default, when you delete files from a shelved changelist, the files are not removed from the associated review.

When you have a review state set in the `process_shelf_delete_when` array, P4 Code Review will automatically carry out the following steps when a file is deleted from a shelf:

1. Check to see if the shelf is associated with a review.
2. The deleted file is only removed from the review if the review state matches the review state in the `process_shelf_delete_when` array.

### Tip:

- More than one review state can be specified in the `process_shelf_delete_when` array.
- When files are removed from a review, any votes on the review become stale. The vote counts are reset, and the vote indicators are muted.
- If `process_shelf_delete_when` is configured and the review is in a state that is specified in the array: When the review contains a file that is a common to multiple changelists that are associated with the review, if the owner of any of the associated pending changelists deletes the common file from their shelf, that file will be removed from the review, and a new version of the review will be created. This is true even if the file is not the most recent version in the review.

### Important:

Required for `process_shelf_delete_when`:

- A supported version of P4 Server (standard maintenance), see ["P4 Server requirements" on page 98](#).
- You must have the `swarm.shelvedel shelf-delete` trigger line in the P4 Server trigger table, see ["Installing triggers" on page 187](#).

- You must satisfy the P4 Code Review trigger dependencies, see "[Trigger dependencies](#)" on page 100.

**Important:**

Do not use the P4 Code Review user that is configured in the [P4 Code Review configuration file](#) when deleting shelves, or deleting files from shelves. The P4 Code Review logic processes the *shelf-delete* trigger event, if the event is invoked by the P4 Code Review user it is rejected. The delete operations will fail.

To remove files from a review when they have been deleted from an associated shelved changelist and the review is in a specified state, update the [SWARM\\_ROOT/data/config.php](#) file to include the following configuration item within the reviews block. In this example the **Needs Review** and **Needs Revision** states are specified:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'process_shelf_delete_when' => array('needsReview', 'needsRevision'),
),
```

**Valid review states for the array:**

- `needsReview`: the review state is **Needs Review**.
- `needsRevision`: the review state is **Needs Revision**.
- `rejected`: the review state is **Rejected**.
- `approved`: the review state is **Approved**.

Leave the array empty to disable this feature, see below. This is the default setting for the array.

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'process_shelf_delete_when' => array(),
),
```

**Note:**

If a review is **Approved and Committed** or **Archived**, files will not be removed from the review when they are deleted in an associated shelf. This is true even if the `approved:commit` or `archived` states are in the `process_shelf_delete_when` array.

## Synchronize review description

By enabling the synchronization of review descriptions, it becomes possible to update the description of a review by updating the description of a changelist associated with the review. Whenever an associated changelist is saved, the text of the review will be updated to match.

**Note:**

The update is not triggered if:

- another changelist is attached to a review.
- files are updated in a changelist attached to a review
- a user selects **Update pending changelist** when editing a review description.

To enable synchronize review descriptions, update the `SWARM_ROOT/data/config.php` file to include the following configuration item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'sync_descriptions' => true,
),
```

The default value is `false`.

## Review complexity

Review complexity is displayed for each review in the **Complexity** column on the "[Reviews list](#)" on [page 406](#) to help users to quickly see how complicated each review in the list is. By default, review complexity is based on the number of lines changed in the files in a review and simply displayed as high, medium, or low for each review. The values for high and low complexity are set using the `high` and `low` configurables.

**Tip:**

- Review complexity is only calculated for a review when the review is updated and the file content has changed.
- Review complexity is only stored for the current version of a review.
- If you change the high or low complexity value, complexity is only recalculated for a review when the review is updated and the file content has changed.

To change the complexity values, update the `SWARM_ROOT/data/config.php` file with item within the reviews block:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'statistics' => array(
 'complexity' => array(
 'calculation' => 'default', // 'default|disabled'
 'high' => 300,
 'low' => 30
)
)
),
```

```
),
),
```

- **calculation**
  - default the complexity is based on the number of lines changed in the files in the review. For the purposes of complexity, a new file is counted as the number of lines in the file. Default value is `default`.
  - `disabled` set to `disabled` to disable the complexity calculation.
- **high** set the number (integer) of line changes that are considered high for your organization. Default value is `300`.
- **low** set the number (integer) of line changes that are considered low for your organization. Default value is `30`.

**By default, complexity is calculated as:**

- **High:** number of lines changed in files in the review  $\geq$  `high`.
- **Medium:** number of lines changed in files in the review  $<$  `high` and  $>$  `low`.
- **Low:** number of lines changed in files in the review  $\leq$  `low`.

## Maximum files limit

**Tip:**

Do not set `max_files` configurable to a low number as this will allow commits to bypass workflow and moderators that are blocking approval. It will not bypass review association. A commit must meet the workflow rule of being associated to a review.

When creating or updating a review you can set a maximum file limit. This limits the maximum number of files that can be added to a review. If a request exceeds the maximum file number limit then a review is not created or updated.

To edit the maximum file limit, update the `max_files` configurable under the reviews section in the [SWARM\\_ROOT/data/config.php](#) file.

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'max_files' => 25,
),
```

By default, `max_files` configurable is set to `0`.

## Automatically resolve outdated files

When committing changes for a review that contains outdated files, you must first update the review or all the shelved files to ensure that every file in the changelist is current. Ensure that the review linked to the changelist is also updated. If a changelist contains files that must be resolved then P4 Code Review will attempt to auto-resolve the files if there are no merge conflicts.

To enable auto-resolve for files in a changelist, update the `auto_resolve` configurable under the reviews section in the [SWARM\\_ROOT/data/config.php](#) file.

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'auto_resolve' => true,
),
```

By default, `auto_resolve` configurable is set to false.

## Search

P4 Code Review's search feature combines user, group, project, and file path searching. The standard P4 Code Review searches can be extended to search file content by using the optional P4 Search API.

You can download the P4 Search API installer from the P4 Search [download page](#) on the Perforce website. For more information, see the [P4 Search Documentation](#) and the P4 Search [release notes](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

Configure P4 Code Review search with the following configuration block in the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
// this block should be a peer of 'p4'
'search' => array(
 'maxlocktime' => 5000, // 5 seconds, in milliseconds
 'p4_search_host' => 'http://myhelixsearch.mydomain.com:4567', // optional URL to the P4
 Search host
),
```



## maxlocktime

The `maxlocktime` key specifies the maximum amount of time, in milliseconds, that any table within the P4 Server should be locked while performing `fstat` command searching. Increasing this value might allow better search results at the expense of potentially blocking other queries on the P4 Server. Decreasing this value impacts the P4 Server less, but may be insufficient for returning the desired search results.

## p4\_search\_host

The `p4_search_host` configurable specifies the URL of your P4 Search server. When configured, P4 Code Review issues API calls to P4 Search to take advantage of its file content and description indexing.

## Comment attachment thumbnail and blur generation

P4 Code Review generates the blur and thumbnail images for P4 Code Review comment attachments and stores them in the [comment attachment depot](#).

To prevent P4 Code Review clashing with P4 Search thumbnail and blur generation, set the P4 Server protections for the P4 Code Review comment attachments depot directory so only the `swarm` user has access to it.

### Tip:

**P4 Code Review 2022.1 Patch 1 and later:** By default, the P4 Code Review package post-installation configuration script offers to create the comment attachment depot directory as `//swarm` and set the protections so only the `swarm` user has access to it.

For example, if the comment attachment depot directory is `//swarm`, set P4 Server protections for `//swarm` to:

```
list user * * -//swarm/...
```

```
admin user swarm * //swarm/...
```

```
super user super * //swarm/...
```

This prevents P4 Search creating thumbnails and blurs for files stored in the comment attachment depot directory.

## More information

- For instructions on configuring the P4 Code Review comment attachments depot directory, see ["Comment attachments" on page 575](#).
- For instructions on setting P4 Server protections, see the [Setting protections with p4 protect](#) section of the [P4 Server Administration Documentation](#).

## Security

There are many strategies for securing a P4 Code Review installation. This section provides guidance on security features P4 Code Review controls, and recommendations for several areas for the system hosting P4 Code Review.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## Require login

### Important:

Prior to P4 Code Review's 2016.1 release, `require_login` defaulted to `false`. For 2016.1 and later releases, the default is `true`.

By default, P4 Code Review prevents anonymous users from viewing any P4 Server resources; users must login to see commits, reviews, etc.

P4 Code Review can be configured to allow anonymous users to access any readable resources (creating or editing resources by anonymous users is not permitted). Add the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'require_login' => false, // defaults to true
),
```

There is one exception: the `/queue/worker` endpoint is available to any user.

### Note:

Service and operator users are not permitted to login. For more information on these user types, see [Types of users](#) in [P4 Server Administration Documentation](#).

## Prevent login

When your P4 Server has users that should not be able to log in to P4 Code Review, for example *service* users involved with P4 Server replicas, the `prevent_login` configuration item can be used to prevent successful authentication.

Add or update the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'prevent_login' => array(
 'service_user1',
 'service_user2',
),
),
```

`prevent_login` defaults to `array()`, which means no users in your P4 Server are prevented from logging into P4 Code Review.

For more information, see [Service users](#) in [P4 Server Administration Documentation](#).

## Auto-create new users

P4 Code Review supports the automatic-creation of new users in Perforce on login if the P4 Server is configured to allow it and the user exists in LDAP. This function is configured in the P4 Server and no P4 Code Review configuration is required. For instructions on how to configure P4 Server to automatically create new users in Perforce on login, see `auth.ldap.userautocreate` and `auth.default.method` in the [Defining LDAP-related configurables](#) section of the [P4 Server Administration Documentation](#).

To enable the auto-creation of users in P4 Code Review, you must set the `auto_create_user` configurable to true in the `SWARM_ROOT/data/config.php` file, which is located under the `security` section. By default, this setting is not defined, which means that user auto-creation is not enabled. The same applies if it is explicitly set to false.

The auto-creation of users also works with non-LDAP users. When user auto-creation is enabled and a non-LDAP user, who does not yet exist in the P4 Server logs in for the first time on the P4 Code Review login page and provides their password, they will see an error message. However, despite this error message, the user will be created in the P4 Server with a blank password. In this case, the user will need to reset their password using the P4 Command-line interface (CLI) or P4 Visual Client (P4V) for security reasons.

## Sessions

P4 Code Review manages logged-in sessions using cookies in the browser, and PHP session storage on the server. P4 Code Review uses reasonable defaults for the cookie and session lifetimes (measured in seconds); when the lifetime is exceeded users need to login again. To specify session lifetimes and garbage collection frequency, add the following configuration block to the `SWARM_ROOT/data/config.php` file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
'session' => array(
 'cookie_lifetime' => 0, // lifetime in seconds, default value is 0=expire when browser
 closed
 'remembered_cookie_lifetime' => 60*60*24*30, // lifetime in seconds, default value is 30
 days
```

```
'cookie_samesite' => 'lax', //SameSite attribute to prevent the web browser from
sending the cookie in cross-site requests
 //default is 'lax' 'strict'|'lax'|'none'
'user_login_status_cache' => 10, // Set in seconds, default value is 10 seconds.
 // Set to 0 to disable the cache and make P4 Code Review
 // check the user login status for every call to Helix Server.
'gc_maxlifetime' => 60*60*24*30, // lifetime in seconds, default value is 30 days
'gc_divisor' => 100, // 100 user requests
),
```

- `cookie_lifetime`

**Optional:** Limits the lifetime of session cookies (in seconds).

Default is 0, which causes the session cookie to expire when the user's browser is closed.

Maximum value is 2147483647 seconds.

**Note:**

When the **Remember Me** checkbox on the [login dialog](#) is checked, the `remembered_cookie_lifetime` value will be used for `cookie_lifetime`.

- `remembered_cookie_lifetime`

**Optional:** Limits the lifetime of session cookies when the **Remember Me** checkbox on the [login dialog](#) is checked.

Default is 60\*60\*24\*30 seconds (30 days). You can enter the value in seconds if you prefer. For example, 2592000 seconds (30 days) is also valid.

Maximum value is 2147483647 seconds.

- `cookie_samesite`

**Optional:** Prevents the web browser from sending cookies in cross-site requests. This can help prevent cross-site request forgery (CSRF) attacks.

- `lax`: The cookie is not sent on cross-site sub-requests, such as calls to load images or frames, but is sent when a user navigates to the URL from an external site, such as by following a link. This is the default.
- `strict`: The web browser will only send the cookie for same-site requests, which means requests that come from the same site that initially set the cookie. If a request comes from a different domain or scheme (even if it's the same domain), no cookies with the attribute `cookie_samesite` set to `strict` will be sent.
- `none`: This can only be used if you are using HTTPS and the cookie is also marked secure; setting `cookie_samesite` to `none` without the secure flag may lead to the cookie being rejected.

- `user_login_status_cache`

**P4 Code Review 2022.1 and later:** P4 Code Review caches the P4 Server login status of users. When P4 Code Review makes any single request, it tests the user's login status many times. The default 10 second cache value is intended to reduce the number of checks performed, reducing the load on the P4 Server.

Default is 10 seconds.

Set to 0 to disable the cache and make P4 Code Review check the login status for every call to the P4 Server.

**Note:**

If a user requests a P4 Code Review operation on the P4 Server when logged out of the P4 Server, but their login cache status shows them as logged in, the operation will fail. If you get too many reports of failed operations, reduce your cache setting.

- `gc_maxlifetime`

**Optional:** If a session is inactive for the specified number of seconds, the user is logged out. User sessions are stored in [SWARM\\_ROOT/data/sessions/](#).

Default is 60\*60\*24\*30 seconds (30 days). You can enter the value in seconds if you prefer. For example, 2592000 seconds (30 days) is also valid.

Maximum value is 2147483647 seconds.

**Note:**

By default, the user's Perforce ticket expires after 12 hours, which also causes them to be logged out.

- `gc_divisor`

**Optional:** Sets how often garbage collection is run based on the number of user requests that are made. The setting range is 1 to 100. Garbage collection deletes user session files that are older than the `gc_maxlifetime` setting from [SWARM\\_ROOT/data/sessions/](#).

- `gc_divisor = 100`: Garbage collection runs after every 100th user request. 100 is the default setting.
- `gc_divisor = 1`: Garbage collection runs after every user request.

**Important:**

If your P4 Code Review system has a large number of users, setting `gc_divisor` to a low number can result in performance issues.

## Security headers

P4 Code Review uses the following security headers to enhance its security:

- "X-Frame-Options" on the next page
- "X-XSS-Protection" on the next page
- "X-Content-Type-Options" on page 691
- "Referrer-Policy" on page 691
- "Strict-Transport-Security (HSTS)" on page 692
- "Subresource Integrity (SRI)" on page 692
- "Content-Security-Policy" on page 693

- ["Permissions-Policy" on page 694](#)
- ["X-Forwarded-For" on page 694](#)

For more information on the security headers, see [this Mozilla Developer Network article](#).

## X-Frame-Options

By default, P4 Code Review emits a `X-Frame-Options` HTTP header set to `SAMEORIGIN`. This prevents embedding of the P4 Code Review interface into other web pages, which avoids *click-jacking* attacks.

If your deployment of P4 Code Review needs to be integrated into another web interface, you can adjust the `X-Frame-Options` header by adjusting the `x_frame_options` item within the security configuration block, found in the `SWARM_ROOT/data/config.php` file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'x_frame_options' => value,
),
),
```

Where value can be one of:

- `'SAMEORIGIN'` - P4 Code Review can only be displayed in a frame hosted on the same domain.
- `'DENY'` - P4 Code Review cannot be displayed in a frame.
- `'ALLOW-FROM URI'` - P4 Code Review can only be displayed in a frame hosted on the specified URI.
- `false` - The `X-Frame-Options` header is not emitted, so P4 Code Review can be embedded without restriction.

For more information on click-jacking attacks, see [this Wikipedia article](#).

## X-XSS-Protection

By default, P4 Code Review emits a `X-XSS-Protection` HTTP header set to `1; mode=block`. This stops P4 Code Review from loading when it detects reflected cross-site scripting (XSS) attacks.

If your deployment of P4 Code Review does not want to stop loading when P4 Code Review detects reflected cross-site scripting (XSS) attacks, you can adjust the `X-XSS-Protection` header by adjusting the `x_xss_protection` item within the security configuration block, found in the `SWARM_ROOT/data/config.php` file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'x_xss_protection' => '1',
),
),
```

For more information on X-XSS-Protection HTTP header, see [Mozilla X-XSS-Protection](#).

## X-Content-Type-Options

By default, P4 Code Review emits a X-Content-Type-Options HTTP header set to nosniff. This header blocks a web browsers' MIME type sniffing, which can transform non-executable MIME types into executable MIME types (MIME Confusion Attacks).

If your deployment of P4 Code Review does not want to emit a X-Content-Type-Options HTTP header, you can adjust the X-Content-Type-Options header by adjusting the `referrer_policy` item within the security configuration block, found in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'x_content_type_options' => 'nosniff',
),
),
```

## Referrer-Policy

By default, P4 Code Review emits a Referrer-Policy HTTP header set to strict-origin-when-cross-origin. This prevents P4 Code Review from sending all the referrer information (origin, path, and query string) to the same site but only sends the origin to other sites.

If your deployment of P4 Code Review needs to send all the referrer information (origin, path, and query string) to the same site, you can adjust the Referrer-Policy header by adjusting the `referrer_policy` item within the security configuration block, found in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'referrer_policy' => 'value',
),
),
```

Where value can be one of:

- 'no-referrer' - No referrer information is sent.
- 'no-referrer-when-downgrade' - Referrer is sent to the same protocol or more secure.
- 'same-origin' - Referrer is sent only for same-origin requests.
- 'origin' - Only the origin (scheme, host, and port) is sent as the referrer.
- 'strict-origin' - Only the origin is sent, but only for HTTPS requests.
- 'origin-when-cross-origin' - Full URL for same-origin requests is sent and origin only for cross-origin requests.
- 'strict-origin-when-cross-origin' - Full URL for same-origin requests, origin only for cross-origin

requests if downgraded. This is the default value.

- 'unsafe-url' - Full URL is sent with all requests.

## Strict-Transport-Security (HSTS)

By default, P4 Code Review uses a Strict-Transport-Security response header set to `max-age=15724800`. This ensures that P4 Code Review is only accessed using HTTPS, instead of HTTP.

If your deployment of P4 Code Review needs to use HTTP, you can adjust the Strict-Transport-Security response header by adjusting the `strict_transport_security_policy` item within the security configuration block, found in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'strict_transport_security_policy' => 'max-age=15724800',
),
),
```

Where value can be one of:

- 'max-age=<expire-time>' - Specifies the time, in seconds, that the browser should remember that a site is only to be accessed using HTTPS.
- 'includeSubDomains': Optional; applies the rule to all subdomains.
- 'preload': Optional; signals the web browser to include the domain in its preload list for HSTS.

For more information on Strict-Transport-Security (HSTS) response header, see [HTTP Strict Transport Security Cheat Sheet](#).

## Subresource Integrity (SRI)

By default, P4 Code Review uses Subresource-Integrity header to enable browsers to verify that resources they fetch (like scripts or stylesheets) are delivered without unexpected manipulation. The value of the integrity attribute is a hash of the resource's contents.

You can adjust the Subresource-Integrity header by adjusting the `subresource_integrity` item within the security configuration block, found in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'subresource_integrity' => 'sha384-
oqVuAfXRRKap7fdgcCY5uykM6+R9Gh0ad+vyT/qVYY5PbJ5r61tP34jA4kFJgNH4=',
),
),
```

The possible hash functions include:



- sha256-<base64-value>
- sha384-<base64-value>
- sha512-<base64-value>

The `crossorigin` attribute is often used with SRI to ensure the correct handling of Cross-Origin Resource Sharing (CORS). Example usage in HTML:

```
<link rel="stylesheet" href="https://example.com/style.css" integrity="sha384-
oqVuAfX RKap7fdgcCY5uykM6+R9Gh0ad+vyT/qVYY5PbJ5r61tP34jA4kFJgNH4="
crossorigin="anonymous">
```

## Content-Security-Policy

By default, P4 Code Review uses the `Content-Security-Policy` header set to `default-src 'unsafe-eval' 'unsafe-inline' 'self' data: blob: *;`. This helps to prevent a variety of attacks such as Cross-Site Scripting (XSS) and other code injection attacks.

You can adjust the `Content-Security-Policy` header by adjusting the `content_security_policy` item within the security configuration block, found in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'content_security_policy' => "default-src 'unsafe-eval' 'unsafe-inline' 'self' data: blob: *;",
),
),
```

The possible values are a combination of directives and their sources. Some common directives include:

- `'default-src'` - Specifies the default source for all content types.
- `'script-src'` - Defines valid sources for JavaScript.
- `'style-src'` - Defines valid sources for stylesheets.
- `'img-src'` - Defines valid sources for images.
- `'connect-src'` - Defines valid sources for fetching via XMLHttpRequest, WebSockets, etc.
- `'font-src'` - Defines valid sources for fonts.
- `'object-src'` - Defines valid sources for plugins like Flash.
- `'media-src'` - Defines valid sources for media files like audio or video.
- `'frame-src'` - Defines valid sources for nested browsing contexts (like iframes).
- `'report-uri'` - Specifies the URI where reports of policy violations are sent.
- `'frame-ancestors'` - Specifies valid parents that may embed a page using ``frame``, ``iframe``, or ``object``.

## Permissions-Policy

By default, P4 Code Review uses the `Permissions-Policy` header to control which origins can use which browser features, both in the top-level page and in embedded frames. For every feature controlled by the `Permissions-Policy` header, the feature is only enabled in the current document or frame if its origin matches the allowed list of origins. This means that you can configure your site to never allow the camera or microphone to be activated. This prevents an injection, such as XSS, which could enable the camera, the microphone, or other browser features.

You can adjust the `Permissions-Policy` header by adjusting the `permissions_policy` item within the security configuration block, found in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'permissions_policy' => 'geolocation=(), camera=(), microphone=()',
),
),
```

## X-Forwarded-For

P4 Code Review uses the `X-Forwarded-For` request header to identify the originating IP address of a client running a load balancer or proxy server in front of P4 Code Review. By default, `X-Forwarded-For` request header `forwarded_address` is set to false.

You can use the `X-Forwarded-For` request header by adjusting the `forwarded_address` item within the security configuration block, found in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'forwarded_address' => false,
),
),
```

### Recommendations for implementation through security groups

Here is a list of best practices for implementation using security groups or the user's preferred setup:

- **Use a trusted proxy:** Ensure to only use a trusted proxy, such as allow lists, Content Delivery Networks (CDN), and API Gateways.
- **Backend servers and other proxies or load balancers should be disabled:** Ensure that direct access to backend servers and other proxies or load balancers is disabled, except through the trusted proxy mentioned above. This will prevent unauthorized access while ensuring that all requests are filtered through the trusted proxy.
- **Continuous monitoring and logging of the X-Forwarded-For header:** Implement monitoring and logging on the `X-Forwarded-For` header to track and identify any suspicious activities. This can help in identifying and preventing potential malicious activity or security threats.

- **Use a secure protocol:** Implement a secure protocol such as HTTPS to encrypt the communications between the client and the load balancers, and between the load balancer and backend server to prevent eavesdropping or tampering with the X-Forwarded-For header.
- **Configure X-Forwarded-For header:** Configure the processing mode of the X-Forwarded-For header (append, preserve, or remove) based on specific technical or security requirements.

## Disable commit

P4 Code Review provides the ability to commit reviews within the P4 Code Review interface. You may want to disable this capability to prevent reviews from being committed by someone other than the review's author. When disabled, the **Approve and Commit** (and **Commit** if the review is already approved) option is removed from the list of [states](#) available to a code review.

To disable commits, set `disable_commit` to `true` within the reviews item in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'disable_commit' => true,
),
),
```

## Restricted Changes

The P4 Server provides two changelist types: `public` (the default), and `restricted`. P4 Code Review honors restricted changelists by preventing access to the changelist, and any associated comments or activity related to the changelist.

If a user has `/list`-level privileges to at least one file in the changelist, P4 Code Review allows the user to see the changelist and any of the files they have permission to see.

To prevent unintended disclosures, email notifications for restricted changes are disabled by default. To enable email notifications for restricted changes, set `email_restricted_changes` to `true` within the security item in the [SWARM\\_ROOT/data/config.php](#) file. For example:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'email_restricted_changes' => true,
),
),
```

### Note:

When `email_restricted_changes` is set to `true`, email notifications for restricted changes are sent to all interested parties with no permissions screening. These notifications might disclose sensitive information.

P4 Code Review can only report on changes that the configured *admin*-level user has access to. When using restricted changes, we advise that you grant the P4 Code Review *admin*-level user access to the restricted files and set `require_login = true` to avoid leaking information to unauthenticated users.

## Limit adding projects to administrators

### Important:

For P4 Code Review 2016.1, the configuration item `add_project_admin_only` was moved from the **security block** to the **projects block**, and the item was renamed to `add_admin_only`. The functionality of this configuration item remains unchanged.

If you do not update your `SWARM_ROOT/data/config.php` configuration file, the old configuration for restricting project creation to administrators continues to work.

If you add the new configuration item `add_admin_only` to the **projects block**, it takes precedence over any remaining `add_project_admin_only` setting in the **security block**.

## Limit adding projects to members of specific groups

### Important:

For P4 Code Review 2016.1, the configuration item `add_project_groups` was moved from the **security block** to the **projects block**, and the item was renamed to `add_groups_only`. The functionality of this configuration item remains unchanged.

If you do not update your `SWARM_ROOT/data/config.php` configuration file, the old configuration for restricting project creation to specific groups continues to work.

If you add the new configuration item `add_groups_only` to the **projects block**, it takes precedence over any remaining `add_project_groups` setting in the **security block**.

## IP address-based protections emulation

A P4 Server can be configured via *protections* to restrict access to a depot in a variety of ways, including by IP address. As P4 Code Review is a web application acting as a client to the P4 Server, often with *admin*-level privileges, P4 Code Review needs to emulate IP address-based restrictions. It does so by checking the user's IP address and applying any necessary restrictions during operations such as browsing files, viewing file content, viewing and adding comments on files.

P4 Code Review also emulates proxy-based protections, in addition to regular IP-based protections emulation. However, P4 Code Review does not detect whether it is connecting to a P4 Proxy or not; it merely attempts to emulate protections table entries that use proxy syntax.

IP address-based protections emulation is disabled by default. To enable IP address-based protections emulation for your P4 Code Review installation, include the following configuration item in the `SWARM_ROOT/data/config.php` file:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
```

```
'emulate_ip_protections' => true,
),
```

**Tip:**

If P4 Code Review is connected to multiple P4 Server instances, you have the following options for `emulate_ip_protections`:

- To apply a global `emulate_ip_protections` setting to all P4 Server instances that P4 Code Review is connected to, set `emulate_ip_protections` in the security block as shown above.
- To apply an individual `emulate_ip_protections` setting to a P4 Server instance, set `emulate_ip_protections` in the `serverid` block for that server. This value overrides the global setting in the security block for the P4 Server instance.

For more information about configuring P4 Code Review for multiple P4 Server instances, see ["Set up the P4 Code Review configuration file for the P4 Servers" on page 639](#).

## Known limitations

- Notification e-mails for reviews or commits include the list of affected files. P4 Code Review cannot reliably know the IP address used to retrieve that e-mail, and makes no attempt to filter the files and their depot paths nor any details included in the description. However, when a user follows a link from the notification e-mail to a restricted resource, that access is denied.
- P4 Code Review filters comments from activity streams, but any comments created prior to upgrading to the 2013.3 release cannot be filtered and may leak sensitive information.
- P4 Code Review displays a comment count in code the review list, code reviews, jobs, and activity streams, but the count does not account for any comments that may be hidden from the user due to association with files the user is restricted from viewing.
- Should P4 Code Review users connect to P4 Code Review via a proxy or VPN, the protections will generally use the IP address of the proxy/VPN.
- When the user's IP address and P4 Code Review's IP address both have restrictions applied, the user experiences the most constraining of the two IP address-based restrictions; P4 Code Review cannot bypass restrictions applied to itself.
- P4 Code Review performs a variety of operations with *admin*-level privileges, on behalf of a user. Even if the P4 Server has IP-based, or `userid`-based protections, installed to prevent access to some or most of its versioned data, P4 Code Review typically does have access to this data. Therefore, *P4 Code Review cannot guarantee that no information leakage will occur*, particularly when custom modules are in use, or P4 Code Review source has been customized.

For more information, see [Access authorization](#) in [P4 Server Administration Documentation](#).

## Disable system info

P4 Code Review provides a **System Information** page, available to users with *admin* or *super* privileges, which displays information about P4 Server that P4 Code Review is configured to use, as well as PHP information and the P4 Code Review log file.

While this information can be invaluable when communicating with Perforce support engineers, you may wish to prevent disclosure of any system information. The **System Information** page can be disabled for all users by adding the following configuration block to the [SWARM\\_ROOT/data/config.php](#) file, at the same level as the `p4` entry:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'disable_system_info' => true, // defaults to false
),
```

Once disabled, the **System Information** link disappears from the [About P4 Code Review](#) dialog, and 403 errors are generated for any attempts to browse to the **System Information** page.

## HTTP client options

P4 Code Review permits configuration of options that are passed through to the underlying Laminas Framework HTTP client. These options can be used to specify SSL certificate locations, request timeouts, and more, and can be specified globally or per host. To use the P4 Code Review Slack integration through a proxy server, see ["Use Slack integration through a proxy server" on page 708](#).

Here is an example configuration:

```
<?php
// this block should be a peer of 'p4'
'http_client_options' => array(
 'timeout' => 10, // default value is 10 seconds

 // path to the SSL certificate directory
 'sslcapath' => "",

 // the path to a PEM-encoded SSL certificate
 'sslcert' => "",

 // the path to Certificate Authority file
 'sslcafile' => "",

 // the passphrase for the SSL certificate file
 'sslpassphrase' => "",

 // optional, per-host overrides;
 // host as key, array of options as value
 'hosts' => array(
 'jira.example.com' => array(
```

```

 'sslcafile' => '/path/to/certs/jira.pem',
 'sslpassphrase' => 'keep my JIRA secure',
 'timeout' => 15,
),
 'jenkins.example.com' => array(
 'sslcafile' => '/path/to/certs/jenkins.pem',
 'sslpassphrase' => 'keep my jenkins very secure',
 'timeout' => 15,
),
 'slack.com' => array(
 'adapter' => 'Laminas\Http\Client\Adapter\Proxy',
 'proxy_host' => 'squidproxy',
 'proxy_port' => '3128',
),
),
),
),

```

See the [Laminas Framework Socket Adapter documentation](#) for more information.

#### Note:

P4 Code Review supports the Laminas component versions in the LICENSE.txt file, features and functions in the Laminas documentation that were introduced in later versions of Laminas will not work with P4 Code Review. The LICENSE.txt file is in the readme folder of your P4 Code Review installation.

#### Warning:

While it is possible to use a self-signed SSL certificate, adding the configuration to do so disables certificate validity checks, making connections to the configured host less secure.

**We strongly recommend against using this configuration option.** For more information about the risks of using a self-signed SSL certificate, see ["Security risks of using a self-signed certificate" on page 287](#).

However, if you need to configure continuous integration, deployment, or JIRA connections and those connections must use a self-signed SSL certificate, set the `sslallowselfsigned` item to `true` for the specific host that needs it, as in the following example:

```

<?php
// this block should be a peer of 'p4'
'http_client_options' => array(
 'hosts' => array(
 'jira.example.com' => array(
 'sslallowselfsigned' => true,
),
),
),
),

```

## Strict HTTPS

To improve the security when users work with P4 Code Review, particularly if they need to do so outside of your network, P4 Code Review provides a mechanism that tries to force web browsers to use HTTPS. When enabled, P4 Code Review's behavior changes in the following ways:

- HTTP requests to P4 Code Review include a meta-refresh to the HTTPS version. If a load balancer handles encryption before requests reach P4 Code Review, the meta-refresh should be disabled. See [below](#).
- A strict transport security header is included for all requests, which pins the browser to using HTTPS for your P4 Code Review installation for 30 days.
- All qualified URLs that P4 Code Review produces use HTTPS for the scheme.
- Cookies are flagged as HTTPS-only.

Here is an example of how to enable strict HTTPS:

```
<?php
// this block should be a peer of 'p4'
'security' => array(
 'https_strict' => true,
 'https_strict_redirect' => true, // optional; set false to avoid meta-refresh
 'https_port' => null, // optional; specify if HTTPS is
 // configured on a non-standard port
),
```

When the `https_strict_redirect` item is set to `false`, P4 Code Review does not add a meta-refresh for HTTP clients. This prevents an endless redirect when a load balancer in front of P4 Code Review applies HTTPS to the client-to-load balancer connection, but not the load balancer-to-P4 Code Review connection.

## Apache security

There are several Apache configuration changes that can improve security for P4 Code Review:

### ▪ Disable identification

By default, each Apache response to a web request includes a list of tokens identifying Apache and its version, along with any installed modules and their versions. Also, Apache can add a signature line to each response it generates that includes similar information. By itself, this identification information is not a security risk, but it helps would-be attackers select attacks that could be successful.

To disable Apache identification, add the following two lines to your Apache configuration:

```
ServerSignature Off
ServerTokens ProductOnly
```

### ▪ Disable TRACE requests



TRACE requests cause Apache to respond with all of the information it has received, which is useful in a debugging environment. TRACE can be tricked into divulging cookie information, which could compromise the credentials being used to login to P4 Code Review.

To disable TRACE requests, add the following line to your Apache configuration:

```
TraceEnable off
```

#### ■ Update SSL configuration

P4 Code Review works correctly with an SSL-enabled Apache. Several attacks on common SSL configurations have been published recently. We recommend that you update your Apache configuration with the following lines:

```
<IfModule mod_ssl.c>
 SSLHonorCipherOrder On
 SSLCipherSuite ECDHE-RSA-AES128-SHA256:AES128-GCM-
 SHA256:RC4:HIGH:!MD5:!aNULL:!EDH
 SSLCompression Off
</IfModule>
```

## PHP security

There are several PHP configuration changes that can improve security for P4 Code Review:

#### ■ Disable identification

By default, PHP provides information to Apache that identifies that it is participating in a web request, including its version.

To disable PHP identification, edit your system's php.ini file and change the line setting `expose_php` to:

```
expose_php = Off
```

#### ■ Remove scripts containing `phpinfo()`

During module development or other debugging, you may need to call `phpinfo()`, which displays PHP's active configuration, compilation details, included modules and their configuration. Typically, you would add a script to P4 Code Review's public directory containing:

```
<?php phpinfo() ?>
```

Any such scripts should be removed from a production instance of P4 Code Review.

## proxy\_mode

By default, P4 Code Review works in proxy mode, passing the end-user's browser IP address to the P4 Server. The P4 Server checks the IP address against its IP protection table to work out what permissions the end-user has.

Set the P4 Code Review `proxy_mode` in the `p4` configuration block of the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
'p4' => array(
 'proxy_mode' => true, // defaults to true
),
```

- `true`: P4 Code Review passes the end-user's browser IP address to the P4 Server allowing the P4 Server to work out the what permissions the end-user has. This is the default setting.
- `false`: the P4 Code Review server IP address is passed to the P4 Server. The P4 Server checks the P4 Code Review IP address against its protections table to work out the permissions the end-user has. The result is that all of the P4 Code Review users will have the same permissions.

### Tip:

If P4 Code Review is connected to multiple P4 Server instances, you have the following options for `proxy_mode`:

- To apply a global `proxy_mode` setting to all P4 Server instances that P4 Code Review is connected to, set `proxy_mode` in the `p4` block as shown above.
- To apply an individual `proxy_mode` setting to a P4 Server instance, set `proxy_mode` in the `serverid` block for that server. This value overrides the global setting in the `p4` block for the P4 Server instance.

For more information about configuring P4 Code Review for multiple P4 Server instances, see ["Set up the P4 Code Review configuration file for the P4 Servers" on page 639](#).

## Short links

[Short links](#) work with your P4 Code Review installation's current hostname, but you have the option of registering/configuring an even shorter hostname to make shareable file/directory links as short as possible.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

1. Register a short domain name, or if you control your own DNS server, a short domain name for your network.
2. Point the short domain name at your P4 Code Review host.
3. Edit the [SWARM\\_ROOT/data/config.php](#) file and add the following configuration block:

```
<?php
// this block should be a peer of 'p4'
'short_links' => array(
 'hostname' => 'myho.st',
),
```

Replace `myho.st` with the short domain name you registered/configured.

If your P4 Code Review is configured to use [HTTPS](#), a [custom port](#), a [sub-folder](#), or any combination of these custom installation options, the short links configuration block should look like:

```
<?php
// this block should be a peer of 'p4'
'short_links' => array(
 'external_url' => 'https://myho.st:port/sub-folder',
),
```

Replace `myho.st` with the short domain name you have registered/configured.

If you have not configured P4 Code Review to use HTTPS, replace `https://` with `http://`.

If you have configured P4 Code Review to run on a custom port, replace `:port` with the correct custom port. Otherwise, remove `:port`.

If you have configured P4 Code Review to run in a sub-folder, replace `/sub-folder` with the correct sub-folder name. Otherwise, remove `/sub-folder`.

#### Important:

The `external_url` configuration item is only honored if you have also configured the ["external\\_url" on page 603](#) item within the ["Environment" on page 602](#) configuration item as well. Otherwise, P4 Code Review could generate short links that cannot correctly link to their corresponding full URLs.

When `external_url` is configured, the `hostname` configuration item is ignored.

## Single Sign-On PHP configuration

P4 Code Review supports P4 AS (HAS) as a Single Sign-On (SSO) provider. This helps to simplify configuration and create a more robust SSO solution.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## P4 AS

To use P4 Code Review with a P4 Server that is configured to use the P4 AS, you must set the P4 Code Review `sso` configurable to either `optional` or `enabled` in the "Swarm configuration file" on [page 179](#).

The P4 AS acts as the SAML Identity Provider (IdP) for P4 Code Review's Service Provider (SP) and enables authentication with your external IdP using the protocol the administrator has configured in the P4 AS.

### Configuring P4 Code Review for P4 AS:

- For an overview of the P4 AS, see [Overview of P4 AS](#) chapter in the [P4 Authentication Service Documentation](#).
- For instructions on configuring the P4 AS, see the [Configuring P4 AS](#) chapter in the [P4 Authentication Service Documentation](#).
- For a more in-depth explanation of P4 Code Review configuration for the P4 AS, see the [Example P4 Code Review configuration](#) chapter in the [P4 Authentication Service Documentation](#).

### Update the `config.php` file to enabled SSO.

Set `sso` to one of the following in the [SWARM\\_ROOT/data/config.php](#) file.

- `enabled` all users must use P4 AS to log in to P4 Code Review.  
The login page displays an input box to enter the email or username. Once the input box is filled, the **Log in with SSO** button is activated. See ["Log in with SSO" on page 384](#).
- `optional` P4 AS is available for users to log in to P4 Code Review but is not enforced.  
The login page displays an input box to enter the email or username. Once the input box is filled, the **Log in with SSO** button is enabled. To log in with SSO, see ["Log in with SSO" on page 384](#).  
The user can also manually log in to P4 Code Review using the **Log in with credentials** button. To log in manually, see ["Log in with a password" on page 383](#).  
If you switch to **Log in with credentials** and then decide to use single sign-on, click **Log in with SSO**.
- `disabled` P4 AS is not available to P4 Code Review. This is the default value.  
The login page displays two input boxes to enter the email or username, and to enter the password. Once the input boxes are filled, the **Log in with credentials** button is activated. See ["Log in with a password" on page 383](#).

Below is an example of the [SWARM\\_ROOT/data/config.php](#) file with SSO enabled.

```
<?php
return array(
 'p4' => array(
 'port' => 'my-helix-core-server:1666',
 'user' => 'admin_userid',
 'password' => 'admin user ticket or password',
 'sso' => 'enabled', // 'disabled'|'optional'|'enabled'
```

```

 // default value is 'disabled'
),
 'log' => array(
 'priority' => 3, // 7 for max, defaults to 3
),
 'mail' => array(
 'transport' => array(
 'host' => 'my.mx.host',
),
),
);

```

## Skip activity

### Important:

We recommend that before enabling the skip activity configurables in the `SWARM_ROOT/data/config.php` file, you must contact Perforce. For contact details, see ["Getting help" on page 1362](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

P4 Code Review activity streams are filtered to display events related to Reviews, Commits (changes), Comments, or Jobs. You can configure P4 Code Review to skip an activity in the `SWARM_ROOT/data/config.php` file, as in the following example:

### Note:

- By default all skip activity configurables are set to false in the `config.php` file.
- When `skip_groups` and `skip_projects` are set to true, this prevents all associated groups and project members from receiving activity.
- When `skip_groups`, `skip_projects` and `skip_followers` are set to true, this prevents all associated groups, project members, and followers from receiving activity.

```

<?php
// this block should be a peer of 'p4'
'activity' => [
 'skip_activity' => false, //If this is enabled all other skip configurables below default to true.
 'skip_groups' => false, // Only skip group activity.
 'skip_projects' => false, // only skip project activity.
 'skip_followers' => false, // only skip followers activity.

```

```
'skip_review' => false, // only skip the review activity.
]
```

#### skip\_activity

When `skip_activity` is set to true, all activities in Swarm are disabled. No events for any activity are sent to any user, group, project, or review.

#### skip\_groups

When `skip_groups` is set to true, no groups will receive activity. If you use any groups as members of projects or reviewers, they will not receive any activity.

#### skip\_projects

When `skip_projects` is set to true, none of the projects associated with the review will receive activity.

#### skip\_followers

When `skip_followers` is set to true, all users that are following the user performing the activity will no longer receive activity.

#### skip\_review

When `skip_review` is set to true, none of the participants (users or groups) will receive activity for actions on the review.

## Slack integration

This section details how to install, configure, and use the P4 Code Review Slack integration. The P4 Code Review Slack integration lets teams collaborate with P4 Code Review notifications in Slack. The Slack integration posts a message in the configured P4 Code Review project Slack channels each time a change is committed, a review is created, or a review is updated in the project.

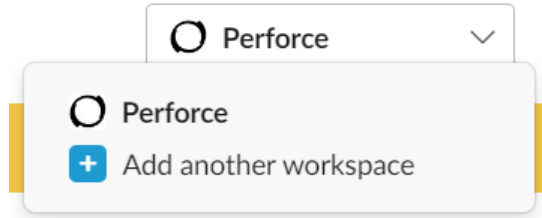
## Prerequisites

- You must be an *admin* or *super* user to integrate P4 Code Review with Slack.
- Find the names of the channels in Slack to which you want project notifications sent.
- Ensure you have write access to the P4 Code Review `config.php` file as you will need to add the Slack channel names as `project ID` to complete the integration.

## Install the Slack integration

To install the Slack integration:

1. Navigate to [Helix Swarm Slack Integration](#).
2. Click **Add to Slack** button.
3. Select the workspace from the top of the page that you wish to integrate Slack with.



4. Request install permissions from your Application Manager.
5. Once the Application Manager approves your request, P4 Code Review Slack integration is installed into your selected workspace.
6. To enable P4 Code Review to work with Slack, edit the P4 Code Review config.php file to include the displayed Slack configuration.

The P4 Code Review config.php file is saved in the `SWARM_ROOT/data/` directory.

**Important:** After the P4 Code Review Slack integration is installed, you must configure the project in the P4 Code Review config.php file. For more information on how to configure a project for Slack integration, see ["Setup project configurations" on page 709](#).

## Slack configuration

Add the following Slack configuration to the P4 Code Review config.php file. The P4 Code Review config.php file is saved in the `SWARM_ROOT/data/` directory.

```
'slack' => array(
 'token' => 'TOKEN',
 'project_channels' => array(), //See "Setup project configurations" on page 709 to configure
a project for Slack integration
 'summary_file_names' => false,
 'bypass_restricted_changelist' => false,
 'summary_file_limit' => 10,
 'user' => array(
 'enabled' => true,
 'name' => 'Helix Swarm',
 'icon' => 'URL',
 'force_user_header' => false,
),
),
```

### token

The Slack app provides a token that is used by P4 Code Review.

## project\_channels

Specifies the P4 Code Review projects ID and Slack channels they post to in the `project_channels` array. You can configure multiple channels to receive a notification for a project. For more information on how to use `project_channels` configurable, see ["Setup project configurations" on the facing page](#).

## summary\_file\_names

Attaches the file to the original notification message sent to a Slack channel.

## bypass\_restricted\_changelist

Allows P4 Code Review to post notification messages to a Slack channel when a change is committed or a review is created for a restricted changelist, default value is false. For more information about a restricted changelist, see ["Restricted Changes" on page 695](#).

## summary\_file\_limit

Limits the number of files shown in the original notification message sent to a Slack channel, default value is 10.

## user

Specifies details about the P4 Code Review user in a Slack notification.

- `enabled`: Forces the Swarm app to use the custom user name. This overrides the P4 Code Review app details.
- `name`: This is the user name shown in the Slack channel when a notification is posted.
- `icon`: This is the avatar icon shown in the Slack channel when a notification is posted, overrides the avatar set in the P4 Code Review app.
- `force_user_header`: The Slack notification shows the user name and avatar only for the first notification by a user. Slack does not apply the user name and avatar to any more notifications from the same user. By default this is set to false.

When set to true, Slack adds the user name and avatar for every notification.

## Use Slack integration through a proxy server

This section details how you can configure the Slack integration to use a proxy server. For example, add the following Squid proxy server configuration to the `SWARM_ROOT/data/config.php` file.

```
<?php
// this block should be a peer of 'p4'
'http_client_options' => array(
 // optional, per-host overrides;
 // host as key, array of options as value
 'hosts' => array(
 'slack.com' => array(
 'adapter' => 'Laminas\Http\Client\Adapter\Proxy',
 'proxy_host' => 'squidproxy',
```



```

 'proxy_port' => '3128',
),
),
),

```

## adapter

Specifies the connection adapter used to establish a connection to the server, sending requests, and receiving responses.

## proxy\_host

Specifies the proxy server address.

## proxy\_port

Specifies the proxy server Transmission Control Protocol (TCP) port.

See the [Laminas Framework Socket Adapter documentation](#) for more information.

# Setup project configurations

This section details how you can configure a project in P4 Code Review to send Slack notifications to a Slack channel.

### Important:

After project configuration is complete, you must restart the workers and the configuration cache for the updated configuration to take effect.

- For more information on how to restart workers, see ["Workers" on page 740](#) section.
- For more information on how to restart configuration cache, see ["Reload Configuration button" on page 720](#) section.

To keep the private projects secure, ensure that the Slack channel you wish to send notifications to is a private channel. For more information on how to add the P4 Code Review application to a private channel, see ["Add P4 Code Review application to a private channel" on page 711](#).

To send notifications to a channel for a project, you must configure the `project_channels` block as follows:

```

<?php
//This block should be a peer of 'p4'
'slack' => array(
 'token' => 'xoxb-3875744143189-3925283455653-pxxZoTKz3JqUeDXGd66XFW4D',
 'project_channels' => array(
 //Project myproject will go into slack channel myproject-channel
 'myproject' => array(
 'myproject-channel',
),
),
),

```

```
//Project with no setting will go to no-project-channel
'*no-project-channel*' => array(
 'no-project-channel',
),
//No project on this review/change
'*without_project*' => array(
 'no-link-project',
),
//All notification get sent here even if we matched above
'*all*' => array(
 'all-notifications',
),
),
),
```

## project\_channels

Specify the P4 Code Review projects ID and the Slack channels they post to in the `project_channels` array. You can configure multiple channels to receive a notification for a project.

### Example

- **P4 Code Review project:** myproject
- **Slack channel:** myproject-channel

All notifications for the project `myproject` will go to the `myproject-channel` Slack channel.

### \*no-project-channel\* (optional)

For a project that has no defined project mapping and with `*no-project-channel*` configurable defined, the Slack notifications are sent to `*no-project-channel*`. For example, all notifications for the project `myproject` goes to `myproject-channel` and similarly all notifications for any other project goes to `*no-project-channel*`.

### \*without\_project\* (optional)

When `*without_project*` configurable is defined, it means that no notification is sent if the notification is not associated with a project.

### \*all\* (optional)

When `*all*` configurable is defined, all notifications are sent here even if `*no-project-channel*` and `*without_project*` options are configured.

P4 Code Review is now configured to send notifications to a Slack channel for a project.

## Add P4 Code Review application to a private channel

To send notifications to a private channel you must add the P4 Code Review application to the private channel as follows:

1. In Slack, open the private channel you would like to send notifications to.
2. Select your picture or avatar in the top right to open the channel menu.
3. Select the **Integrations** tab.
4. For the **P4 Code Review Bot** select **Add apps** to add the application to your private channel.

P4 Code Review can now send notifications to your private channel.

## Slack notification trigger

P4 Code Review will post a message on a Slack channel for every commit. This includes each time a change is committed, a review is created, or a review is updated. Any update to a review in P4 Code Review is displayed as a recently occurred activity stream in the associated Slack channel.

Here is a list of actions that will trigger P4 Code Review to send a notification to the associated project Slack channel.

Event	Description
joined	When a user joins the review
left	When a user leaves the review
made their vote required on	When a user's vote has been made required
made their vote optional on	When a user's vote has been made optional
disabled notifications on	When a user disables the notifications for a review
re-enabled notifications on	When a user re-enables the notifications for a review
edited reviewers on	When the reviewers are edited for a review
voted	When a user votes up or down

Event	Description
cleared their vote on	When a user clears their vote
requested further review of	When a user changes the review state to <b>needsReview</b>
requested revisions to	When a user changes the review state to <b>needsRevision</b>
rejected	When a user changes the review state to <b>rejected</b>
archived	When a user changes the review state to <b>archived</b>
approved	When a user changes the review state to <b>approved</b>
updated description of	When a user updates the description of the review
updated files in	When a user updates the files in a review

## swarm\_root

`swarm_root` refers to the directory where P4 Code Review lives in the filesystem. Depending on how you installed P4 Code Review, the default location could be:

- For a package installation: `/opt/perforce/swarm`.
- For a tarball installation: wherever you unpacked P4 Code Review. If you are unsure, you could check your web server's configuration to discover where P4 Code Review exists.

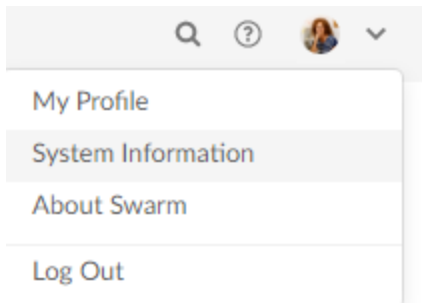
## System Information

### Note:

The system information page is available to users with *admin* or *super* privileges.

The system information page provides details that can be useful to Perforce support engineers when you ask for assistance.

To display the system information page, click **System Information** from the **User id** dropdown menu:



#### In this section:

- ["Perforce" below tab](#)
- ["Log" on the next page tab](#)
- ["PHP Info" on the next page tab](#)
- ["Queue Info" on page 715 tab](#)
- ["Cache Info" on page 719 tab](#)

## Perforce

The **Perforce** tab provides information similar to the `p4 info` command.

### System Information

	Perforce	Log	PHP Info	Queue Info	Cache Info
User Name	admin				
Client Name	*unknown*				
Client Cwd	/				
Client Host	swarm-deploy				
Peer Address	127.0.0.1:56220				
Client Address	127.0.0.1				
Server Address	localhost:4433				
Server Root	/opt/perforce/servers/master				
Server Date	2018/04/27 07:02:59 -0700 PDT				
Server Uptime	242:13:25				
Server Version	P4D/LINUX26X86_64/2018.1/1637071 (2018/03/23)				
Server Services	standard				
Server License	none				
Case Handling	sensitive				

## Log

Click the **Log** tab to display the most recent entries, up to 1 megabyte, in P4 Code Review's log file, which resides in [SWARM\\_ROOT/data/log](#). Review the logging levels for [P4 Code Review logs](#) to ensure that the entries you want to see are included.

If your log entries include a critical error, an arrow is displayed on the left of the log entry. Click the error to display the stack trace that accompanies the error.

## System Information

Perforce	Log	PHP Info	Queue Info	Cache Info	Refresh Log	Download Log
Time	Severity	Message				
2023-11-23T05:16:21-08:00	DEBUG	P4 (000000003a840a55000000004adf2b14) start command: Authenticating 10.153.120.243 for user super {"request":"/main/info/log?format=partial&_=1700745379858","pid":1389629,"REMOTE_ADDR":"10.153.120.243"}				
2023-11-23T05:16:21-08:00	DEBUG	P4 (00000000704284d10000000005b53d5fb) start command: Authenticating 10.153.120.243 for user super {"request":"/main/info/phpinfo/", "pid":1385167, "REMOTE_ADDR":"10.153.120.243"}				
2023-11-23T05:16:21-08:00	DEBUG	P4 (00000000908d83110000000021c9cf7a) start command: spec -o user {"request":"/main/api/v11/users?fields%5B%5D=avatar&fields%5B%5D=User&fields%5B%5D=FullName&fields%5B%5D=Email&fields%5B%5D=JobView&fields%5B%5D=Reviews", "pid":1389639, "REMOTE_ADDR":"10.153.120.243"}				
2023-11-23T05:16:21-08:00	DEBUG	P4 (00000000908d83110000000021c9cf7a) start command: info {"request":"/main/api/v11/users?fields%5B%5D=avatar&fields%5B%5D=User&fields%5B%5D=FullName&fields%5B%5D=Email&fields%5B%5D=JobView&fields%5B%5D=Reviews", "pid":1389639, "REMOTE_ADDR":"10.153.120.243"}				
2023-11-23T05:16:21-08:00	DEBUG	P4 (000000003f463ef90000000006126b26) start command: info {"request":"/main/api/v11/groups?fields%5B%5D=1d&fields%5B%5D=name&expand=false", "pid":1385246, "REMOTE_ADDR":"10.153.120.243"}				
2023-11-23T05:16:21-08:00	DEBUG	P4 (000000003f463ef90000000006126b26) start command: spec -o group {"request":"/main/api/v11/groups?fields%5B%5D=1d&fields%5B%5D=name&expand=false", "pid":1385246, "REMOTE_ADDR":"10.153.120.243"}				
2023-11-23T05:16:21-08:00	INFO	User super making a request for endpoint http://freya.das.perforce.com/main/api/v11/users?fields%5B%5D=avatar&fields%5B%5D=User&fields%5B%5D=FullName&fields%5B%5D=Email&fields%5B%5D=JobView&fields%5B%5D=Reviews for method GET {"request":"/main/api/v11/users?fields%5B%5D=avatar&fields%5B%5D=User&fields%5B%5D=FullName&fields%5B%5D=Email&fields%5B%5D=JobView&fields%5B%5D=Reviews", "pid":1389639, "REMOTE_ADDR":"10.153.120.243"}				
2023-11-23T05:16:21-08:00	INFO	User super making a request for endpoint http://freya.das.perforce.com/main/api/v11/groups?fields%5B%5D=1d&fields%5B%5D=name&expand=false for method GET {"request":"/main/api/v11/groups?fields%5B%5D=1d&fields%5B%5D=name&expand=false", "pid":1385246, "REMOTE_ADDR":"10.153.120.243"}				

## Log tab buttons:

- **Refresh Log** button: click to load the latest log entries, up to 1 megabyte.
- **Download Log** button: click to download the log and all of its log entries.

## PHP Info

Click the **PHP Info** tab to display PHP's own information display generated by executing `phpinfo()`, PHP's internal [diagnostic display](#).

## System Information

[Perforce](#)
[Log](#)
[PHP Info](#)
[Queue Info](#)
[Cache Info](#)

PHP Version 7.0.32-0ubuntu0.16.04.1



System	Linux vagrant-swarm-devline-ub1604 4.4.0-116-generic #140-Ubuntu SMP Mon Feb 12 21:23:04 UTC 2018 x86_64
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.0/apache2
Loaded Configuration File	/etc/php/7.0/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.0/apache2/conf.d
Additional .ini files parsed	/etc/php/7.0/apache2/conf.d/10-opcache.ini, /etc/php/7.0/apache2/conf.d/10-pdo.ini, /etc/php/7.0/apache2/conf.d/15-xml.ini, /etc/php/7.0/apache2/conf.d/20-apcu.ini, /etc/php/7.0/apache2/conf.d/20-calendar.ini, /etc/php/7.0/apache2/conf.d/20-ctype.ini, /etc/php/7.0/apache2/conf.d/20-curl.ini, /etc/php/7.0/apache2/conf.d/20-dom.ini, /etc/php/7.0/apache2/conf.d/20-exif.ini, /etc/php/7.0/apache2/conf.d/20-fileinfo.ini, /etc/php/7.0/apache2/conf.d/20-ftp.ini, /etc/php/7.0/apache2/conf.d/20-gettext.ini, /etc/php/7.0/apache2/conf.d/20-iconv.ini, /etc/php/7.0/apache2/conf.d/20-imagick.ini, /etc/php/7.0/apache2/conf.d/20-intl.ini, /etc/php/7.0/apache2/conf.d/20-json.ini, /etc/php/7.0/apache2/conf.d/20-mbstring.ini, /etc/php/7.0/apache2/conf.d/20-perforce.ini, /etc/php/7.0/apache2/conf.d/20-phar.ini, /etc/php/7.0/apache2/conf.d/20-posix.ini, /etc/php/7.0/apache2/conf.d/20-readline.ini, /etc/php/7.0/apache2/conf.d/20-shmop.ini, /etc/php/7.0/apache2/conf.d/20-simplexml.ini, /etc/php/7.0/apache2/conf.d/20-sockets.ini, /etc/php/7.0/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.0/apache2/conf.d/20-sysvsem.ini, /etc/php/7.0/apache2/conf.d/20-sysvshm.ini, /etc/php/7.0/apache2/conf.d/20-tokenizer.ini, /etc/php/7.0/apache2/conf.d/20-wddx.ini, /etc/php/7.0/apache2/conf.d/20-xdebug.ini, /etc/php/7.0/apache2/conf.d/20-xmlreader.ini, /etc/php/7.0/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.0/apache2/conf.d/20-xsl.ini, /etc/php/7.0/apache2/conf.d/20-zip.ini
PHP API	20151012
PHP Extension	20151012
Zend Extension	320151012
Zend Extension Build	API320151012,NTS
PHP Extension Build	API20151012,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled

## Queue Info

Click the **Queue Info** tab to display the P4 Code Review task queue information.

### Note:

Workers should be started automatically using a recurring task. For instructions on how to set up a recurring task to spawn workers automatically, see ["Set up a recurring task to spawn workers"](#) on page 208.

# System Information

[Perforce](#)[Log](#)[PHP Info](#)[Queue Info](#)[Cache Info](#)

Tasks	0
Future Tasks	4
Workers	1
Max Workers	3
Worker Lifetime	595s
Ping Error	false

[Start a Worker](#)[Refresh Tab](#)[Show Task Queue](#)[Start a Temp Worker](#)

- **Tasks:** displays the number of tasks in the queue. If the task queue starts to build up, consider increasing the number of workers.
- **Future Tasks:** displays the number of future tasks in the queue. There are usually at least four future tasks in the queue for checking cache integrity, but there can be more. For example, when a user is making comments, the comment notification is delayed by 30 minutes by default. This is a future task.
- **Workers:** displays the number of workers available to service tasks.

**Tip:****If the number of Workers is 0 (zero):**

- Your workers are not being created automatically. For instructions on how to set up a recurring task to spawn workers automatically, see "[Set up a recurring task to spawn workers](#)" on page 208.
- To process the current task queue while you get your recurring tasks working again, click **Start a Temp Worker**. This is a quick way to get P4 Code Review running again while you fix the problem.

- **Max Workers:** displays the [maximum number of workers](#) that are allowed. The default is three.
- **Worker Lifetime:** displays the [worker lifetime](#) that is configured. The default is 595 seconds (10 minutes less 5 seconds).
- **Ping Error:** not used.



## Queue Info buttons:

- **Start a Worker** button, only available if **Workers** is less than the [Max Workers](#) setting: click to manually start a new P4 Code Review worker. The new worker will only last for the [configured Worker Lifetime](#). If the [number of workers running](#) matches the configured limit, the new worker is not started.

### Tip:

If your recurring task has stopped working and you need to process the task queue while you get it working again, start a temporary worker by clicking **Start a Temp Worker**. This is a quick way to get P4 Code Review running again while you fix the problem.

- **Refresh Tab** button: click to refresh the queue information in the tab.
- **Show Task Queue** button: click to display the Task Queue information. When the task queue is displayed the button changes to **Hide Task Queue**. The task queue shows more information about the tasks and future tasks in the queue.
- **Start a Temp Worker** button, only available if **Workers** is less than the [Max Workers](#) setting: click to manually start a P4 Code Review temporary worker. Typically you use a temporary worker to process the current tasks in the queue while you fix the recurring task that automatically spawns the workers.

This is a special P4 Code Review worker that is not subject to the worker lifetime setting and will not stop until it has processed all of the tasks in the current queue or one of the tasks fails.

## Task Queue

Click the **Show Task Queue** button to display the tasks in the P4 Code Review queue.

## System Information

[Perforce](#) [Log](#) [PHP Info](#) [Queue Info](#) [Cache Info](#)

Tasks	0
Future Tasks	4
Workers	0
Max Workers	3
Worker Lifetime	595s
Ping Error	false

[Start a Worker](#) [Refresh Tab](#) [Show Task Queue](#) [Start a Temp Worker](#)

## Task Queue

Future task - 1627639200.0000.b7b69a2f99a3b1a1abdb2ad48d9e44d4.0

File /opt/perforce/swarm/data/servers/main/queue/1627639200.0000.b7b69a2f99a3b1a1abdb2ad48d9e44d4.0

Time Fri Jul 30 2021 11:00:00 GMT+0100 (British Summer Time)

Type cache.integrity

Id project

Data swarm, project,

Future task - 1627639200.0000.6b2b3b0dba18ad5b3697bd09a3785391.0

File /opt/perforce/swarm/data/servers/main/queue/1627639200.0000.6b2b3b0dba18ad5b3697bd09a3785391.0

Time Fri Jul 30 2021 11:00:00 GMT+0100 (British Summer Time)

Type cache.integrity

Id workflow

Data swarm, workflow,

Future task - 1627639200.0000.4d90339f34d63ac3dd776502b5ca42e4.0

File /opt/perforce/swarm/data/servers/main/queue/1627639200.0000.4d90339f34d63ac3dd776502b5ca42e4.0

Time Fri Jul 30 2021 11:00:00 GMT+0100 (British Summer Time)

Type cache.integrity

Id group

Data swarm, group,

Future task - 1627639200.0000.33981a32e56f1ad82c10d78b4ef01d39.0

File /opt/perforce/swarm/data/servers/main/queue/1627639200.0000.33981a32e56f1ad82c10d78b4ef01d39.0

Time Fri Jul 30 2021 11:00:00 GMT+0100 (British Summer Time)

Type cache.integrity

Id user

Data swarm, user,

If the task queue starts to build up, consider increasing the number of workers.

## Cache Info

Click the **Cache Info** tab to display the P4 Code Review cache information.

### System Information

Perforce
Log
PHP Info
Queue Info
Cache Info

Context	State	Progress
User	Running	Step 4 of 5: Clearing any extraneous Redis cache entries
Group	Queued	Verification complete
Project	Queued	Verification complete
Workflow	Queued	Verification complete

Refresh Status
Verify All
Verify User
Verify Group
Verify Project
Verify Workflow
Reload Configuration

#### Tip:

By default, P4 Code Review runs a regular verification check at 03:00 every day to make sure that the Redis caches and P4 Server are in sync, see ["P4 Code Review connection to Redis"](#) on page 663.

The **Cache Info** tab displays the cache verification state and progress for each of the Redis caches:

- Valid states for verify are; Not Queued, Queued, Running, and Failed.
- Valid states for progress are, Step x of y; (where x is the current step and y is the total number of steps) followed by:
  - Getting Redis cache entries
  - Calculating checksums for Redis keys
  - Calculating checksums for Perforce models
  - Clearing any extraneous Redis cache entries
  - Indexing any missing Perforce models into the Redis cache
  - Verification complete

### Refresh Status button

To refresh the cache information on the tab, click **Refresh Status**.

## Verify buttons

### Important:

On P4 Code Review systems with a large number of users, groups, and projects verification of the caches can take some time and can impact performance.

In normal operation, the Redis cache and P4 Server data should stay in sync and there is no need to verify the caches.

If you have a large number of users, groups, and projects, we recommend that you verify the individual user and group caches rather than using **Verify All**.

In normal operation the Redis cache and P4 Server data should stay in sync and there is no need to verify the caches. However, in some circumstances, such as when changes are made in the P4 Server when P4 Code Review is down or if errors occur during updates, the Redis cache can get out of sync with the P4 Server.

If you suspect that you have a cache issue, the verify buttons enable you to manually verify the caches. Before using the **Verify All** button, see the Important note above. If verification finds the cache is out of sync with the P4 Server, P4 Code Review automatically updates the data in that cache.

### For example, to verify the User cache:

1. Navigate to the **Cache Info** tab on the P4 Code Review **System Information** page.
2. Click **Verify User**.

P4 Code Review responds with a message to say that it has successfully verified the user cache.

## Reload Configuration button

The P4 Code Review configuration is stored in the `SWARM_ROOT/data/config.php` file. If you make a configuration change, P4 Code Review will not use it until the configuration cache has been deleted and reloaded, this forces P4 Code Review to use the new configuration.

### To delete and reload the P4 Code Review configuration cache:

1. Navigate to the **Cache Info** tab on the P4 Code Review **System Information** page.
2. Click **Reload Configuration**.

P4 Code Review responds with a message to say that it has successfully reloaded the config.

P4 Code Review is now using the updated P4 Code Review `config.php` file.

## Tag processor

Including a work-in-progress tag (`#wip` by default) in a pending changelist tells P4 Code Review not to process the changelist when you update files in the shelf. This means that if the changelist already has a review that review will not be updated. If it doesn't have a review, a new review will not be created. For information on using the work-in-progress tag, see ["Work-in-progress tag" on page 371](#).

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the

new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on the previous page](#).

## wip tag configuration

The work-in-progress tag is configured in the `tags` section of the `tag_processor` block in the ["swarm\\_root" on page 712/data/config.php](#) file. It is enabled by default:

```
<?php
// this block should be a peer of 'p4'
'tag_processor' => array(
 'tags' => array(
 'wip' => '/(^|\s)+#wip($|\s)+/i'
),
),
```

## Disabling the work-in-progress tag

You can disable the work-in-progress tag completely if you don't want anyone to use it.

To disable the work-in-progress tag, delete the content of the wip string in the ["swarm\\_root" on page 712/data/config.php](#) file so that it is empty ( `"` ):

```
<?php
// this block should be a peer of 'p4'
'tag_processor' => array(
 'tags' => array(
 'wip' => ""
),
),
```

## Customizing the work-in-progress tag

By default, the work-in-progress tag is `#wip`, but you can change it to whatever you want to use. P4 Code Review uses Perl Compatible Regular Expressions ( PCRE ) to process the wip tag, ensure that your new tag can be processed by PCRE before changing it. If your wip tag cannot be processed by PCRE, P4 Code Review might ignore the tag or stop working correctly.

For example, to change the work-in-progress tag to `#noreview`, update the tag processor wip configuration in the ["swarm\\_root" on page 712/data/config.php](#) file to:

```
<?php
// this block should be a peer of 'p4'
'tag_processor' => array(
 'tags' => array(
 'wip' => '/(^|\s)+#noreview($|\s)+/i'
```

```
),
),
```

## Making the work-in-progress tag case sensitive

To make the work-in-progress tag case sensitive, delete the trailing `i` from the `wip` configuration in the `"swarm_root"` on [page 712](#)/data/config.php file:

```
<?php
// this block should be a peer of 'p4'
'tag_processor' => array(
 'tags' => array(
 'wip' => '/(^|s)+#wip($|s)+/'
),
),
```

## Test definitions

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on [page 720](#).

## project\_and\_branch\_separator

### Note:

The `project_and_branch_separator` configurable character is only applied to tests created from the **Tests** menu, see ["Tests"](#) on [page 528](#).

Specifies the character used to separate the `{projects}` and `{branches}` IDs in the URL that is passed to your CI system. Some CI systems, such as Jenkins, escape the default colon `:` character resulting in a failed test request. If your CI system needs a different separator character, the `project_and_branch_separator` character can be changed to a character that your CI system will accept, for example a dash `-` or an underscore `_`.

Be careful when selecting your separator character, some characters can cause issues when calling tests. For example, a `#` character in a URL call means the branch is treated as an anchor and a `%` character in a JSON body encoding is escaped. Both of these will result in a failed test request.

Configure the `project_and_branch_separator` character with the following configuration block in the `SWARM_ROOT/data/config.php` file:

```
'test_definitions' => array(
 'project_and_branch_separator' => ':',
),
```

The default value for the `project_and_branch_separator` configurable is a colon `:`.

## Trigger options

**Warning:**

Do not install the P4 Code Review extension on your P4 Server if you intend on using P4 Code Review triggers.

**Tip:**

We recommend you use P4 Server Extensions, P4 Server Extensions are easier to install and maintain than triggers. See ["P4 Server Extensions dependencies" on page 99](#).

The P4 Code Review trigger script, `swarm-trigger.pl`, provides the following command line options and configuration items.

## Command-line options

### Synopsis

```
swarm-trigger.pl -t type -v ID [-c config_file]
```

```
swarm-trigger.pl -o
```

```
swarm-trigger.pl -h
```

### Informational options

The following options perform no processing, and simply provide information useful to a P4 Code Review administrator. These are not intended to be used within trigger entries in the P4 Server.

- **-h**  
Displays a list of the available options, some guidance on its usage, and a copy of the trigger table entries that should be configured in the P4 Server to execute the script in its current path.
- **-o**  
Displays a copy of the trigger table entries that should be configured in the P4 Server to execute the script in its current path.

### Operational options

The following command-line options are used in the trigger table entries in the P4 Server to specify how the P4 Code Review trigger script should be executed.

- **-t type**

Specifies the type of processing that the trigger script undertakes. *type* can be one of:

- **job**: this type should be used with P4 Server **form-commit** events for job forms, and informs P4 Code Review when a job is created or modified.
- **user**: this type should be used with P4 Server **form-commit** events for user forms, which informs P4 Code Review when a user is added or modified.
- **userdel**: this type should be used with P4 Server **form-delete** events for user forms, which informs P4 Code Review when a user is deleted.
- **group**: this type should be used with P4 Server **form-commit** events for group forms, and informs P4 Code Review when a group is created or modified.
- **groupdel**: this type should be used with P4 Server **form-delete** events for group forms, and informs P4 Code Review when a group is deleted.
- **changesave**: this type should be used with P4 Server **form-save** events for change forms, and informs P4 Code Review when a changelist is created or modified.
- **shelve**: this type should be used with P4 Server **shelve-commit** events, and informs P4 Code Review when a changelist is shelved, which can create or update a review.
- **commit**: this type should be used with P4 Server **change-commit** events, and informs P4 Code Review when a changelist is committed.
- **shelvedel**: this type is used with P4 Server **shelve-delete** events, and informs P4 Code Review when a file is deleted from a shelf.

**Important:**

Do not use the P4 Code Review user that is configured in the [P4 Code Review configuration file](#) when deleting shelves, or deleting files from shelves. The P4 Code Review logic processes the *shelve-delete* trigger event, if the event is invoked by the P4 Code Review user it is rejected. The delete operations will fail.

- **enforce** comment out if the Workflow feature is [enabled](#) (enabled by default): this type is used to verify that commits to specific depot paths are associated with approved reviews. If a commit includes a file within the specified depot path, and it is not associated with a review (or a review that is not approved), the commit is rejected.

Using the **enforce** type prevents users from committing changes to specific depot paths without those changes being reviewed and approved.

- **strict** comment out if the Workflow feature is [enabled](#) (enabled by default): this type is used to verify that the file content in a commit matches the file content of its associated approved review. If one or more files in a commit do not match the content of the file in its associated review, the commit is rejected.



Using the strict type prevents users from making changes to the file content after a review has been approved and then submitting the unapproved changes.

**Note:**  
strict implies enforce.

- **Workflow trigger types** comment out if the Workflow feature is [disabled](#) (enabled by default). The enforce and strict triggers above must be commented out if the Workflow feature is [enabled](#):

**Important:**  
**Known limitations**

The workflow triggers do not support the following trigger functionality available in P4 Code Review 2018.1 and earlier:

- **EXEMPT\_FILE\_COUNT:** When set to a positive integer, commits with a file count greater or equal to this value are exempt from **enforce** or **strict** verifications.
- **EXEMPT\_EXTENSIONS:** A comma separated list of file extension. Commits with files having only these extensions are exempt from **enforce** or **strict** verifications.

To continue to use this functionality, you must keep your existing **enforce** and **strict** triggers and [disable the Workflow feature](#).

**Tip:**

By default, all group members and users must follow any workflow rules that apply to changelists and reviews. However, your P4 Code Review administrator can configure specific groups and users so that they are excluded from following the workflow rules. This applies to actions taken in the P4 Code Review UI, the P4 command-line (P4), a P4D client, and the P4 Code Review API.

The following trigger types are used by P4 Code Review for workflow rules. They are not intended for use in custom trigger scripts.

- **checkenforced**: this type is used by the P4 Code Review **On commit with a review workflow rule** and cannot be configured. **checkenforced** triggers on a P4 Server **change-submit** event.

If a commit is made to a branch that has **On commit with a review** set to **Reject unless approved**, and it is not associated with a review (or the review is not approved), the commit is rejected. If the commit is associated with an approved review, a content check is carried out by **checkstrict**.

- **checkstrict**: this type is used by the P4 Code Review **On commit without a review workflow rule** and cannot be configured. **checkstrict** triggers on a P4 Server **change-content** event.

This check is performed after **checkenforced**. If the **On commit without a review** rule is set to **Reject**, and one or more files in a commit do not match the content of the file in its associated review, the commit is rejected.

- **checkshelve**: this type is used by the P4 Code Review **On update of a review in an end state workflow rule** and cannot be configured. **checkshelve** triggers on a P4 Server **shelve-submit** event.

If a shelf is associated with a review, and it is committed to a branch that has **On update of a review in an end state** set to **Reject**, P4 Code Review checks what state the review is in to make sure it is not in a **protected state**. If the review is in a protected state, the commit is rejected.

### Important:

You cannot mix P4 Code Review trigger types with unrelated P4 Server events; the behavior is undetermined, and the information required for each type of processing may not be available to the trigger.

On Windows Perforce servers using Strawberry Perl, multithreading issues and the spawning of numerous perl.exe child processes can occur if the `swarm-trigger.pl` script is referenced from a depot path.

To mitigate these issues for non-workflow triggers, add the `-z` flag to the end of your trigger definitions. This forces Perl processes to run synchronously, bypassing potential problems with Win32 process forking. For example:

```
swarm.shelvesub shelve-submit //... "perl %//.swarm/triggers/swarm-
trigger.pl% -c %//.swarm/triggers/swarm-trigger.conf% -t checkshelve -v
%change% -u %user% -z"
```

Do not use the `-z` flag with workflow triggers. Using this flag with workflow triggers will prevent the `enforce` and `strict` code sections within the trigger script from executing correctly. Workflow triggers should be configured without the `-z` flag.

- **-v ID**

Specifies *ID*, which is the identifier for the current trigger type.

When the *type* is `job`, `user`, `userdel`, `group`, `groupdel`, or `changesave`, *ID* should be `%formname%` to specify the specific form identifier P4 Code Review should process.

When the *type* is `shelve`, `commit`, `shelvedel`, `checkenforced`, `checkstrict`, or `checkshelve` *ID* should be `%change%` to specify the specific changelist P4 Code Review should process.

- **-p port (optional) Do not use if the Workflow feature is enabled (default)**  
Specifies the P4 Server port (P4PORT). This value is optional, and is only used for types `enforce` or `strict` as the trigger has to run its own commands against the P4 Server during its processing.
- **-r (optional) Do not use if the Workflow feature is enabled (default)**  
Specifies that, when types `enforce` or `strict` are being processed, the verifications should only be performed on commits that are currently in review.
- **-g (optional) Do not use if the Workflow feature is enabled (default)**  
Specifies a group to exclude from `enforce` or `strict` verifications. Members of the group (including sub-groups) are not subject to `enforce` or `strict` verifications.
- **-c config\_file (optional)**  
Specifies an optional *config\_file* which is used to specify configuration items.

## Configuration items

The following configuration items are used in the `swarm-trigger.conf` (or another file, if the `-c` option is used in the trigger entries).

- **SWARM\_HOST (required)**  
Specifies the host URL of your P4 Code Review instance, with the leading `http://` or `https://`. For example: `https://myswarm.url`.
- **SWARM\_TOKEN (required)**  
A token used when talking to P4 Code Review. To obtain the token, log into P4 Code Review as a user with *super* privileges and select the [About P4 Code Review](#) from the user menu in the main navigation bar.  
  
You can also manually create additional tokens. Tokens are empty files stored within `SWARM_ROOT/data/queue/tokens` (the filename is the token and is reported on the **About P4 Code Review** dialog), that should be readable by the web server.  
  
You might manually create additional tokens to allow other processes to talk to P4 Code Review, such as JIRA build tasks, and to selectively invalidate access to P4 Code Review without interfering with regular P4 Code Review operations.
- **ADMIN\_USER (optional) Do not use if the Workflow feature is enabled (default)**  
When `enforce` or `strict` verifications are to be performed, you may need specify a username of a user in the P4 Server that has *admin* privileges. If you do not specify a username, the trigger script uses the P4 Server user set in the environment.
- **ADMIN\_TICKET\_FILE (optional) Do not use if the Workflow feature is enabled (default)**

When enforce or strict verifications are to be performed, you may need specify the path to the `.p4tickets` file, if your P4 Server tickets file is not the default `$HOME/.p4tickets`.

**Important:**

Ensure that the ticket belongs to a user with *admin* privileges in the P4 Server, and is a member of a group with an unlimited or very long ticket timeout. If this user's authentication times out, enforce and strict verifications stop working.

- **P4\_PORT (optional) Do not use if the Workflow feature is enabled (default)**

When enforce or strict verifications are to be performed, you may need to set the port value (P4PORT) of the P4 Server, particularly if the P4 Server is on a non-standard port, or if the P4 Server is not using the default hostname.

- **P4 (optional) Do not use if the Workflow feature is enabled (default)**

Specifies the full path to `p4`, the P4 Server command-line client. This is only required when `p4` is not found in the PATH of the P4 Server environment, and when enforce or strict verifications are to be performed.

- **EXEMPT\_FILE\_COUNT (optional) Do not use if the Workflow feature is enabled (default)**

When set to a positive integer, commits with a file count greater or equal to this value are exempt from enforce or strict verifications.

- **EXEMPT\_EXTENSIONS (optional) Do not use if the Workflow feature is enabled (default)**

A comma-separated list of file extensions. Commits with files having only these extensions are exempt from enforce or strict verifications.

- **VERIFY\_SSL (optional)**

If HTTPS is used on the P4 Code Review web server, this controls whether the SSL certificate is validated or not.

By default `VERIFY_SSL` is set to `1`, this means any SSL certificates must be valid. If the P4 Code Review web server is using a self signed certificate, `VERIFY_SSL` must be set to `0`.

For more information about the risks of using a self-signed SSL certificate, see "[Security risks of using a self-signed certificate](#)" on page 287.

- **TIMEOUT (optional)**

Sets the number of seconds to wait before returning an error when communicating with the P4 Code Review server.

The default setting is `30` seconds.

- **IGNORE\_TIMEOUT (optional)**

Configure the action to take if communication with the P4 Code Review server times out. For example, if you are submitting a large number of files to the P4 Server.

Options are:

- **0** if communication with the P4 Code Review server times out, the submit will fail. This is the default setting.
- **1** if communication with the P4 Code Review server times out, the submit will continue. This increases the reliability of submits.
- **IGNORE\_NOSERVER (optional)**  
Configure the action to take if communication with the P4 Code Review server fails. For example, the P4 Code Review server is down.  
Options are:
  - **0** if communication with the P4 Code Review server fails, the submit will fail. This is the default setting.
  - **1** if communication with the P4 Code Review server fails, the submit will continue. This increases the reliability of submits.

## Unapprove modified reviews

By default, when an approved review is committed or updated, P4 Code Review changes the state to **Needs Review** if the files have been modified since the review was approved.

If one or more files in a review has the filetype +k (**ktext**), this behavior is undesirable because the files will appear to be modified as the P4 Server replaces RCS keywords with their current values. This behavior can be disabled.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

To disable this behavior, edit the [SWARM\\_ROOT/data/config.php](#) file, and add or update the `unapprove_modified` item to `false`, within the reviews configuration block. For example:

```
<?php
// this block should be a peer of 'p4'
'reviews' => array(
 'unapprove_modified' => false,
),
```

For information on file types, see [File Types](#) in [P4 CLI Reference](#).

## Uninstall P4 Code Review

This section covers the steps required to uninstall P4 Code Review.

## Background

The bulk of P4 Code Review's metadata (activity, comments, review records, followers) is stored in p4 keys under `swarm-*`. If you are using a 2012.1+ server, P4 Code Review also defines user groups for each project that you define. The names of these groups correspond 1-to-1 with projects, for example `swarm-project-fantastico`. P4 Code Review manages a pool of client workspaces that it uses to shelf and commit files. These clients are named `swarm-{uuid}`, for example `swarm-5ad4a9c0-06e7-20eb-897f-cbd4cc934295`.

## Uninstall steps

- Depending on how you have configured your Helix Core Server events, do one of the following:
  - Uninstall the P4 Server extension. To do this, run the following command on your P4 Server:
 

```
p4 extension --delete Perforce::helix-swarm --yes
```
  - Uninstall the P4 Code Review triggers. As a *super* user, run the `p4 triggers` command from your P4 Server and manually remove all the P4 Code Review trigger code lines.
- Remove your web server's virtual host configuration for P4 Code Review by disabling the P4 Code Review site (`perforce-swarm-site`). If you have made any non-standard changes for P4 Code Review such as SSL configuration or if an alternate web server is used, you must remove these changes for P4 Code Review.

Depending on your OS distribution, do one of the following:

- Ubuntu** (run the following command as root):
    - Disable the Swarm site:
 

```
a2dissite perforce-swarm-site
```
    - Delete the Swarm site:
 

```
Delete /etc/apache2/sites-available/perforce-swarm-site.conf
```
  - RHEL/Amazon Linux 2** (run the following command as root):
 

Delete the Swarm site:

```
Delete /etc/httpd/conf.d/perforce-swarm-site.conf
```
- Restart your web server.
  - Delete groups/clients/keys that are prefixed with `swarm-*`.

### Note:

The clients could contain shelved files for reviews. Determine how you want to handle those files prior to deleting the clients.

- Additional indexed information is stored in the database file `db.ixtext`. Unfortunately, indexed jobs and other generic indexed information would be lost if this table was simply removed, and modifying the database file can be a dangerous operation in a number of P4 Server deployment scenarios.

**Important:**

Contact [Perforce support](#) for assistance if you feel the need to remove P4 Code Review's indexed information.

6. Rebuild the job index. The best approach is to run:

```
$ p4 jobs -R
```

which rebuilds the `db.ixtext` table. There are two caveats that likely require discussion with [Perforce support](#):

- If you make use of the unsupported `p4 index` command, you **cannot** use this approach, as it would remove all of your indexes.
- If you have indexing turned on for the domain table, you must also run:

```
$ p4d -xf index.domain.owner
```

7. If the `P4.Swarm.URL` or `P4.Swarm.CommitURL` properties were set (for details, see "[Client integration](#)" on page 573 and "[Commit-edge deployment](#)" on page 582 respectively), they should be unset to prevent P4V (and potentially other clients) from attempting P4 Code Review operations. We recommend you view all the properties and delete them all.

To delete each P4 Code Review property:

```
$ p4 property -d -n <P4.Swarm property> -s <sequence>
```

The sequence is provided in the property list. A property with `#none` is sequence 0.

To discover all of the definitions of the `P4.Swarm.*` properties and their sequence numbers, as a Perforce super user run the following command:

```
$ p4 property -AI -n P4.Swarm
```

For example, the following property values are returned that apply to all users and groups:

```
$ p4 property -AI -n P4.Swarm
```

```
P4.Swarm.CommitURL = https://myswarm.url/ (any) #none
```

```
P4.Swarm.CommitURL = https://myswarm1.url/ (any) #1
```

```
P4.Swarm.URL = https://myswarm.url/ (any) #none
```

```
P4.Swarm.URL = https://myswarm3.url/ (any) #3
```

```
P4.Swarm.URL = https://myswarm2.url/ (any) #2
```

```
P4.Swarm.URL = https://myswarm1.url/ (any) #1
```

To delete all of these property definitions, you would run the following commands:

```
$ p4 property -d -n P4.Swarm.CommitURL -s0
```

```
$ p4 property -d -n P4.Swarm.CommitURL -s1
```

```
$ p4 property -d -n P4.Swarm.URL -s0
```

```
$ p4 property -d -n P4.Swarm.URL -s3
```

```
$ p4 property -d -n P4.Swarm.URL -s2
$ p4 property -d -n P4.Swarm.URL -s1
```

To verify if all the properties have been deleted, run the following command as a Perforce super user:

```
$ p4 property -AI P4.Swarm
```

## Upgrade index

The P4 Code Review Upgrade index process can be configured to suit the performance of your P4 Code Review system via the following config items.

### Note:

If you are upgrading from P4 Code Review version 2017.3 or later, the index upgrade step is not required.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## status\_refresh\_interval

The `status_refresh_interval` sets the refresh rate of the index upgrade status page. The default is 10 seconds.

```
<?php
// this block should be a peer of 'p4'
'upgrade' => array(
 'status_refresh_interval' => 10, //Refresh page every 10 seconds
)
```

## batch\_size

The `batch_size` sets the number of reviews held in memory and processed for each batch. This value can be increased or decreased depending on your P4 Code Review machines system memory. The default is 1000 reviews.

### For example:

If the P4 Code Review machine does not have a lot of memory, for instance 128MB, a `batch_size` of 1000 might be too high and may make the upgrade run very slowly. In this instance reduce the `batch_size` to a more manageable size.



```
<?php
// this block should be a peer of 'p4'
'upgrade' => array(
 'batch_size' => 1000, //Fetch 1000 reviews to lower memory usage
),
```

## Users

This section describes the configurables that can be set for users.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## Dashboard refresh interval

By default, when your dashboard is open, P4 Code Review checks for new content every five minutes. If P4 Code Review finds new content for a user's dashboard, the **Refresh** button is displayed on their dashboard. For more information about using the **Refresh** button, see [Refresh button](#).

Configure the dashboard refresh interval in milliseconds with the following configuration block in the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
// this block should be a peer of 'p4'
'users' => array(
 'dashboard_refresh_interval' => 300000, // Default 300000 milliseconds
),
),
```

Default value is 300000 milliseconds (300 seconds, 5 minutes)

## Display full names

**Note:**

For P4 Code Review 2022.1 and later, the `display_fullname` value defaults to true. For earlier P4 Code Review releases, the default was false. Upgrading your P4 Code Review version to 2022.1 and later will not change your `display_fullname` setting.

By default, P4 Code Review displays full user names. Set the `display_fullname` configurable to `false` to display *userids* on the:

- [Project Settings](#) page
- [Group Settings](#) page
- [Activity streams](#)
- [Commits](#) page
- [Edit reviewers](#) dialog

To display *userid*s in P4 Code Review, set the following configuration block in the [SWARM\\_ROOT/data/config.php](#) file to false:

```
<?php
// this block should be a peer of 'p4'
'users' => array(
 'display_fullname' => false,
),
),
```

Default value is true, full user names are displayed.

## Review preferences

The review preferences configure the default way that [diffs](#) are initially displayed to users when they view them in changelists and code reviews. The review preference settings are used for a user until they change their user preferences, see user profile [Settings](#).

Configure the default user diff view settings with the following configuration block in the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
// this block should be a peer of 'p4'
'users' => array(
 'settings' => array(
 'review_preferences' => array(
 'show_comments_in_files' => true,
 'view_diffs_side_by_side' => true,
 'show_space_and_new_line_characters' => false,
 'ignore_whitespace' => false,
),
),
),
```

[show\\_comments\\_in\\_files](#):

- true: display comments in files and in the **Comments** tab. This is the default value.
- false: display comments only in the **Comments** tab.

[view\\_diffs\\_side\\_by\\_side](#):

- true: display diffs in files side by side in two panes. This is the default value.
- false: display diffs in files inline in a single pane.

show\_space\_and\_new\_line\_characters:

- `true`: display whitespace characters as dots, tabs as arrows that point to a bar, and line endings as arrows that point down.
- `false`: whitespace, tabs and line endings are not displayed as special characters. This is the default value.

ignore\_whitespace:

- `false`: changes made to whitespace are highlighted in file diffs. This makes it easier to identify changes in file types where whitespace is important. This is the default value.
- `true`: changes made to whitespace are not highlighted in file diffs. This makes it easier to see the important changes in file types where whitespace changes are not important.

**Note:**

`ignore_whitespace` replaces `ignore_whitespace_default` that was used in P4 Code Review version 2017.4 and earlier. By default `ignore_whitespace` is set to `false`, this is the same original default value set for `ignore_whitespace_default`.

## Time preferences

The time preferences configure the default way that time is initially displayed to users. The time preference settings are used for a user until they change their user preferences, see user profile [Settings](#).

Configure the default user time settings with the following configuration block in the [SWARM\\_ROOT](#)/data/config.php file:

```
<?php
// this block should be a peer of 'p4'
'users' => array(
 'settings' => array(
 'time_preferences' => array(
 'display' => 'Timeago', // Default to 'Timeago' but can be set to 'Timestamp'
),
),
),
```

display:

- `'Timeago'`: time is displayed as how long ago the event happened. This is the default value.
- `'Timestamp'`: time is displayed as the date and time the event happened using the local machine browser time.

**Tip:**

If timestamp is selected, the date format used by P4 Code Review matches the date format configuration of the local machine browser.

## How P4 Code Review handles deleted users

P4 Code Review's users are based on the users configured in the P4 Server. When a user is deleted from the P4 Server, P4 Code Review automatically removes the deleted user from the following locations:

- ["User profile" below](#)
- ["Test definitions" below](#)
- ["Workflows " below](#)
- ["Projects " on the facing page](#)
- ["Groups" on page 738](#)

A user who has been deleted from P4 Code Review can be identified by the suffix "deleted" at the end of their userid. This only works for ["My Dashboard" on page 288](#), ["Reviews list" on page 406](#), ["Project settings" on page 478](#), and ["Review display" on page 411](#) pages that are written in React. An event is added to the activity stream whenever a deleted user is removed from workflows, projects, groups, test definition, and project followers.

For more information how P4 Code Review handles users deleted outside of the platform, see ["When an owner is deleted outside of P4 Code Review" on page 506](#).

### User profile

When a user is deleted from the P4 Server, P4 Code Review automatically performs the following actions on the user profile:

- Removes the profile of the deleted user from the `SWARM_ROOT/data/config.php` file.
- Removes the profile of the deleted user from the Perforce database.
- Removes all the followers of the user that is deleted.
- Removes the deleted user from the followers list of any user or projects they might be following.

### Test definitions

When a user is deleted from the P4 Server, P4 Code Review will also remove the user from the test definition ownership. P4 Code Review handles the following two scenarios for test definitions:

- **Scenario 1:** If the user who was deleted is the sole owner of a test definition, on user deletion, the P4 Code Review user mentioned in the `SWARM_ROOT/data/config.php` file will become the owner of the test definition.
- **Scenario 2:** If there are two owners for a test definition, on user deletion, the other user automatically becomes the owner of the test definition.

### Workflows

When a user is deleted from the P4 Server, P4 Code Review will also remove the user from any workflow ownership. P4 Code Review handles the following two scenarios for workflows:

- **Scenario 1:** If the user who was deleted is the sole owner of a workflow, on user deletion, the P4 Code Review user mentioned in the `SWARM_ROOT/data/config.php` file will become the owner of the workflow.
- **Scenario 2:** If there are two owners for a workflow, on user deletion, the other user automatically becomes the owner of the workflow.

## Projects

When a user is deleted from the P4 Server, P4 Code Review will also remove the user from any associated projects that the user is part of. A project must have at least one member or a subgroup.

P4 Code Review handles the following scenarios for projects:

- **Scenario 1:** If the user who was deleted is the sole member of a project, on user deletion, the P4 Code Review user mentioned in the `SWARM_ROOT/data/config.php` file will become the member of the project.
- **Scenario 2:** If there are only two members for a project, on user deletion, the other user becomes the sole member the project.
- **Scenario 3:** If the user who was deleted is the sole member and owner of a project, on user deletion, the P4 Code Review user mentioned in the `SWARM_ROOT/data/config.php` file will become the member of the project.
- **Scenario 4:** If there are only two members for a project and the user who was deleted is the sole owner of the project, on user deletion, the other user becomes the sole member of the project.
- **Scenario 5:** If there are only two members for a project of which one user is the owner of the project and the other user is deleted, the P4 Code Review user mentioned in the `SWARM_ROOT/data/config.php` file will become the member of the project.

The table below outlines all the scenarios after the deletion of the **Test\_User** from the P4 Server. This is the default behavior.

Scenario	Members	Owners	Members post deletion	Owners post deletion
1	Test_User		P4 Code Review	
2	Test_User, Bruno		Bruno	
3	Test_User	Test_User	P4 Code Review	

Scenario	Members	Owners	Members post deletion	Owners post deletion
4	Bruno	Test_User	Bruno	
5	Test_User	Bruno	P4 Code Review	Bruno

### Private projects

When a user is deleted from the P4 Server, P4 Code Review handles the following scenarios for private projects:

- **Scenario 1:** If the user who was deleted is the sole owner of the private project, on user deletion, the P4 Code Review user mentioned in the [SWARM\\_ROOT/data/config.php](#) file will become the owner of the project.
- **Scenario 2:** If a private project has more than one owner, on user deletion, P4 Code Review removes the deleted user.

### Groups

When a user is deleted from the P4 Server, P4 Code Review will also remove the user from any group that the user is part of. Groups can only be modified by group owners or super users. Therefore, if the P4 Code Review user is not the owner of the group or a super user, the group cleanup activity will not occur. In such cases, P4 Code Review logs a message indicating that it was unable to perform the clean-up due to the Swarm user permissions being limited to admin and not having owner or super user access.

P4 Code Review handles the following scenarios for groups:

- **Scenario 1:** If a P4 Code Review user deletes a sole group owner without having owner or super user permissions, P4 Code Review logs a message indicating that it was unable to perform the clean-up due to the limited permissions of the P4 Code Review user.
- **Scenario 2:** If a P4 Code Review user with owner or super user permissions deletes a sole group owner, the group owner is removed and the P4 Code Review user mentioned in the [SWARM\\_ROOT/data/config.php](#) file becomes the owner of the group.
- **Scenario 3:** If a group has two owners, 'Test\_User' and 'P4 Code Review', and a P4 Code Review user deletes owner 'Test\_User' without having owner or super user permissions, the 'Test\_User' owner is removed and P4 Code Review becomes the sole group owner.
- **Scenario 4:** If a P4 Code Review user deletes the sole member of a group without having owner or super user permissions, P4 Code Review is unable to remove the member from the group due to the limited permissions of the P4 Code Review user.
- **Scenario 5:** If a P4 Code Review user with owner or super user permissions deletes the sole member of a group, the P4 Code Review user mentioned in the [SWARM\\_ROOT/data/config.php](#) file will become the member of the group.

- **Scenario 6:** If a group has only one member and one owner, and a P4 Code Review user without owner or super user permissions deletes the sole member of the group, the deleted user is removed from the group.
- **Scenario 7:** If a group has two members, 'Test\_User' and 'Bruno', and a P4 Code Review user without owner or super user permissions deletes one member of the group, P4 Code Review is unable to remove the member from the group due to the limited permissions of the P4 Code Review user.
- **Scenario 8:** If a group has two users, 'Test\_User' and 'Bruno', and a P4 Code Review user with owner or super user permissions deletes the 'Test\_User' from the group, Swarm removes the 'Test\_User' from the group and 'Bruno' becomes the sole member of the group.
- **Scenario 9:** If a group has two users, 'Test\_User' and 'Bruno', and a sole group owner, P4 Code Review, and if a P4 Code Review user with owner or super user permissions deletes the 'Test\_User' from the group, Swarm removes the 'Test\_User' from the group and 'Bruno' becomes the sole member of the group. The group owner remains unchanged.

The table below outlines all the scenarios after the deletion of the **Test\_User** from the P4 Server. This is the default behavior.

Scenario	Members	Owners	P4 Code Review user is a super user or not	Members post deletion	Owners post deletion
1		Test_User	No		Test_User
2		Test_User	Yes		P4 Code Review
3		Test_User, P4 Code Review	No		P4 Code Review
4	Test_User		No	Test_User	
5	Test_User		Yes	P4 Code Review	

Scenario	Members	Owners	P4 Code Review user is a super user or not	Members post deletion	Owners post deletion
6	Test_User	P4 Code Review	No		P4 Code Review
7	Test_User, Bruno		No	Test_User, Bruno	
8	Test_User, Bruno		Yes	Bruno	
9	Test_User, Bruno	P4 Code Review	No	Bruno	P4 Code Review

## Workers

P4 Code Review uses background processes called *workers* to respond to events in the P4 Server. The default number of workers is three, and each worker waits to process events for up to 10 minutes before terminating. When a worker terminates, a new one is spawned.

A worker processes the longest-pending task from the queue and triggers an event for that task. Each event has two parameters:

- **id** – The unique identifier of the task
- **time** – The timestamp of the event.

The name of the event is `task.type` (e.g. `task.change`).

Immediately after processing a task, a worker attempts to process another task from the queue. If there are no task in the queue, the worker sleeps for one second and then checks the queue for another task.

When the worker's running time exceeds their lifetime, it terminates to avoid potential memory leakage. By default, a worker's lifetime is 10 minutes . To adjust this lifetime, see "[Worker configuration](#)" on the facing page.

A given task can run for a long time and consume as much memory as is available (to accommodate large changes for instance). Workers run in the background by default.

Each worker maintains a connection to the P4 Server for the duration of its lifetime. This may impact your P4 Server management practices.



**Tip:** To view P4 Code Review task queue information, as an *admin* or *super* user, navigate to the **User id** dropdown menu, select **System Information**, and open the ["Queue Info" on page 715](#) tab. If the task queue starts to build up, consider increasing the number of workers.

## Worker status

To determine the current status of workers, do one of the following:

- As an *admin* or *super* user, navigate to the **User id** dropdown menu, select **System Information**, and open the ["Queue Info" on page 715](#) tab.

or

- Visit the URL: <https://myswarm.url/queue/status>.

The response is formatted in JSON, and looks like this:

```
{"tasks":0,"futureTasks":1,"workers":3,"maxWorkers":3,"workerLifetime":"595s"}
```

During normal use of P4 Code Review, the following error message appears for logged-in users when P4 Code Review detects that no workers are running:

Hmm... no queue workers? Ask your administrator to check the [worker setup](#).

## Worker configuration

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

To adjust the configuration for workers, add a configuration block to the [SWARM\\_ROOT/data/config.php](#) file:

```
<?php
// this block should be a peer of 'p4'
'queue' => array(
 'workers' => 3, // defaults to 3
 'worker_lifetime' => 595, // defaults to 10 minutes (less 5 seconds)
 'worker_task_timeout' => 1800, // defaults to 30 minutes
 'worker_memory_limit' => '1G', // defaults to 1 gigabyte
),
```

where:

- `workers` specifies the number of worker processes that should be available. The default is 3. The [cron job](#) ensures that new worker processes are started when necessary. If the limit is reached or exceeded, new worker processes are not started. Workers do not communicate with each other. If the task queue starts to build up, consider increasing the number of workers.
- `worker_lifetime` specifies the amount of time in seconds that a worker process should run for. The default is 595 seconds (10 minutes less 5 seconds). If a worker process exceeds this limit while processing a task, it will complete the active task and then terminate. `worker_lifetime` does not cause tasks to terminate mid-processing.
- `worker_task_timeout` specifies the maximum amount of time in seconds that a worker process can spend processing a single task. The default is 1800 seconds (30 minutes). This is useful for terminating workers that might get stalled in a variety of situations.
- `worker_memory_limit` specifies the maximum amount of memory that a worker process is allowed to use while processing a task. The default is 1G (1 gigabyte).

## Manually start workers

To kick off a new worker process, do one of the following:

- As an *admin* or *super* user, navigate to the **User id** dropdown menu, select **System Information**, open the ["Queue Info" on page 715](#) tab, and click **Start a worker**.
- or
- Visit the URL: <https://myswarm.url/queue/worker>.

If the number of workers running matches the configured limit, the requested worker process is not started.

**Note:** This technique does start a worker, but it lasts only for its configured lifetime. Typically, you would always want at least one worker running. See ["Set up a recurring task to spawn workers" on page 208](#) for details.

## Manually restart workers

To restart an idle worker process, remove its lock file:

```
rm data/queue/workers/worker_id
```

A worker process that is busy processing a task will continue operation until its task is complete. Immediately afterwards, if the worker notices that its lock file is missing it exits.

If you have a recurring task to start workers, the recurring task starts a fresh worker, if necessary. See ["Set up a recurring task to spawn workers" on page 208](#) for details.

## Workflow global rules

This section provides information on how to enable workflow and configure the global workflow rules for P4 Code Review.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

## Workflow prerequisites

For P4 Code Review 2019.2 and later, the workflow feature is enabled by default.

**Tip:**

If you are upgrading from an earlier version you will need to update your triggers, see ["Upgrading P4 Code Review"](#) on page 231.

**The workflow feature requires:**

- Workflow must be enabled, see the ["workflow"](#) below system configurable.
- Your `swarm-trigger.conf` file must be configured correctly, see ["Setup up P4 Code Review triggers"](#) on page 188.
- Your P4 Server trigger table must contain the workflow trigger lines, see ["Setup up P4 Code Review triggers"](#) on page 188.

## workflow

From P4 Code Review 2019.2, workflow is enabled by default.

To disable the workflow feature, edit the `SWARM_ROOT/data/config.php` file and set the `enabled` configurable to `false` in the `workflow` section of the codeblock:

```
'workflow' => array(
 'enabled' => false, // Switches the workflow feature on. Default is true
),
```

The default value is `true`.

**Tip:**

If you disable the workflow feature in the P4 Code Review `config.php` file, workflows are not processed by P4 Code Review but a small overhead is still incurred by the P4 Server each time it runs a workflow trigger script. This overhead can be eliminated by commenting out the `swarm.enforce change-submit`, `swarm.strict change-content`, and `swarm.shelvesub shelve-submit` workflow triggers.

## Global workflow

**Note:**

- The **Global Workflow** page can be edited by a P4 Code Review user with *super* or *admin* privileges, or users that have been made an owner.
- In earlier versions of P4 Code Review, global workflow rules were set in the P4 Code Review config.php file. From P4 Code Review 2020.1, they are set in the global workflow on the P4 Code Review **Workflows** page. If you upgrade from an earlier version of P4 Code Review, your global workflow settings are automatically migrated to a workflow called **Global Workflow**.
- In earlier versions of P4 Code Review, global tests were set in the P4 Code Review config.php file. From P4 Code Review 2020.1, tests are added to the global workflow on the P4 Code Review **Workflows** page so that they operate as global tests. If you upgrade from an earlier version of P4 Code Review, your global tests are automatically migrated. Each global test is migrated as follows: the owner is set as the P4 Code Review admin user (not shared), the name is set to the original test name, the description is set to the original test title, and it is added to the **Global Workflow**.

Global workflow rules enable you apply rules globally if required, ensuring basic company policies are followed by every project. Each rule in the global workflow can be set with Enforce off (default) or Enforce on, this determines how that rule is used by P4 Code Review:

- **Enforce off:** applies the workflow rule setting to projects and project branches that don't have an associated [workflow](#). If a project or project branch has an associated P4 Code Review workflow, the global rule is ignored.
- **Enforce on:** applies a minimum workflow rule setting to all projects and project branches. If a project or project branch has an associated [workflow](#), the global rule is merged with the workflow rule and the most restrictive setting is used.

**To edit the global workflow rules:**

1. On the P4 Code Review **Workflows** page.
2. Click **Global Workflow** to open the settings page:

**Tip:**

By default, the global workflow is called **Global Workflow**, but it can be changed. To help you identify the global workflow it is always shown as the first workflow in the list and it has a different background color to the other workflows.

Global Workflow x

Name

Description

Owners

Individuals:  
swarm

Global Rules ⓘ

On commit without a review
Allow

On commit with a review
Reject unless approved

On update of a review in an end state
Reject

Count votes up from
Anyone

Automatically approve reviews
Never

Enforce ⓘ

☐
☒
☒
☐
☐

Members or groups who can ignore ALL workflow rules

Groups:  
swarm-test

Tests ⓘ

	When	Blocks	
Code sniff	On Update	Nothing	
Build tests	On Submit	Nothing	
Security check	On Update	Approve	
<a href="#">+Add Test</a>			

Save

Cancel

- Optional:** change the name if required, we recommend that you leave it set to **Global Workflow** if possible for ease of identification.
- Optional:** provide a description for the global workflow.
- Optional:** add more owners if required. This field auto-suggests groups, and users within P4 Server as you type (up to a combined limit of 20 entries).

### Important:

- The global workflow must have at least one owner.
- If you remove yourself as an owner, you cannot edit this workflow configuration later unless you have *super* user rights.

### 6. Global Rules:

Each rule in the global workflow can be set with Enforce off (default) or Enforce on using the **Enforce** slider next to the rule, this determines how that rule is used by P4 Code Review:

- Enforce off:** applies the workflow rule setting to projects and project branches that don't have an associated [workflow](#). If a project or project branch has an associated P4 Code Review workflow, the global rule is ignored.
- Enforce on:** applies a minimum workflow rule setting to all projects and project branches. If a project or project branch has an associated [workflow](#), the global rule is merged with the workflow rule and the most restrictive setting is used.

**Important:**

- Changes to global workflow rules that are set with **Enforce off** will change the workflow for projects and project branches that do not have an associated workflow.
- Changes to global workflow rules that are set with **Enforce on** will change the workflow for all projects and project branches.

**On commit without a review:**

This rule is applied when a changelist without an associated review is submitted from outside of P4 Code Review.

Select one of the following options:

- **Allow:** the changelist is not checked, the changelist is submitted without a review.
- **Create a review:** the changelist is submitted and a review is created automatically for the changelist.
- **Reject:** the changelist submit is rejected.

**Tip:**

The selected rule is also applied when a changelist is submitted with **#review** in the description. For more information about creating a review by including a keyword in the changelist description, see ["Create a review" on page 669](#).

**On commit with a review:** This rule is applied when:

- A review is committed.
- A changelist with an associated review is submitted from outside of P4 Code Review.

Select one of the following options:

- **Allow:** the changelist review state is not checked. The changelist is committed even if its associated review is not approved.
- **Reject unless approved:** the changelist is only submitted if its associated review is approved and the content of the changelist is identical to the content of the approved review.

**Tip:**

The selected rule is also applied when a changelist is submitted with **#review-nnnnn**, **#replace-nnnnn**, or **#append-nnnnn** in the description (nnnnn = review ID). For more information about adding a changelist to a review by including a keyword in the changelist description, see ["Add a changelist to a review" on page 670](#).

**On update of a review in an end state:**

Used to stop review content being automatically changed for reviews that are in specific states. By default, the protected end states are **Archived**, **Rejected**, and **Approve:Commit**. The end states are set by the P4 Code Review administrator, see [end\\_states](#).

**Tip:**

When a review is in a protected end state, it can still be updated manually by a user from the P4 Code Review UI.

This rule is applied when a changelist is added to a review.

Select one of the following options

- **Allow:** the review is not checked. The changelist is added to the review no matter what the review state is. This is the default setting.
- **Reject:** if the review is in one of the states specified in the `end_state` configurable:
  - The **Add Change** button is disabled for the review.
  - The changelist is rejected if it is added outside of P4 Code Review using `#review-nnnnn`, `#replace-nnnnn`, or `#append-nnnnn` in the description (nnnnn = review ID).

**Count votes up from:**

By default, all of the up votes on a review are counted for the **Minimum up votes** value set on the project/branch the review is associated with. Limit the up votes that are counted to just the members of the project the review is associated with by using this rule.

This rule is applied when a user votes on a review.

Select one of the following options:

- **Anyone:** votes are counted for all reviewers on a review. This is the default setting.
- **Members:** only the up votes of members of the project the review is associated with are counted for the **Minimum up votes** set on projects/branches.

**Tip:**

For instructions on how to set **Minimum up votes** for projects and branches, see [Project minimum up votes](#) and [Branch minimum up votes](#).

**Automatically approve reviews:**

By default, reviews must be manually approved. Enable automatic approval of reviews with this rule.

This rule is applied when a user votes on a review.

Select one of the following options:

- **Never:** reviews are not automatically approved. This is the default setting.
- **Based on vote count:** reviews are automatically approved if:
  - There are no down votes on the review.
  - There are no moderators on the review. If a review has moderators it cannot be automatically approved.
  - All of the [Required reviewers](#) on the review have voted up.
  - The **Minimum up votes** on the review has been satisfied for **each** of the projects and branches the review spans.
  - All of the tests configured to block approval have passed.

**Important:**

Moderators prevent the automatic approval of reviews. For more information about moderators, see ["Moderators" on page 455](#).

**Tip:**

After a review has been automatically approved it needs to be manually committed.

**Members or groups who can ignore ALL workflow rules**

This field auto-suggests users, and groups within P4 Server as you type (up to a combined limit of 20 entries).

Typically, these permissions are given to a small set of trusted users and groups, for example project owners and administrators.

**Tip:**

This applies to actions taken in the P4 Code Review UI, the P4 command-line (P4), a P4D client, and the P4 Code Review API.

**7. Tests**

**Optional:** add tests to the workflow, the tests are run when a review associated with the workflow is either created/updated or submitted. Selected from the **When** dropdown for the test.

**Tip:**

- Tests are only saved on the workflow when you click the **Save** button.
- If a review is associated with multiple workflows, all of the tests on all of those workflows are run for the review. If the same test is on more than one of the associated workflows, it is only run once.



**Add a test:**

- a. Click **+Add Test** in the Tests area.
- b. Select the test to run from the **Tests** dropdown list.
- c. Select when you want the test to run from the **When** dropdown list.

**Tip:**


- Review content change is when the files or content of files in a review change. A change to the review description does not trigger a test.
- **On Demand** tests are unaffected by the review content change check.

- **On Update** the test will run when:



- a review is created.
- a review is submitted and the review content has changed compared to the previous review version.
- a review is updated and the review content has changed since the previous review version.

**Tip:**

If the review content has not changed since a test last successfully reported a pass, it will not be re-run when a review is versioned or submitted. If a test is still running or has not updated P4 Code Review with a pass state, it will be re-run, even on a submit.

- **On Submit** the test will run when a pre-commit review is submitted or a post-commit review is created.
  - **On Demand** the test will not run automatically, but you can run it manually from the ["Tests" on page 428](#) dialog on the review page.
- d. Select whether P4 Code Review blocks approval if the test fails from the **Blocks** dropdown list:
    - **Nothing** if the test fails it will not block approval.
    - **Approve** if the test fails it will prevent the review from being approved.
  - e. Click the **Accept**  button to add the test to the workflow.

**Edit a test on the workflow:**

- a. Click the **Edit**  button next to the test you want to edit.
- b. The **Test**, **When**, and **Blocks** dropdown lists are displayed for the test.
- c. Make changes as required.
- d. Click the **Accept**  button to confirm your changes.

**Remove a test from the workflow:**

Click the **Delete**  button next to the test to remove it from the workflow.

The test is removed immediately, no confirmation is requested. You must save the workflow to complete the test removal.

8. Click **Save**.

**Note:**

The **Save** button is disabled if any required fields are empty.

# 12 | Extending P4 Code Review

P4 Code Review is built with open source technologies, using modern development methods, resulting in a platform that is capable, modular, and can be extended in a variety of ways. While this section is currently incomplete, it does identify the technologies that make up P4 Code Review and provides notes regarding some of the available extension points.

**In this section:**

## Resources

P4 Code Review is built with a number of different technologies. As such it can be a bit of a challenge to ramp up on if you haven't done web development before. This section lists some of the best resources we've found for learning how to develop for P4 Code Review.

### jQuery

A free three hour course on jQuery basics. Highly recommended if you haven't used jQuery before. It also covers some JavaScript and CSS basics:

[jQuery Learning Center](#)

### JavaScript resources

A free, in-browser JavaScript course:

[W3 Schools JavaScript Tutorial](#)

### PHP resources

A free PHP tutorial:

[W3 Schools PHP Tutorial](#)

### Laminas Framework resources

**Note:**

P4 Code Review supports the Laminas component versions in the LICENSE.txt file, features and functions in the Laminas documentation that were introduced in later versions of Laminas will not work with P4 Code Review. The LICENSE.txt file is in the readme folder of your P4 Code Review installation.

- The documentation overview page for Laminas Framework:

[Laminas documentation](#)

- Laminas tutorial that covers most of the basics:

[Laminas component tutorials](#)

- Laminas Quick Start documentation:

[Laminas Quick Start documentation](#)

## Development mode

P4 Code Review has a *development* mode that, when enabled, changes P4 Code Review's behavior in the following ways:

- CSS and JavaScript code is not aggregated. Each CSS or JavaScript file is fetched individually, which makes it much easier to identify where styles or functions exist and how they apply to P4 Code Review. Due to the additional HTTP requests, development mode should not be used in production environments.
- The ["My Dashboard" on page 288](#), ["Reviews list" on page 406](#), ["Project settings" on page 478](#), and ["Review display" on page 411](#) pages are written in React. These pages are minified and cannot be modified.
- Errors and exceptions that may occur are displayed in P4 Code Review's UI. This is particularly useful to developers of P4 Code Review modules. As the error information might disclose system paths or configuration, development mode should not be used in production environments.

**Tip:** If you make a configuration change, P4 Code Review will not use it until the configuration cache has been reloaded, this forces P4 Code Review to use the new configuration.

You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

## To enable development mode:

Adjust the `SWARM_ROOT/data/config.php` file to include:

```
<?php
return array(
 'p4' => ...
 'environment' => array(
 'mode' => 'development'
)
);
```

## To disable development mode:

Adjust the `SWARM_ROOT/data/config.php` file to exclude the `'mode' => 'development'` line.

## Modules

You can design and implement your own custom P4 Code Review modules to modify the behavior of P4 Code Review. This section covers the design and implementation of P4 Code Review custom modules.

### Important:

The ["My Dashboard" on page 288](#), ["Reviews list" on page 406](#), ["Project settings" on page 478](#), and ["Review display" on page 411](#) pages are written in React. These pages cannot be customized. If you already have custom modules for these pages and you upgrade P4 Code Review, the custom modules will not work.

If you have custom modules for any of these pages, you can send them to the P4 Code Review team to see if there is enough demand to productize them and include them in a future release. [Submit a Support request](#) to request productization of a custom module.

### Important:

The operation and testing of custom modules is the responsibility of the module creator.

Perforce Software, Inc. is not responsible for the operation of your custom modules, nor does Perforce Software, Inc. make any representations or warranties regarding the interoperability of your custom modules with P4 Code Review.

### Tip:

You must test your custom modules on a test system before transferring them to your production system. This avoids any negative impact on the operation of your production system. If you have more than one custom module, the modules should all be tested at the same time on the same test system as this ensures that the modules operate correctly with each other and with P4 Code Review.

### Tip:

If you add or edit a module, P4 Code Review will not use that change until the config cache has been reloaded, this forces P4 Code Review to use the module change. You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

**In this section:**

## Module overview

### Important:

The operation and testing of custom modules is the responsibility of the module creator.

Perforce Software, Inc. is not responsible for the operation of your custom modules, nor does Perforce Software, Inc. make any representations or warranties regarding the interoperability of your custom modules with P4 Code Review.

**Tip:**

You must test your custom modules on a test system before transferring them to your production system. This avoids any negative impact on the operation of your production system. If you have more than one custom module, the modules should all be tested at the same time on the same test system as this ensures that the modules operate correctly with each other and with P4 Code Review.

**Tip:**

If you add or edit a module, P4 Code Review will not use that change until the config cache has been reloaded, this forces P4 Code Review to use the module change. You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button" on page 720](#).

A P4 Code Review custom module is created in a folder in the `modules` folder of your P4 Code Review installation. The folder name must match the custom module name and must contain at least a `Module.php` file and a `module.config.php` file. A custom module will not work in P4 Code Review until you enable it. Enable your custom modules in P4 Code Review by adding them to the `custom.modules.config.php` file in the `config` directory (a peer of the `modules` directory).

This chapter provides basic information about how custom modules integrate with P4 Code Review and assumes a basic knowledge of the Laminas framework. For more information about the Laminas framework, see [Laminas Quick Start documentation](#).

**This section includes:**

- ["Migrate existing custom modules to the Laminas framework" below](#)
- ["Influence activity events, emails, etc." on page 756](#)
- ["Templates" on page 758](#)
- ["Custom module file locations" on page 758](#)
- ["IndexControllers" on page 758](#)
- ["View helpers" on page 759](#)
- ["Event listeners" on page 759](#)
- ["Enabling your custom modules" on page 761](#)
- ["Further reading" on page 762](#) , includes links to example custom modules and the official Laminas documentation

## Migrate existing custom modules to the Laminas framework

P4 Code Review 2020.1 and later uses the Laminas framework, previous P4 Code Review versions used the Zend framework. The move to Laminas is part of our commitment to move away from using versions of platforms that have reached End-of-Life (EOL).

If you have any custom P4 Code Review modules that were created for P4 Code Review 2019.3 or earlier, you must modify them so that they will work in P4 Code Review 2020.1 and later. The modifications required depend on the version of P4 Code Review you were using the custom modules with before your upgraded P4 Code Review:

- ["Custom modules on P4 Code Review 2019.1, 2019.2, or 2019.3 " below](#)
- ["Custom modules on P4 Code Review 2018.3 and earlier" below](#)

### Custom modules on P4 Code Review 2019.1, 2019.2, or 2019.3

**Note:**

Migrate your custom modules on a test system before transferring them to your production system. This avoids any negative impact on the operation of your production system. If you have more than one custom module, the modules should all be tested at the same time on the same test system as this ensures that the modules operate correctly with each other and with P4 Code Review.

Custom modules used with P4 Code Review 2019.1, 2019.2, and 2019.3 use the Zend 3 framework, they must be migrated to the Laminas framework before they can be used with P4 Code Review 2020.1 and later:

1. Migrate your Zend 3 custom modules to the Laminas framework by pointing the Laminas migration tool at the folder you store your custom modules in. For the migration tool and full instructions on migrating your custom Zend 3 modules to Laminas, see [Migration to Laminas](#).
2. P4 Code Review will not use the custom modules until the P4 Code Review config cache has been deleted. For instructions on deleting the config cache, see ["P4 Code Review config cache file delete" on page 801](#).
3. Check that your custom modules work as expected before putting them on your production system.

### Custom modules on P4 Code Review 2018.3 and earlier

**Note:**

Update and migrate your custom modules on a test system before transferring them to your production system. This avoids any negative impact on the operation of your production system. If you have more than one custom module, the modules should all be tested at the same time on the same test system as this ensures that the modules operate correctly with each other and with P4 Code Review.

Custom modules used with P4 Code Review 2018.3 and earlier use the Zend 2 framework, they must be migrated to the Laminas framework before they can be used with P4 Code Review 2020.1 and later:

1. Convert your Zend 2 custom modules to the Zend 3 framework, see [Upgrade custom modules to work with Zend 3](#).

**Tip:**

Laminas is based on the Zend 3 framework, this makes it is relatively simple to convert your Zend 3 custom modules to the Laminas framework:

- If you only have one or two custom modules, you can manually change all of the Zend references in your modules to Laminas while you are converting them from Zend 2. This means you can skip the Laminas migration step.
- If you have a larger number of custom modules, it is probably quicker to automatically change all of the references in your custom modules by running the Laminas migration tool as described in the next step.

2. Migrate your Zend 3 custom modules to the Laminas framework by pointing the Laminas migration tool at the folder you store your custom modules in. For the migration tool and full instructions on migrating your custom Zend 3 modules to Laminas, see [Migration to Laminas](#).
3. P4 Code Review will not use the custom modules until the P4 Code Review config cache has been deleted. For instructions on deleting the config cache, see ["P4 Code Review config cache file delete" on page 801](#).
4. Check that your custom modules work as expected before putting them on your production system.

## Influence activity events, emails, etc.

When something happens in P4 Server (change submitted, files shelved, job added/edited), or state changes within P4 Code Review (comment added, review state changed, etc.), the event is pushed onto the P4 Code Review task queue. A background worker process pulls events off of the queue and publishes an event alerting modules about activity they may be interested in processing. This architecture allows the P4 Code Review user interface to be fairly quick while accommodating tasks that might require notable processing time, or need to wait for related information to become available.

Subscribers to the worker event flesh the item out (fetch the change/job details, for example) and indicate if it should result in an activity entry, email notification, etc. By subscribing to various event topics, your custom module can pay attention to specific types of events. While your custom module is processing an event, it can modify the text of activity events, change the contents of emails, drop things entirely from activity, etc.

When your custom module subscribes to an event, set the priority to influence how early or late in the process it runs. You will likely want your module to run after most other modules have done their work to flesh out the event, but before P4 Code Review's events module processes it. The events module is a good example of subscribing to these events:

```
swarm_install/module/Events/config/module.config.php
```

**Tip:**

Note that its priority is set to -100. Select a value before that for your own module (for example, 0 would be neutral and -90 would indicate that you are interested in being last).

Event priority is from the highest positive number to the lowest negative number with 0 in the middle. For a full list of tasks and their event priorities, see ["Task details" on page 762](#).



## Task types

For details of each of the following tasks, see ["Task details" on page 762](#).

- `task.commit`
- `task.shelve`
- `task.review`
- `task.change`
- `task.mail`
- `task.cache.integrity`
- `task.cleanup.attachment`
- `task.attachment.thumbnail`
- `task.cleanup.archive`
- `task.shelvedel`
- `task.changesave`
- `task.changesaved`
- `task.comment`
- `task.comment.batch`
- `task.commentSendDelay`
- `task.group`
- `task.groupdel`
- `task.job`
- `task.ping`
- `task.project.created`
- `task.project.updated`
- `task.project.updated`
- `task.testdefinition.migration`
- `task.testdefinition.updated`
- `task.testdefinition.created`
- `task.testdefinition.deleted`
- `task.testrun`
- `task.testrun.onDemand.started`
- `task.testrun.upgrade`
- `task.user`
- `task.userdel`
- `task.workflow.created`
- `task.workflow.updated`

- `task.workflow.deleted`
- `task.workflow.upgrade`

## Templates

Override existing view templates using your custom module. Have a look at this [example module](#) that demonstrates how to customize the email templates P4 Code Review uses for comment notifications.

For more information about views, see the [Laminas-View Quick Start](#).

### Note:

P4 Code Review supports the Laminas component versions in the LICENSE.txt file, features and functions in the Laminas documentation that were introduced in later versions of Laminas will not work with P4 Code Review. The LICENSE.txt file is in the readme folder of your P4 Code Review installation.

## Custom module file locations

The custom module files must be stored using the PSR-4 standard locations shown below:

```
config/
 custom.modules.config.php (required)
module/
 ModuleName/ (required)
 config/ (required)
 module.config.php (required)
 src/ (php files here, required)
 Controller/
 MyIndexController.php
 Listener/ (optional)
 MyListener.php (optional)
 ...
 other directories (non-php files here, optional)
 ...
 Module.php (required)
```

## IndexControllers

Create your IndexControllers using factories. The IndexControllers should extend `Application\Controller\AbstractIndexController` and use the P4 Code Review `IndexControllerFactory`, this means that the `IndexControllerFactory` can automatically inject the services .

### Tip:

We recommend you follow the `::class` description to avoid errors in string representation.

```
'controllers' => array(
 'factories' => [
 MyModuleName\Controller\IndexController::class =>
```

```
Application\Controller\IndexControllerFactory::class,
],
),
```

## View helpers

Create your view helpers using factories. The factories that create the view helpers must inject any services that are required.

### Tip:

We recommend using `::class` rather than arbitrary strings as this can limit the possibility of errors. The use of classes also makes your factory simpler because reflection can be used to construct the helpers if they all extend a common parent class.

```
'view_helpers' => array(
 'factories' => array_fill_keys(
 [
 MyModuleName\View\Helper\Helper1::class,
 MyModuleName\View\Helper\Helper2::class,
],
 <YourHelperFactory>::class
)
)
```

### Set options on existing helpers

It is possible to influence the behavior of existing view helpers by setting options on them; for an example see: [swarm\\_install/module/Application/Module.php](#).

### Register new helpers

It is also possible to register new view helpers by placing them within your module's hierarchy, for example, `MyModule/src/View/Helper/Foo.php`. Use the following P4 Code Review view helper for inspiration: [swarm\\_install/module/Activity/src/View/Helper/Activity.php](#).

Then register your view helper with the view manager via your `ModuleConfig`: [swarm\\_root/module/Activity/config/module.config.php](#).

## Event listeners

Your event listeners should extend `Events\Listener\AbstractEventListener`. This means that the `ListenerFactory` can create them without you having to write your own factory.

The `Listener.php` file contains the implementation of your event listener and the `module.config.php` file configures your event listeners.

**Tip:**

- Look at the code in the `AbstractEventListener.php` file in `module/events/src/Listener` to see the functions that are available to you.
- We strongly recommend that you create your event listeners to use the declarative model because it has a number of advantages over the non-declarative model:
  - `AbstractEventListener` provides common code for listeners to use
  - The declarative model offers better debug options (logging)
  - The declarative model is neater, easier to read, easier to support, and easier to maintain

**Example `module.config.php` file that uses the declarative model for event listeners:****Tip:**

`Listener::class` example details:

- `Events\Listener\ListenerFactory::ALL => [`  
 We are listening to ALL here for convenience because we are listening for mail events but this means it will trigger on every event. Usually is better to just listen for the events you are interested in. For example, if you are interested in commits and shelves you would listen on COMMIT and SHELVE events.  
`]`
- `Events\Listener\ListenerFactory::PRIORITY => -199,`  
 Declares an event priority of -199 for the event listener because email delivery events are processed with a priority of -200 and the example event needs to run just before the email delivery event.
- `Events\Listener\ListenerFactory::CALLBACK => 'handleEmail',`  
 Declares the function name within the listener class that is called.
- `Events\Listener\ListenerFactory::MANAGER_CONTEXT => 'queue'`  
 Triggers your custom listener when P4 Code Review processes the P4 Code Review event queue.

```
<?php
/**
 * Perforce Swarm
 *
 * @copyright 2019 Perforce Software. All rights reserved.
 * @license Please see LICENSE.txt in top-level folder of this distribution.
 * @version <release>/<patch>
 */

$listeners = [EmailExample\Listener\Listener::class];
return [
 'listeners' => $listeners,
 'service_manager' => [
 'factories' => array_fill_keys(
```

```

 $listeners,
 Events\Listener\ListenerFactory::class
)
],
Events\Listener\ListenerFactory::EVENT_LISTENER_CONFIG => [
 Events\Listener\ListenerFactory::ALL => [
 EmailExample\Listener\Listener::class => [
 [
 Events\Listener\ListenerFactory::PRIORITY => -199,
 Events\Listener\ListenerFactory::CALLBACK => 'handleEmail',
 Events\Listener\ListenerFactory::MANAGER_CONTEXT => 'queue'
]
]
]
];

```

#### Example of an email Listener.php file:

```

namespace MyModuleName\Listener;

use Events\Listener\AbstractEventListener;
use Laminas\EventManager\Event;

class Listener extends AbstractEventListener
{
 public function handleEmail(Event $event)
 {
 $mail = $event->getParam('mail');
 if (!$mail || !isset($mail['htmlTemplate'], $mail['textTemplate'])) {
 return;
 }
 }
}

```

## Enabling your custom modules

P4 Code Review uses the `custom.modules.config.php` file to check which custom modules it should load. This gives you control over which modules P4 Code Review loads and prevents modules from being loaded by mistake.

Create the `custom.modules.config.php` file in the `config` directory (a peer of the `modules` directory) if it does not already exist. Edit the file so that it contains the following details for each of your modules:

- `namespaces` array: enter your custom module names and paths
- `return` array : enter your custom module names

For example, if you have three modules called `MyModuleName`, `AnotherModuleName`, and `NewModuleName` the file content would look similar to:

```

<?php
\Laminas\Loader\AutoloaderFactory::factory(
 array(
 'Laminas\Loader\StandardAutoloader' => array(
 'namespaces' => array(
 'MyModuleName' => BASE_PATH . '/module/MyModuleName/src',
 'AnotherModuleName' => BASE_PATH . '/module/AnotherModuleName/src',
 'NewModuleName' => BASE_PATH . '/module/NewModuleName/src',
)
)
)
);
return [
 'MyModuleName',
 'AnotherModuleName',
 'NewModuleName',
];

```

## Further reading

For detailed information about the Laminas framework and examples see the following:

- The [Laminas Quick Start documentation](#) and [Laminas framework documentation portal](#), it is useful to have a basic knowledge of the Laminas framework before you create your own modules. .

### Note:

P4 Code Review supports the Laminas component versions in the LICENSE.txt file, features and functions in the Laminas documentation that were introduced in later versions of Laminas will not work with P4 Code Review. The LICENSE.txt file is in the readme folder of your P4 Code Review installation.

- ["Example email module" on page 767](#), a simple custom module that makes P4 Code Review use a custom email template when it sends out when a comment notification is sent out.
- P4 Code Review JIRA module, a simple module implementation within P4 Code Review: [swarm\\_install/module/Jira](#)

## Task details

P4 Code Review has lots of events that can be listened to and it uses these to process the tasks to generate data for P4 Code Review.

This section lists the P4 Code Review tasks available for use with your modules. They are listed along with the priorities of the task events. Events for each task are listed in highest to lowest priority order.

## task.commit

- **Priority 200:** the listener searches for all of the projects that are impacted by the change and checks for any user mentions that need to be linked. It sets up the email list here and queues the activity to be processed.
- **Priority 100:** the listener determines whether it should update or create any reviews for the changelist.
- **Priority -100:** the listener checks if any users have the Review daemon setup and attaches those users to the email to list.
- **Priority -200:** has two Listeners:
  - One listener checks the `disable_change_emails` configurable setting:
    - `true`, the email will not be sent.
    - `false`, P4 Code Review adds any reviews that might not be part of the *to list* because they are not a member of the project.
  - The other listener creates the activity for all users/projects that are linked to this event.
- **Priority -300:** the listener sends out the email that is related to the committed change.
- **Priority -400:** the listener links to JIRA for this change list.

## task.shelve

- **Priority 100:** the listener processes the shelve task to determine whether it should create or updates any reviews.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.
- **Priority -300:** the listener sends out the email that is related to the shelved change.

## task.review

- **Priority 100:** the listener processes the review and updates any records that are linked to the review. For example, @mention, new author, new participants, activity, and requirement level.
- **Priority -200:** the listener creates the activity for all users/project that are linked to this event.
- **Priority -300:** has two Listeners:
  - One listener sends out the email that is related to the review change.
  - The other listener fetches any associated changes and ensures they are linked to JIRA.

## task.change

**Priority 1:** the listener ensures the change and any mentions in the description are linked to JIRA issues.

## task.mail

**Priority 1:** this event is manually triggered from the `review` task. Other events do not manually trigger this task, they wait for it to run at the end. Once we are here P4 Code Review will validate all of the email settings and send the email out.

## task.cache.integrity

**Priority 1:** Listener to Build the cache integrity and update any records that are missing or incorrect. At present this is for users, groups, projects and workflow.

## task.cleanup.attachment

**Priority 1:** the listener will delete the attachment if a cleanup attachment task has been queued.

## task.attachment.thumbnail

**Priority 1:** the listener generates a thumbnail image from a file associated with an attachment. If the file is a simple image type the PHP gd library (if present) will be used for thumbnail generation, otherwise this is delegated to ImageMagick. Once generated this image will be set as an attribute on the depot file associated with the attachment.

## task.cleanup.archive

**Priority 1:** the listener will clean up any archive files if their life time has expired.

## task.shelvedel

- **Priority 1:** the listener processes any changelist that has had files deleted from it.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.

## task.changesave

**Priority 1:** the listener processes the change to ensure that it is valid and puts a future task in the queue to do the `task.changesaved` tasks.

## task.changesaved

**Priority 1:** the listener takes the changelist description and updates the review description if `sync description` is enabled.

## task.comment

- **Priority 1:** the listener processes the comment being created.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.
- **Priority -300:** the listener sends out an email related to the committed change.

## task.comment.batch

- **Priority 1:** If comments are being batched, the listener will process the comment(s) and put a task into the queue to be processed when the `notification_delay_time` has elapsed.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.
- **Priority -300:** the listener sends out an email related to the committed change.



## task.commentSendDelay

- **Priority 1:** if the user or system has forced a batched comment to be sent the listener will setup the email here.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.
- **Priority -300:** the listener sends out an email related to the committed change.

## task.group

**Priority 1:** the listener invalidates the group cache if the group is changed.

## task.groupdel

**Priority 1:** the listener invalidates the group cache if the group has been deleted.

## task.job

- **Priority 1:** the listener sets up the Activity for the job and attaches any users/projects to the activity. It also checks if any users want to be notified about Job changes.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.
- **Priority -300:** has two Listeners:
  - One listener sends out an email related to the committed change.
  - The other listener fetches associated changes and ensures they are linked to JIRA.

## task.ping

**Priority 1:** Receive Ping, saves the ping record

## task.project.created

- **Priority 1:** The listener sets up the Activity for the project created and attaches any users to the activity.
- **Priority -200:** The listener creates the activity for all users linked to this event.

## task.project.updated

- **Priority 1:** The listener sets up the Activity for the project created and attaches any users to the activity.
- **Priority -200:** The listener creates the activity for all users linked to this event.

## task.testdefinition.migration

- **Priority 1:** the listener Handle the test definitions migration event by creating activity to be processed
- **Priority -200:** the listener creates the activity when testdefinition is migrated

## **task.testdefinition.updated**

- **Priority 1:** the listener Handle the test definitions update event
- **Priority -200:** the listener creates the activity when testdefinition is updated

## **task.testdefinition.created**

- **Priority 1:** the listener Handle the test definitions creation event
- **Priority -200:** the listener creates the activity when testdefinition is created

## **task.testdefinition.deleted**

- **Priority 1:** the listener Handle the test definitions deletion event
- **Priority -200:** the listener creates the activity when testdefinition is deleted

## **task.testrun**

**Priority 1:** Listener for TestRun add/update. Updates the overall test status on a review if the TestRun changed is for the latest version.

## **task.testrun.onDemand.started**

- **Priority 1:** Listener for 'on demand' test requests and attaches an activity when one of these tests has been started.
- **Priority -200:** the listener creates the activity when one of these 'on demand' tests has been started.

## **task.testrun.upgrade**

- **Priority 1:** Listener to upgrade existing TestRun instances triggered by an upgrade task in the queue
- **Priority -200:** The listener creates the activity when the testrun is upgraded.

## **task.user**

**Priority 1:** the listener invalidates the user cache if the user is changed.

## **task.userdel**

**Priority 1:** the listener invalidates the user cache if the user has been deleted.

## **task.workflow.created**

- **Priority 1:** the listener sets up the Activity for the new workflow and attaches any users/projects to the activity.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.

## task.workflow.updated

- **Priority 1:** the listener sets up the Activity for the updated workflow and attaches any users/projects to the activity.
- **Priority -200:** the listener creates the activity for all users/projects that are linked to this event.

## task.workflow.deleted

- **Priority 1:** the listener sets up the Activity for the deleted workflow and attaches any users/projects to the activity.
- **Priority -200:** the listener creates the activity for all users/projects linked to this event.

## task.workflow.upgrade

- **Priority 1:** the listener sets up the Activity for the upgraded workflow and attaches any users/projects to the activity.
- **Priority -200:** the listener creates the activity for all users/projects linked to this event.

## Example email module

The following example module demonstrates how to customize the email template P4 Code Review uses when sending notifications for comments.

### Note:

P4 Code Review supports the Laminas component versions in the LICENSE.txt file, features and functions in the Laminas documentation that were introduced in later versions of Laminas will not work with P4 Code Review. The LICENSE.txt file is in the readme folder of your P4 Code Review installation.

### Tip:

You must test your custom modules on a test system before transferring them to your production system. This avoids any negative impact on the operation of your production system. If you have more than one custom module, the modules should all be tested at the same time on the same test system as this ensures that the modules operate correctly with each other and with P4 Code Review.

### Tip:

If you add or edit a module, P4 Code Review will not use that change until the config cache has been reloaded, this forces P4 Code Review to use the module change. You must be an *admin* or *super* user to reload the P4 Code Review config cache. Navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on page 720.

**Basic steps needed to create the Email Example module are:**

- "Create the Module.php file" below
- "Create the module.config.php file" on the facing page
- "Create the Listener.php file" on page 771
- "Create the commit-html.phtml file" on page 773
- "Create the commit-text.phtml file" on page 775
- "Enable the EmailExample module for P4 Code Review in custom.modules.config.php" on page 775

**Further reading:**

"Extending the EmailExample module to other email templates" on page 776

## File locations

For reference, the EmailExample module uses the following filenames and locations:

```
config/
 custom.modules.config.php
module/
 EmailExample/
 config/
 module.config.php
 src/
 Listener/
 Listener.php
 view/
 mail/
 commit-html.phtml
 commit-text.phtml
 comment-html.phtml
 comment-text.phtml
 review-html.phtml
 review-text.phtml
 Module.php
```

## Create the Module.php file

Module.php will:

- Declare the module namespace, this must match the directory name of the module
- Provide the getConfig() function

**Tip:**

**Optional:** You can also implement the onBootstrap (Event \$event) function if you want to do some early setup when the module is loaded:

```
public function onBootstrap(Event $event)
{
}
```

Event is a Laminas class and is added after namespace:

```
namespace EmailExample
use Laminas\EventManager\Event
```

### Create the Module.php file:

1. Create a directory called `EmailExample` in the module directory.
2. Create the file `Module.php` in `module/EmailExample`.
3. Edit `Module.php` to contain:

```
<?php
/**
 * Perforce Swarm
 *
 * @copyright 2019 Perforce Software. All rights reserved.
 * @license Please see LICENSE.txt in top-level folder of this distribution.
 * @version <release>/<patch>
 */

namespace EmailExample;

class Module
{
 public function getConfig()
 {
 return include __DIR__ . '/config/module.config.php';
 }
}
```

4. Now ["Create the module.config.php file" below](#).

### Create the module.config.php file

The `module.config.php` file will:

- Configure your module including event listeners
- Declare an event priority of `-199` for the event listener because email delivery events are processed with a priority of `-200` and the example event needs to run just before the email delivery event.

#### Tip:

Listener::class example details:

- `Events\Listener\ListenerFactory::ALL => [`  
 We are listening to ALL here for convenience because we are listening for mail events but this means it will trigger on every event. Usually is better to just listen for the events you are interested in. For example, if you are interested in commits and shelves you would listen on COMMIT and SHELVE events.
- `Events\Listener\ListenerFactory::PRIORITY => -199,`  
 Declares an event priority of -199 for the event listener because email delivery events are processed with a priority of -200 and the example event needs to run just before the email delivery event.
- `Events\Listener\ListenerFactory::CALLBACK => 'handleEmail',`  
 Declares the function name within the listener class that is called.
- `Events\Listener\ListenerFactory::MANAGER_CONTEXT => 'queue'`  
 Triggers your custom listener when P4 Code Review processes the P4 Code Review event queue.

#### Create the `module.config.php` file:

1. Create a directory called `config` in the `EmailExample` directory.
2. Create a file called `module.config.php` in `config`.
3. Edit `module.config.php` to contain:

```
<?php
/**
 * Perforce Swarm
 *
 * @copyright 2019 Perforce Software. All rights reserved.
 * @license Please see LICENSE.txt in top-level folder of this distribution.
 * @version <release>/<patch>
 */

$listeners = [EmailExample\Listener\Listener::class];
return [
 'listeners' => $listeners,
 'service_manager' => [
 'factories' => array_fill_keys(
 $listeners,
 Events\Listener\ListenerFactory::class
)
],
 Events\Listener\ListenerFactory::EVENT_LISTENER_CONFIG => [
 Events\Listener\ListenerFactory::ALL => [
 EmailExample\Listener\Listener::class => [
 [
 Events\Listener\ListenerFactory::PRIORITY => -199,
```

```
Events\Listener\ListenerFactory::CALLBACK => 'handleEmail',
Events\Listener\ListenerFactory::MANAGER_CONTEXT => 'queue'
]
]
]
];
```

4. Now ["Create the Listener.php file"](#) below.

## Create the Listener.php file

The Listener.php file will:

- Contains the implementation of your event listener
- Allow logging

### Create the Listener.php file:

1. Create a directory called src in the EmailExample directory.
2. Create a directory called Listener in the src directory.
3. Create a file called Listener.php in Listener.
4. Edit Listener.php to contain:

```
<?php
/**
 * Perforce Swarm
 *
 * @copyright 2019 Perforce Software. All rights reserved.
 * @license Please see LICENSE.txt in top-level folder of this distribution.
 * @version <release>/<patch>
 */

namespace EmailExample\Listener;

use Events\Listener\AbstractEventListener;
use Laminas\EventManager\Event;

class Listener extends AbstractEventListener
{
 /**
 * Automatically uses any custom email templates found under this
 * module's view/mail folder (e.g. Example/view/mail/commit-html.phtml).
 *
 * Valid templates include:
 *
 * commit-html.phtml (HTML version of commit notification)
 * commit-text.phtml (text version of commit notification)
 * comment-html.phtml (HTML version of comment notification)
 * comment-text.phtml (text version of comment notification)
 * review-html.phtml (HTML version of review notification)
 * review-text.phtml (text version of review notification)
 *
 * Note: you need to provide custom templates for both HTML and text;
 * if you do not provide both, it is possible that the search for
 * customized templates only finds the non-customized versions, making
 * it appear that this module is not working.
 */

 /**
 * Handle the Email and set the new templates.
 *
 * @param Event $event
 */
 public function handleEmail(Event $event)
 {
 $logger = $this->services->get('logger');
 $logger->info("EmailExample: handleEmail");
 $mail = $event->getParam('mail');
 if (!$mail || !isset($mail['htmlTemplate'], $mail['textTemplate'])) {
 return;
 }
 }
}
```



```

 }

 $html = __DIR__ . '/view/mail/' . basename($mail['htmlTemplate']);
 $text = __DIR__ . '/view/mail/' . basename($mail['textTemplate']);

 if (file_exists($html)) {
 $mail['htmlTemplate'] = $html;
 }
 if (file_exists($text)) {
 $mail['textTemplate'] = $text;
 }

 $event->setParam('mail', $mail);
 $logger->info("EmailExample: handleEmail end.");
}
}

```

- Now "Create the comment-html.phtml file" below.

## Create the comment-html.phtml file

The comment-html.phtml file is a view script that provides the content for the HTML portion of the comment notification email.

### Tip:

It is considered best practice to use inline CSS for styling emails.

### Create the comment-html.phtml file:

- Create a directory called `view` in the `module/Example` directory.
- Create a directory called `mail` in the `module/Example/view` directory.
- Create the `comment-html.phtml` file in `module/Example/view/mail`.
- Edit `comment-html.phtml` to contain:

```

<?php
 $user = $activity->get('user');
 $userLink = $user
 ? $this->qualifiedUrl('user', array('user' => $user))
 : null;
 $targetLink = $activity->getUrl($this->plugin('qualifiedUrl'));
?>
<html>
<body style="font-family: sans-serif; background-color: #eee; padding: 1em;">
<div style="background-color: #fff; border: 1px solid #ccc; padding: 1em;">
<div style="font-size: 115%;">
 <?php if ($user): ?>
 <a style="text-decoration: none;" href="<?php echo $userLink ?>">
 <?php echo $this->escapeHtml($user) ?>

```

```

<?php endif; ?>
<?php echo $this->escapeHtml($activity->get('action')) ?>
<a style="text-decoration: none;" href="<?php echo $targetLink ?>">
 <?php echo $this->escapeHtml($activity->get('target'))?>

</div>

<?php
 // if the comment has file context, show it.
 $comment = $event->getParam('comment');
 $context = $comment
 ? $comment->getFileContext()
 : array('content' => null, 'line' => null);
 if (is_array($context['content']) && $context['line']) {
 $line = $context['line'] - count($context['content']) + 1;
 echo '<div style="font-family: monospace; white-space: nowrap;'
 . ' padding: .5em 1em; overflow-x: auto; color: #444;'
 . ' border: 1px solid #ddd; background-color: #f7f7f7;">';
 foreach ((array) $context['content'] as $i => $content) {
 echo '<div>'
 . str_pad($line + $i,
 strlen($context['line']),
 "0",
 STR_PAD_LEFT
)
 . ' ';
 $this->preformat($content)
 ->setLinkify(false)
 ->setEmojify(false)
 ->setWordWrap(900)
 . "</div>\n";
 }
 echo '</div>
';
 }
?>
<div style="padding-bottom: .5em;">
<?php
 echo $this->preformat($activity->get('description'))
 ->setBaseUrl($this->qualifiedUrl())
 ->setEmojify(false)
 ->setWordWrap(900)
?>
</div>
</div>
</body>
</html>
```

- Now ["Create the comment-text.phtml file" below.](#)

## Create the comment-text.phtml file

The `comment-text.phtml` is a view script that provides the content for the text-only portion of the comment notification email.

### Create the comment-text.phtml file:

- Create the file `comment-text.phtml` in the `module/Example/view/mail` directory.
- Edit `comment-text.phtml` to contain:

```
<?php
 echo trim($activity->get('user')
 . ' commented on '
 . $activity->get('target'));

 // if the comment has file context, show it.
 $comment = $event->getParam('comment');
 $context = $comment
 ? $comment->getFileContext()
 : array('content' => null);
 if (is_array($context['content'])) {
 echo "\n\n> " . $this->wordWrap(
 implode("\n> ", $context['content']), 900
);
 }

 echo "\n\n" . trim($this->wordWrap($activity->get('description'), 900));
 echo "\n\n" . $activity->getUrl($this->plugin('qualifiedUrl'));
?>
```

- Now ["Enable the EmailExample module for P4 Code Review in custom.modules.config.php" below.](#)

## Enable the EmailExample module for P4 Code Review in custom.modules.config.php

P4 Code Review uses the `custom.modules.config.php` file to auto-load classes and to check which custom modules it should run. This gives you control over which modules P4 Code Review loads and prevents modules from being loaded by mistake.

### Create the custom.modules.config.php file:

- Create the `config` directory at the same level as the `module` directory if it does not exist.
- Create the `custom.modules.config.php` file in the `config` directory if it does not exist.
- Edit the `custom.modules.config.php` file so that it contains the auto-loader and the EmailExample module details:

**Tip:**

If you already have one or more custom modules enabled for P4 Code Review, the auto-loader and the existing module details will already be in the file.

Just add EmailExample to the namespaces and return arrays of the custom.modules.config.php file.

```
<?php
\Laminas\Loader\AutoloaderFactory::factory(
 array(
 'Laminas\Loader\StandardAutoloader' => array(
 'namespaces' => array(
 'EmailExample' => BASE_PATH . '/module/EmailExample/src',
)
)
)
);
return [
 'EmailExample'
];
```

4. The P4 Code Review config cache must be reloaded so that P4 Code Review can see your new module. As an *admin* or *super* user, navigate to the **User id** dropdown menu, select **System Information**, click the **Cache Info** tab, and click the ["Reload Configuration button"](#) on [page 720](#).

The Email Example module is now enabled for P4 Code Review. P4 Code Review will use the new custom email template whenever a comment notification is sent out.

5. Check that the module works correctly before moving it to your production server.

## Extending the EmailExample module to other email templates

If you need to customize any other types of P4 Code Review notification email messages, locate the view scripts (both HTML and text) and copy them into `module/Example/view/mail`, maintaining the existing filenames, then modify the new files as desired.

**Note:**

If you do not copy both the HTML and text templates, it is possible for the search for customized templates to only find non-customized versions, making it appear that your module is not working.

## CSS & JavaScript

Custom CSS and JavaScript files will be loaded automatically, following a browser refresh, if you place them under either:

- `SWARM_ROOT/public/custom/*.css|js`
- or
- `SWARM_ROOT/public/custom/sub-folder/*.css|js`

**Note:**

- P4 Code Review only supports customizations placed directly within the `SWARM_ROOT/public/custom` folder or one sub-folder down. Customizations are added after all standard CSS and JavaScript. If more than one custom file is present, they are added in alphabetical order.
- Before creating any customizations, make sure that the `SWARM_ROOT/public/custom` folder exists. P4 Code Review does not ship with or create this folder.
- When you add custom CSS or JavaScript to P4 Code Review, only a browser refresh is required. You do not need to restart `apachectl`.
- If you have checked the custom CSS or JavaScript code for accuracy and it still doesn't work after a browser refresh, check the folder permissions for the folder the customization is in.

## Sample JavaScript extensions

The following is an example JavaScript customization that you might wish to apply to your P4 Code Review installation. JavaScript customizations can be implemented separately, but ideally, you would apply the JavaScript customizations in a single file to reduce the number of web requests required.

**Warning:**

Coding errors in your custom JavaScript files could cause the P4 Code Review UI to stop working. If this occurs, use your browser's development tools to identify which file contains the problem, and move that file out of the `SWARM_ROOT/public/custom` folder. When the problem has been resolved, the file can be returned.

## Add text to the Log in dialog

This customization can only be used if `require_login` is set to true, see ["Require login" on page 686](#).

```
// Custom js to put text into the login box
$(function() {
 setTimeout(function () {
 if ($('.login-form')) {
 $('.form-body')
 .prepend("<b style='padding-bottom: 20px; display: inline-block'>Please use your
Helix Core Server login credentials.");
 $('.form-body button').html('Login with Helix Core Server credentials');
 }
 },
```

```
800);
});
```

Replace the *Please use your P4 Server login credentials* with the text you want to add to the top of the P4 Code Review log in dialog.

Save this line in a file, perhaps `customize-login-text.js`, within the `SWARM_ROOT/public/custom`. P4 Code Review automatically includes the JavaScript file, adding the text to the top of the log in dialog immediately for subsequent log ins.

## Sample CSS customizations

The following are example CSS customizations that you might wish to apply to your P4 Code Review installation. Each example can be implemented separately. Ideally, you would apply the CSS customizations in a single file to reduce the number of web requests required.

### Tip:

If your P4 Code Review administrator has configured P4 Code Review to connect to more than one P4 Server instance you can customize P4 Code Review for each server instance it is connected to. For instructions on customizing P4 Code Review for individual P4 Server instances, see ["CSS customization when P4 Code Review is connected to multiple P4 Server instances" on page 781](#).

## Adjust the default tab size

```
body {
 tab-size: 4;
}
```

Replace the `4` with the tab size you prefer.

Save these lines in a file, perhaps `tab-size.css`, within the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, adjusting the tab size immediately for subsequent page views.

## Apply a custom background to the login screen

```
.session-container .modal-overlay.login::after {
 background-image: url('/custom/login_background.jpg');
 background-position: top center;
 background-size: 100%;
}
```

Replace the `/custom/login_background.jpg` URL fragment with the image you want to use. If you do not specify a full URL, the image you specify must exist within the `SWARM_ROOT/public` folder, preferably within the `SWARM_ROOT/public/custom` folder.

Save these lines in a file, perhaps `login-background.css`, within the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, immediately replacing the login screen's background for subsequent page views.

## Replace P4 Code Review's logo in the main navigation bar

```
#react-swarm-app-container .primary.navigation .swarmLogo a svg{
 display:none;
}
#react-swarm-app-container .primary.navigation .swarmLogo a {
 background-repeat: no-repeat;
 background-image: url('/custom/navbar_logo.jpg');
 background-size: 116px 25px;
 background-position: 15px center;
 background-clip: content-box;
}
```

Replace the `/custom/navbar_logo.jpg` URL fragment with the image you want to use. If you do not specify a full URL, the image you specify must exist within the `SWARM_ROOT/public` folder, preferably within the `SWARM_ROOT/public/custom` folder.

Save these lines in a file, perhaps `navbar-logo.css`, within the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, immediately replacing the P4 Code Review logo for subsequent page views.

### Note:

P4 Code Review's navbar design supports logos up to 24 pixels tall. Even if your logo fits within that height, you may need to also adjust the width, height, margins, or padding to suit your logo.

## Adjust the appearance of avatars

The P4 Code Review UI is changing and moving toward using React, currently there is a mix of React and legacy PHP controlling the UI. This means that there are currently two methods of rendering avatars and you need to customize both.

- For avatars rendered by P4 Code Review using PHP, see ["Adjust avatar appearance when rendered by PHP" below](#)
- For avatars rendered by P4 Code Review using React, see ["Adjust avatar appearance when rendered by React" on the next page](#)

For instructions on adding custom avatars, see ["Add custom avatars" on the next page](#)

### Adjust avatar appearance when rendered by PHP

```
img.avatar {
 border: 2px dashed red;
 border-radius: 10%;
}
```

You can make a number of adjustments to the way P4 Code Review presents avatars rendered by PHP, such as adding a border and adjusting the border radius, as the example above demonstrates. You should avoid attempting to set specific sizes because P4 Code Review uses different sizes depending on where the avatar is displayed.

Save these lines in a file, perhaps `avatars.css`, within the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, immediately changing the appearance of avatars for subsequent page views.

#### Adjust avatar appearance when rendered by React

```
#react-swarm-app-container .userAvatar img{
 border: 2px dashed red;
 border-radius: 10%;
}
```

You can make a number of adjustments to the way P4 Code Review presents avatars rendered by React, such as adding a border and adjusting the border radius, as the example above demonstrates. You should avoid attempting to set specific sizes because P4 Code Review uses different sizes depending on where the avatar is displayed.

Save these lines in a file, perhaps `avatars-react.css`, within the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, immediately changing the appearance of avatars for subsequent page views.

#### Add custom avatars

Custom avatars for individual users or groups are added to P4 Code Review using custom CSS. Custom avatars are intended for use with system users and groups.

##### To add a custom avatar for a user:

```
img.avatar[data-user="<userid>"], .userAvatar img[alt="<user name>"]{
 content: url(<path to avatar>);
}
```

##### To add a custom avatar for a group:

```
img.avatar[data-user="<groupid>"], .userAvatar img[alt="<group name>"]{
 content: url(<path to avatar>);
}
```

Save these lines in a file, perhaps `avatars-custom.css`, within the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, immediately using your custom avatars for subsequent page views.

## Add a custom icon to a menu item

When you create a custom menu item in the `menu_helpers` configuration block, you can specify the menu icon to use in the P4 Code Review CSS.



Add some custom CSS to P4 Code Review to replace the default icon with a custom icon for the menu item. For example, the CSS below replaces the default icon with the `double-speech-bubbles.svg` icon for the `custom01` menu item by modifying the `svg.svgIcon.custom01MenuIcon` and `.menuitem-custom01` classes:

```
.swarmMenu .menuitem.menuitem-tests .svgIcon{
 display:none;
}.swarmMenu .menuitem.menuitem-tests a::before{
 content: "";
 background-image: url('/swarm/img/icons/line/double-speech-bubbles.svg');
 background-size: 16px;
 background-repeat: no-repeat;
 padding-left: 20px;
 padding-right: 4px;
 margin-left: 20px;
}
```

Save these lines in a file, perhaps `menu_custom01.css` in a folder called `custom_menus`, in the `SWARM_ROOT/public/custom` folder. P4 Code Review automatically includes the CSS file, immediately using your custom menus for subsequent page views. If you are using your own custom images, we recommend you store them in the same folder as your custom css.

## CSS customization when P4 Code Review is connected to multiple P4 Server instances

If P4 Code Review is configured to connect to more than one P4 Server instance, P4 Code Review can be customized for each of the P4 Server instances it is connected to. This gives users an simple visual prompt as to which P4 Server instance they are currently viewing in P4 Code Review. All of the customizations described in the section above are available for multiple P4 Server instances.

### Note:

- The P4 Code Review Global Dashboard cannot be customized.
- For instructions on how to configure P4 Code Review to connect to multiple P4 Server instances, see ["Multiple P4 Server instances" on page 638](#).

To customize P4 Code Review for a P4 Server instance, include the `[data-server-id="<server-name>"]` data attribute in the CSS selector you are customizing. Replace `<server-name>` with the P4 Server name the customization is for.

### Tip:

To apply a customization to all P4 Server instances, don't include the data attribute in the CSS selector you are customizing.

See the examples below to see how the data attribute is used to customize P4 Code Review for the individual P4 Server instances. As with the customization of P4 Code Review when it is connected to a single P4 Server, you should ideally apply the CSS customizations in a single file to reduce the number of web requests required.

## Apply a custom background color to the navigation bar of each of the P4 Server instances

```
body[data-server-id="serverA"] #react-swarm-app-container .swarm-navigation {
 background-color: #d21544;
}
body[data-server-id="serverB"] #react-swarm-app-container .swarm-navigation {
 background-color: pink;
}
```

## Apply custom background images to the login screen of each of the P4 Server instances

```
body.route-login[data-server-id="serverA"] .session-container .modal-overlay.login::after {
 background-position: top center;
 background-size: 100%;
 background-image: url(/custom/login_background_A.jpg);
}
body.route-login[data-server-id="serverB"] .session-container .modal-overlay.login::after {
 background-position: top center;
 background-size: 100%;
 background-image: url(/custom/login_background_B.jpg);
}
```

# P4 Code Review API

**Important:**

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

This chapter describes the REST-like API provided by P4 Code Review, which can be used to automate common P4 Code Review interactions or integrate with external systems.

**In this section:**

## API versions

From P4 Code Review 2022.1, we added a new set of v10 APIs to P4 Code Review that replace the v9 APIs. The v10 APIs refine and extend the v9 APIs, and further standardize the endpoint and response pattern.

The v9 and v10 APIs will continue to be available for some time to come, but their functionality will be duplicated and expanded in the v11 APIs over the next few releases. v10 does not currently provide a full set of features, so it is recommended that you use v9 for now if the features that you need are not present in v10.

**Important:**

- From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9. P4 Code Review will continue to support the v9 APIs and you can continue to use them if you prefer.
- Any improvements made in the migration to v11 will not be backported to v9 or v10.
- New API endpoints will be created as v11 and will not be backported to v9 or v10.

## Current API versions

API version	P4 Code Review Release	Date	Description
v11	2022.1	March 2022	Extended the v10 APIs and further standardized the endpoint and response pattern for the new rich User Interface.

API version	P4 Code Review Release	Date	Description
v10	2019.3	October 2019	Include support for integration with CI tools.
v9	2018.1	April 2018	Include support for append and replace changelist to a review, 2 Factor Authentication and mark a comment as read.

## Unsupported API versions

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

API version	P4 Code Review Release	Date	Description
v8	2017.4	December 2017	Include support for default reviewers on a project or project branch.
v7	2017.3	October 2017	Include support for groups as participants of a review and groups as moderators of a project branch.
v6	2017.1	May 2017	Include support for activity dashboard, archiving of inactive reviews, cleaning completed reviews and for voting reviews up and down.
v5	2016.3	December 2016	Include support for limiting comments to a specific review version.

API version	P4 Code Review Release	Date	Description
v4	2016.2	October 2016	Include support for private projects, as well as file-level and line-level inline comments.
v3	2016.1 SP1	September 2016	Include new endpoint for comments.
v2	2016.1	May 2016	Include new endpoints for projects, groups, etc.
v1.2	2015.3	October 2015	Add author filter to the list reviews endpoint.
v1.1	2014.4	January 2015	Addition of required reviewers, and apiVersions.
v1	2014.3	July 2014	Initial release.

## Get P4 Code Review version information

The version endpoint lets you see the currently-installed P4 Code Review version, the year of release, and the included API versions.

GET /api/version

**Get P4 Code Review version and API versions:**

<https://myswarm.url/api/version>

**Example response:**

```
{"apiVersions":[1,1.1,1.2,2,3,4,5,6,7,8,9,10,11],"version":"SWARMV2022.1V2139325(2022V01V28)","year":"2022"}
```

**Note:**

year refers to the year of the P4 Code Review release, not necessarily the current year.

## P4 Code Review API basics

**Important:**

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

This section describes the API endpoints that are commonly used when constructing an API call to P4 Code Review.

## Authentication

P4 Code Review's API requires an authenticated connection for all data-modifying endpoints. Authenticated connections are achieved using HTTP Basic Access Authentication.

**Note:**

API endpoints require authentication unless `require_login` is set to false.

For example:

```
curl -u "apiuser:ticket" https://myswarm.url/api/v9/projects
```

P4 Code Review accepts a ticket from the P4 Server (previous versions of P4 Code Review required this ticket to be host-unlocked, this is no longer true since P4 Code Review 2017.1). It might also be possible to use a password in place of the ticket if your P4 Server allows it.

To acquire a ticket, run the following command:

```
p4 -p myp4host:1666 -u apiuser login -p
```

**Important:**

For a P4 Server that has been configured for security level 3, passwords are not accepted. For more information on security levels, see on [Server security levels](#) in the [P4 Server Administration Documentation](#).

**Note:**

If you use a ticket to authenticate against the P4 Code Review API and the ticket expires, you need to acquire a new ticket to continue using the API.

If you make a request that requires authentication and you have not authenticated, the response is:

```
HTTP/1.1 200 OK
```

```
{
 "error": "Unauthorized"
}
```

## Requests

**Important:**

Some API endpoints have undocumented metadata parameters, these are for internal P4 Code Review use only. The metadata returned is subject to change without notice and is not suitable for use in customer API calls.

P4 Code Review's API includes endpoints that provide, create, and update information within P4 Code Review.

If you make a request against an endpoint with a method that is not supported, the response is:

```
HTTP/1.1 200 OK
```

```
{
 "error": "Method Not Allowed"
}
```

## GET information

**Tip:**

GET requests require authentication unless [require\\_login](#) is set to false.

Use HTTP GET requests to ask for information from the API.

For example, to get the list of reviews using the v10 API:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/reviews"
```

Certain API calls support a `fields` parameter that allows you to specify which fields to include in the response, enabling more compact data sets.

Fields can be expressed as a comma-separated list, or using array-notation. For example:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/projects/jam?fields=id,description,members"
```

The following endpoints support fields:

### API (v9)

- `/api/v9/activity`
- `/api/v9/comments`
- `/api/v9/groups`
- `/api/v9/groups/{id}`
- `/api/v9/projects`
- `/api/v9/reviews`
- `/api/v9/reviews/{id}`
- `/api/v9/users`

- `/api/v9/workflows`
- `/api/v9/workflows/{id}`

### API (v10)

- `/api/v10/files/{id}`
- `/api/v10/projects`
- `/api/v10/projects/{id}`
- `/api/v10/reviews`
- `/api/v10/reviews/{id}`
- `/api/v10/reviews/{id}/files`
- `/api/v10/reviews/{id}/testruns`
- `/api/v10/search`
- `/api/v10/specs/{spec}/fields`
- `/api/v10/testdefinitions`
- `/api/v10/workflows`
- `/api/v10/workflows/{id}`

## POST new information

Use HTTP `POST` requests to create information via the API.

For example, to create a review using form-encoded values:

```
curl -u "apiuser:ticket" -X POST -d"change=12345" https://myswarm.url/api/v9/reviews
```

The response should be similar to:

```
HTTP/1.1 200 OK
```

```
{
 "isValid": true,
 "id": 12206
}
```

To create a review using JSON:

```
curl -u "apiuser:ticket" -H "Content-type: application/json" -X POST -d '{"change": 12345}'
https://myswarm.url/api/v9/reviews
```

## Update

Use HTTP `PATCH` requests to update information via the API.

If your HTTP client does not support `PATCH` requests, you can emulate this behavior by submitting an HTTP `POST` with a `"?_method=PATCH"` parameter.



## Pagination

Most P4 Code Review endpoints that provide data include the ability to paginate their results.

Each time data is requested, up to `max` results are included in the response, as is a value called `lastSeen`. `lastSeen` identifies the `id` of the last entry included in the results. If there are no further results, `lastSeen` is `null`.

To get the next set of results, include `after` set to the value of `lastSeen` in the API request. Entries up to and including the `id` specified by `after` are excluded from the response, and the next `max` entries are included.

See the [Activity endpoint](#) for example usage that demonstrates pagination.

## Responses

P4 Code Review's API responses are JSON formatted.

### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

## API endpoints (v9)

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

This section includes coverage for the endpoints provided by the API (v9).

**In this section:**

## Activity : P4 Code Review Activity List

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## List Activity Entries

### Summary

List Activity Entries

GET /api/v9/activity

### Description

Retrieve the Activity List.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
change	Optionally filter activity entries by associated Changelist ID. This only includes records for which there is an activity entry in P4 Code Review.	integer	form	No	

---

Parameter	Description	Type	Parameter Type	Required	Default Value
stream	Optional activity stream to query for entries. This can include user-initiated actions (user-alice), activity relating to a user's followed projects/users (personal-alice), review streams (review-1234), and project streams (project-exampleproject).	string	form	No	
type	Type of activity, for example, change, comment, job, or review.	string	form	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
after	An activity ID to seek to. Activity entries up to and including the specified ID are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.	integer	query	No	
max	Maximum number of activity entries to return. This does not guarantee that max entries are returned. It does guarantee that the number of entries returned won't exceed max. Server-side filtering may exclude some activity entries for permissions reasons.	integer	query	No	100

Parameter	Description	Type	Parameter Type	Required	Default Value
fields	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No	

### Example response

#### Successful Response:

To get the latest activity entry on a review:

```
curl -u "username:password" http://myswarm.url/api/v9/activity?stream=review-290&max=1
```

P4 Code Review responds with an array of activity entities, and a `lastSeen` value that can be used for pagination:

HTTP/1.1 200 OK

```
{
 "activity": [
 {
 "id": 619,
 "action": "updated files in",
 "behalfOf": null,
 "behalfOfExists": false,
 "behalfOfFullName": "",
 "change": 290,
 "comments": [
 0,
 0
],
 "date": "2019-01-23T02:46:59-08:00",
 "depotFile": null,
 "description": "Start of Project Blue Book.",
 "details": [
```

```
,
"followers": [

],
"link": [
 "review",
 {
 "review": "290",
 "version": 2
 }
],
"preposition": "for",
"projectList": {
 "blue-book": [
 "main"
]
},
"projects": {
 "blue-book": [
 "main"
]
},
"streams": {
 "0": "review-290",
 "1": "user-allison.clayborne",
 "2": "personal-allison.clayborne",
 "3": "project-blue-book",
 "7": "group-triage",
 "8": "personal-alex.randolph",
 "9": "personal-jack.boone"
},
"target": "review 290 (revision 2)",
"time": 1548240419,
"topic": "reviews/290",
"type": "review",
"url": "/main/reviews/290/v2/",
"user": "allison.clayborne",
"userExists": true,
"userFullName": "Allison Clayborne"
}
],
"lastSeen": 618
}
```

## Examples of usage

### Fetching review activity

To get the latest activity entries on a review:

**Tip:**

In this example, the review has more than 2 activities but the request has been made to limit the response to a max of 2 activities. This will return the most recent activities unless `lastseen` is included in the request. This allows for pagination of your results, see ["Activity pagination" below](#).

```
curl -u "username:password" http://myswarm.url/api/v9/activity?stream=review-1234&fields=id,action,date,description,type&max=2
```

You can tweak `max` and `fields` to fetch the data that works best for you.

P4 Code Review responds with an array of activity entities, and a `lastSeen` value that can be used for pagination:

HTTP/1.1 200 OK

```
{
 "activity": [
 {
 "id": 619,
 "action": "updated files in",
 "date": "2019-01-23T02:46:59-08:00",
 "description": "Start of Project Blue Book.",
 "type": "review"
 },
 {
 "id": 618,
 "action": "commented on",
 "date": "2019-01-23T02:46:19-08:00",
 "description": "This is a short comment.",
 "type": "comment"
 }
],
 "lastSeen": 618
}
```

**Activity pagination**

To get the second page of activity entries for a review (based on the previous example):

```
curl -u "username:password" "https://myswarm.url/api/v9/activity?stream=review-1234&fields=id,date,description,type&max=2&after=618"
```

P4 Code Review again responds with a list of activity entities and an `lastSeen` value:

HTTP/1.1 200 OK

```
{
```

```
"activity": [
 {
 "id": 617,
 "action": "commented on",
 "date": "2018-12-30T12:12:12-07:00",
 "description": "This is the first test comment.",
 "type": "comment"
 },
 {
 "id": 616,
 "action": "commented on",
 "date": "2018-12-27T12:13:14-07:00",
 "description": "Start of Project Blue Book.",
 "type": "review"
 }
],
"lastSeen": 616
}
```

## Create Activity Entry

### Summary

Create Activity Entry

POST /api/v9/activity

### Description

Creates an entry in the Activity List.

#### Important:

The authenticated user must match the user provided unless the authenticated user has minimum admin-level privileges in which case any user may be specified.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.



## Parameters

Parameter	Description	Type	Parameter Type	Required
type	Type of activity, used for filtering activity streams (values can include change, comment, job, review).	string	form	Yes
user	User who performed the action.	string	form	Yes
action	Action that was performed - past-tense, for example, created or commented on.	string	form	Yes
target	Target that the action was performed on, for example, issue 1234.	string	form	Yes
topic	Optional topic for the activity entry. Topics are essentially comment thread IDs. Examples: reviews/1234 or jobs/job001234.	string	form	No

Parameter	Description	Type	Parameter Type	Required
description	Optional description of object or activity to provide context.	string	form	No
change	Optional changelist ID this activity is related to. Used to filter activity related to restricted changes.	integer	form	No
streams[]	Optional array of streams to display on. This can include user-initiated actions (user-alice), activity relating to a user's followed projects/users (personal-alice), review streams (review-1234), and project streams (project-exampleproject).	array (of strings)	form	No
link	Optional URL for target.	string	form	No

### Examples of usage

#### Creating an activity entry

To create a plain activity entry:

```
curl -u "username:password"
-X POST
```

```
-d "type=job"
-d "user=jira"
-d "action=punted"
-d "target=review 123" \
 "https://myswarm.url/api/v9/activity"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "activity":{
 "id":620,
 "action":"punted",
 "behalfOf":null,
 "change":null,
 "depotFile":null,
 "description":null,
 "details":[

],
 "followers":[

],
 "link":null,
 "preposition":"for",
 "projects":[

],
 "streams":[

],
 "target":"review 123",
 "time":1560783425,
 "topic":null,
 "type":"job",
 "user":"jira"
 }
}
```

#### Linking an activity entry to a review

Linking activity entries to reviews is useful. This involves providing `link`, `streams`, and `topic` fields in the activity data. The `link` field is used to make the `review 123` string in the activity entry clickable. The `stream` field is needed so that the activity entry can be attached to the review in the P4 Code Review interface. The `topic` field is used to link the activity entry to the comment thread for that topic, in the event that a user wants to comment on the activity.

To create a fully linked activity entry:

```
curl -u "username:password"
-X POST
-d "type=job"
-d "user=jira"
-d "action=punted"
-d "target=review 123" \
-d "streams[]=review-123" \
-d "link=reviews/123" \
-d "topic=reviews/123" \
"https://myswarm.url/api/v9/activity"
```

P4 Code Review responds with an activity entity:

HTTP/1.1 200 OK

```
{
 "activity": {
 "id": 1375,
 "action": "punted",
 "behalfOf": null,
 "change": null,
 "depotFile": null,
 "description": null,
 "details": [

],
 "followers": [

],
 "link": "reviews/123",
 "preposition": "for",
 "projects": [

],
 "streams": [
 "review-123"
],
 "target": "review 123",
 "time": 1461607739,
 "topic": "reviews/123",
 "type": "job",
 "user": "jira"
 }
}
```

## Cache: P4 Code Review Cache

**Important:**

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

### P4 Code Review config cache file delete

**Tip:**

Alternatively, use the **Cache Info** tab of the **System Information** page. Navigate to the **User id** dropdown menu, select **System Information**, and click the **Cache Info** tab. For more information, see ["Cache Info" on page 719](#).

**Summary**

Introduced for P4 Code Review 2019.2 and later, deletes the P4 Code Review configuration cache files and restarts the workers.

DELETE /api/v9/cache/config/

**Description**

Used to delete the P4 Code Review configuration cache files after the P4 Code Review configuration or custom modules have changed. When workers complete their current tasks they are automatically restarted so that they use the new P4 Code Review configuration and custom modules for their next tasks.

**Important:**

Only *admin* users can delete the P4 Code Review configuration cache.

**Example response**

**When cache files are not deleted and workers are not shutdown:**

For example, this can happen if you make the call and then make the same call again before the cache files have rebuilt.

HTTP/1.1 200 OK

```
{
 "isValid":true,
 "messages":[
 "File <SWARM_ROOT>/data/cache/module-config-cache.php was not deleted",
 "File <SWARM_ROOT>/data/cache/module-classmap-cache.php was not deleted"
]
}
```

### Example of usage

**Tip:**

It might also be possible to use a password in place of the ticket if your P4 Server allows it.

```
curl -u "username:ticket" -X DELETE "https://myswarm.url/api/v9/cache/config/"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the cache deleted and workers shutdown messages:

HTTP/1.1 200 OK

```
{
 "isValid":true,
 "messages":[
 "File <SWARM_ROOT>/data/cache/module-config-cache.php deleted",
 "File <SWARM_ROOT>/data/cache/module-classmap-cache.php deleted",
 "Workers are shutdown. They will start automatically by recurring task"
]
}
```

### Verify Redis cache

**Tip:**

Alternatively, use the **Cache Info** tab of the **System Information** page. Navigate to the **User id** dropdown menu, select **System Information**, and click the **Cache Info** tab. For more information, see ["Cache Info" on page 719](#).

### Summary

Introduced for P4 Code Review 2019.2 and later, verifies the P4 Code Review Redis cache.

POST /serverid/api/v9/cache/redis/verify?context=user,group,project,workflow

### Description

The data in the Redis cache for groups, users, projects, and workflows should be the same as the data held by the P4 Server. However, in some circumstances, such as when changes are made in the P4 Server when P4 Code Review is down or if errors occur during updates, the Redis cache can get out of sync with the P4 Server.

Used to verify the content of the Redis cache. Use the context parameter to verify specific areas of the Redis cache, if no context is specified the entire Redis cache is verified. If a verification finds an out of sync cache file, P4 Code Review automatically updates the data in that cache.

**Important:**

Only *admin* users can verify the Redis cache.

## Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
serverid	If P4 Code Review is connected to <a href="#">multiple P4 Servers</a> , serverid specifies which P4 Server the cache should be verified for.	string	path	See description	

Parameter	Description	Type	Parameter Type	Required	Default Value
context	<p>Add context to the call to verify a specific area of the Redis cache:</p> <ul style="list-style-type: none"><li>■ No context specified, verifies all of the Redis caches.</li><li>■ user verifies the Redis user cache.</li><li>■ group verifies the Redis group cache.</li><li>■ project verifies the Redis project cache.</li><li>■ workflow verifies the Redis workflow cache.</li></ul>	string	path	No	



Parameter	Description	Type	Parameter Type	Required	Default Value
	To verify multiple areas of cache, separate the entries with a comma ,				

### Example usage

#### Tip:

It might also be possible to use a password in place of the ticket if your P4 Server allows it.

To start verification of the user and group caches:

```
curl -u "username:ticket" -X POST
```

```
"https://myswarm.url/api/v9/cache/redis/verify?context=user,group"
```

JSON response

HTTP/1.1 200 OK

```
{
 isValid: true,
 data: {
 user: {
 state: "Queued",
 progress: false
 },
 group: {
 state: "Queued",
 progress: false
 }
 },
 messages: {
 "Verifying Redis cache for [user,group]"
 }
}
```

## Check progress of Redis cache verification

**Tip:**

Alternatively, use the **Cache Info** tab of the **System Information** page. Navigate to the **User id** dropdown menu, select **System Information**, and click the **Cache Info** tab. For more information, see "[Cache Info](#)" on page 719.

### Summary

Introduced for P4 Code Review 2019.2 and later, check progress of the P4 Code Review Redis cache verification.

GET /serverid/api/v9/cache/redis/verify?context=user,group,project,workflow

### Description

Used to check the progress of Redis cache verification process. Use the context parameter to check verification of specific areas of the Redis cache, if no context is specified progress is checked for all areas of the entire Redis cache.

### Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
serverid	If P4 Code Review is connected to <a href="#">multiple P4 Servers</a> , serverid specifies which P4 Server cache verification is being checked for.	string	path	See description	

Parameter	Description	Type	Parameter Type	Required	Default Value
context	<p>Add context to the check verification progress for a specific area of the Redis cache:</p> <ul style="list-style-type: none"><li>■ No context specified, checks verification progress for all of the Redis caches.</li><li>■ user checks verification progress for the Redis user cache.</li><li>■ group checks verification progress for the Redis group cache.</li><li>■ project checks verification</li></ul>	string	path	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
	<p>progress for the Redis project cache.</p> <ul style="list-style-type: none"> <li>workflows checks verification progress for the Redis workflow cache.</li> </ul> <p>To check the verification progress for multiple areas of cache, separate the entries with a comma ,</p>				

### Example usage

#### Tip:

It might also be possible to use a password in place of the ticket if your P4 Server allows it.

To check verification progress for the user and group caches:

```
curl -u "username:ticket" "http://myswarm.url/api/v9/cache/redis/verify?context=user,group"
```

JSON response

HTTP/1.1 200 OK

```
{
 isValid: true,
 data: {
 user: {
 state: "Running",
```

```

 progress: "Step 2 of 5; Calculating checksums for Redis keys"
 },
 group: {
 state: "Running",
 progress: "Step 1 of 5; Getting Redis cache entries"
 }
}
}
}

```

**Tip:**

- Valid states for verify are; Not Queued, Queued, Running, and Failed.
- Valid states for progress are, Step x of y; (where x is the current step and y is the total number of steps) followed by:
  - Getting Redis cache entries
  - Calculating checksums for Redis keys
  - Calculating checksums for Perforce models
  - Clearing any extraneous Redis cache entries
  - Indexing any missing Perforce models into the Redis cache
  - Verification complete

## Redis cache delete

### Summary

Introduced for P4 Code Review 2019.2 and later, deletes the P4 Code Review Redis cache.

DELETE /serverid/api/v9/cache/redis?context=user,group,project,workflow

### Description

Used to delete the Redis cache. Use the context parameter to delete specific areas of the Redis cache, if no context is specified the entire Redis cache is deleted.

**Important:**

Only *admin* users can delete the Redis cache.

**Parameters**

Parameter	Description	Type	Parameter Type	Required	Default Value
serverid	If P4 Code Review is connected to <a href="#">multiple P4 Servers</a> , serverid specifies which P4 Server the cache should be deleted for.	string	path	See description	

---

Parameter	Description	Type	Parameter Type	Required	Default Value
context	<p>Add context to the call to delete a specific area of the Redis cache:</p> <ul style="list-style-type: none"><li>■ No context specified, deletes all of the Redis caches.</li><li>■ user deletes the Redis user cache.</li><li>■ group deletes the Redis group cache.</li><li>■ project deletes the Redis project cache.</li><li>■ workflow deletes the Redis workflow</li></ul>	string	path	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
-----------	-------------	------	----------------	----------	---------------

cache.

To delete multiple areas of cache, separate the entries with a comma ,

### Example response

When an invalid Redis cache context is specified:

If an invalid context is specified in the call:

HTTP/1.1 200 OK

```
{
 "isValid":true,
 "messages":[
 "Invalid context [usr], must contain only [user, group, project, workflow]",
]
}
```

### Example of usage

#### Tip:

It might also be possible to use a password in place of the ticket if your P4 Server allows it.

```
curl -u "username:ticket" -X DELETE
"https://myswarm.url/serverA/api/v9/cache/redis?context=user,group"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the cache deleted message:

HTTP/1.1 200 OK

```
{
 "isValid":true,
 "messages":[
 "Deleted key(s) [Swarm^serverA:user-populated-status, Swarm^serverA:group-
populated-status]",
]
}
```



```
]
}
```

## Changes : API controller providing a service for changes

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Get projects and branches affected by a given change id

### Summary

Get projects, and branches, affected by a given change id.

GET /api/v9/changes/{change}/affectsprojects

### Description

All authenticated users are able to use this API.

### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "change": {
 "id": "1050",
 "projects": {
 "jam": [
 "live",
 "main"
]
 }
 }
}
```

```
}
}
```

## Get default reviewers for a given change id

### Summary

Get default reviewers for a given change id.

GET /api/v9/changes/{change}/defaultreviewers

### Description

All authenticated users are able to use this API.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "change": {
 "id": "1050",
 "defaultReviewers": {
 "groups": {
 "group1": {"required": "1"},
 "group2": {}
 },
 "users": {
 "user1": {},
 "user2": {"required": "true"}
 }
 }
 }
}
```

## Perform checks on a change if enabled

### Summary

Performs checks on the change if enabled

GET /api/v9/changes/{id}/check

### Description

Performs checks on the change if workflow configuration requires it.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Change to check	string	form	Yes
type	The type of check. Must have a value of enforced, strict or shelve	string	form	Yes

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "status": "OK",
 "isValid": "true",
```

```
"messages": []
}
```

### Examples of usage

#### Carry out enforced checks on a change if configured

To check change 42:

```
curl -u "username:password" "https://my-swarm-
host/api/v9/changes/42/check?type=enforced"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "status": "OK",
 "isValid": "true",
 "messages": []
}
```

#### Carry out enforced checks on a change if configured. Example of a failed 'requires review'

To check change 42:

```
curl -u "username:password" "https://my-swarm-
host/api/v9/changes/42/check?type=enforced"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "status": "NO_REVIEW",
 "isValid": "false",
 "messages": ["Change [42] must be associated with a review"]
}
```

#### Carry out strict checks on a change if configured

To check change 42:

```
curl -u "username:password" "https://my-swarm-host/api/v9/changes/42/check?type=strict"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "status": "OK",
 "isValid": "true",
 "messages": []
}
```

## Comments : P4 Code Review Comments

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Get List of Comments

GET /api/v9/comments/

### Summary

Get List of Comments

### Description

List comments.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see "[Private projects](#)" on page 366.
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required	Default Value
after	A comment ID to seek to. Comments up to and including the specified ID are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.	integer	query	No	
max	Maximum number of comments to return. This does not guarantee that max comments are returned. It does guarantee that the number of comments returned won't exceed max.	integer	query	No	100

---

Parameter	Description	Type	Parameter Type	Required	Default Value
topic	Only comments for given topic are returned. Examples: reviews/1234, changes/1234 or jobs/job001234.	string	query	No	
context [version]	If a reviews/1234 topic is provided, limit returned comments to a specific version of the provided review.	integer	query	No	
ignoreArchived	Prevents archived comments from being returned. (v5+)	boolean	query	No	
tasksOnly	Returns only comments that have been flagged as tasks. (v5+)	boolean	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
taskStates	Limit the returned comments to ones that match the provided task state (one or more of open, closed, verified, or comment). (v5+)	array (of strings)	query	No	
fields	An optional comma-separated list (or array) of fields to show for each comment. Omitting this parameter or passing an empty value shows all fields.	string	query	No	

### Examples of successful responses

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "topic": "",
 "comments": {
 "51": {
 "id": 51,
 "attachments": [],
 "body": "Short loin ground round sin reprehensible, venison west participle triple.",
 "context": [],
```



```

 "edited": null,
 "flags": [],
 "likes": [],
 "taskState": "comment",
 "time": 1461164347,
 "topic": "reviews/885",
 "updated": 1461164347,
 "user": "bruno"
 }
},
"lastSeen": 51
}

```

**Note:**

lastSeen can often be used as an offset for pagination, by using the value in the **after** parameter of subsequent requests.

When no results are found, the **comments** array is empty:

HTTP/1.1 200 OK

```

{
 "topic": "jobs/job000011",
 "comments": [],
 "lastSeen": null
}

```

**Examples of usage****Listing comments**

To list comments:

```
curl -u "username:password" "https://my-swarm-
host/api/v9/comments?topic=reviews/911&max=2&fields=id,body,time,user"
```

P4 Code Review responds with a list of the first two comments for review 911 and a **lastSeen** value for pagination:

HTTP/1.1 200 OK

```

{
 "topic": "reviews/911",
 "comments": {
 "35": {
 "id": 35,

```

```
"body": "Excitation thunder cats intelligent man braid organic bitters.",
"time": 1461164347,
"user": "bruno"
},
"39": {
 "id": 39,
 "body": "Chamber tote bag butcher, shirk truffle mode shabby chic single-origin coffee.",
 "time": 1461164347,
 "user": "swarm_user"
}
},
"lastSeen": 39
}
```

#### Paginating a comment listing

To obtain the next page of a comments list (based on the previous example):

```
curl -u "username:password" "https://my-swarm-
host/api/v9/comments?topic=reviews/911&max=2&fields=id,body,time,user&after=39"
```

P4 Code Review responds with the second page of results, if any comments are present after the last seen comment:

HTTP/1.1 200 OK

```
{
 "topic": "reviews/911",
 "comments": {
 "260": {
 "id": 260,
 "body": "Reprehensible do lore flank ham hock.",
 "time": 1461164349,
 "user": "bruno"
 },
 "324": {
 "id": 324,
 "body": "Sinter lo-fi temporary, nihilist tote bag mustache swag consequence interest
flexible.",
 "time": 1461164349,
 "user": "bruno"
 }
 },
 "lastSeen": 324
}
```

## Send notification for comments

### Summary

Sends notification for comments

POST /api/v9/comments/notify

### Description

Sends notification for comments.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
topic	This is going to send a single notification for any comments that were not notified immediately for the user authenticated for a given topic they are posting for. Example: reviews/1234.	string	form	Yes

Parameter	Description	Type	Parameter Type	Required
preview	Set to true to check if there are comments waiting for a notification to be sent, the notification is not sent.	string	form	No

---

#### Examples of successful responses

Comment notification sent.

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "message": "Sending a notification for 3 comments",
 "code": 200
}
```

No comments are waiting for a notification.

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "message": "No comment notifications to send",
 "code": 200
}
```

#### Examples of usage

Sending notification for comments

To send a notification for all delayed comments:

```
curl -u "username:password" \
 -X POST \
```

```
-d "topic=reviews/911" \
"https://my-swarm-host/api/v9/comments/notify"
```

P4 Code Review responds with a message detailing a notification sent with the number of comments it applied to.

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "message": "Sending a notification for 3 comments",
 "code": 200
}
```

#### Checking to see if any comments are waiting for a notification to be sent

To check if there are any comments awaiting a notification without sending the notification, include `"preview=true"` in your request.

```
curl -u "username:password" \
-X POST \
-d "topic=reviews/911" \
-d "preview=true" \
"https://my-swarm-host/api/v9/comments/notify"
```

P4 Code Review responds with a message without sending a notification.

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "message": "Sending a notification for 3 comments",
 "code": 200
 "preview": true
}
```

## Add a Comment

### Summary

Add a Comment

POST /api/v9/comments/

### Description

Add a comment to a topic (such as a review or a job).

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required	Default Value
topic	Topic to comment on. Examples: reviews/1234, changes/1234 or jobs/job001234.	string	form	Yes	
body	Content of the comment.	string	form	Yes	
silenceNotification	If set to 'true' no notifications will ever be sent for this created comment	string	form	No	
delayNotification	If set to 'true' notifications will be delayed	string	form	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
taskState	Optional task state of the comment. Valid values when adding a comment are <code>comment</code> and <code>open</code> . This creates a plain comment or opens a task, respectively.	string	form	No	comment
flags[]	Optional flags on the comment. Typically set to <code>closed</code> to archive a comment.	array (of strings)	form	No	
context[file]	File to comment on. Valid only for <code>changes</code> and <code>reviews</code> topics. Example: <code>//depot/main/REAME.txt</code> .	string	form	No	
context[leftLine]	Left-side diff line to attach the inline comment to. Valid only for <code>changes</code> and <code>reviews</code> topics. If this is specified, <code>context[file]</code> must also be specified.	integer	form	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
context [rightLine]	Right-side diff line to attach the inline comment to. Valid only for <b>changes</b> and <b>reviews</b> topics. If this is specified, <b>context[file]</b> must also be specified.	integer	form	No	
context [content]	Optionally provide content of the specified line and its four preceding lines. This is used to specify a short excerpt of context in case the lines being commented on change during the review. When not provided, P4 Code Review makes an effort to build the content on its own - as this involves file operations, it could become slow.	array (of strings)	form	No	
context [version]	With a <b>reviews</b> topic, this field specifies which version to attach the comment to.	integer	form	No	

**Example of response**

Successful Response contains **Comment** entity:

HTTP/1.1 200 OK



```
{
 "comment": {
 "id": 42,
 "attachments": [],
 "body": "Best. Comment. EVER!",
 "context": [],
 "edited": null,
 "flags": [],
 "likes": [],
 "taskState": "comment",
 "time": 123456789,
 "topic": "reviews/2",
 "updated": 123456790,
 "user": "bruno"
 }
}
```

### Examples of usage

#### Create a comment on a review

To create a comment on a review:

```
curl -u "username:password" \
 -X POST \
 -d "topic=reviews/2" \
 -d "body=This is my comment. It is an excellent comment. It contains a beginning, a middle,
and an end." \
 "https://my-swarm-host/api/v9/comments"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "comment": {
 "id": 42,
 "attachments": [],
 "body": "This is my comment. It is an excellent comment. It contains a beginning, a middle,
and an end.",
 "context": [],
 "edited": null,
 "flags": [],
 "likes": [],
 "taskState": "comment",
 "time": 123456789,
 "topic": "reviews/2",
 "updated": 123456790,
```

```
"user": "username"
}
```

#### Open a task on a review

To create a comment on a review, and flag it as an open task:

```
curl -u "username:password" \
 -X POST \
 -d "topic=reviews/2" \
 -d "taskState=open" \
 -d "body=If you could go ahead and attach a cover page to your TPS report, that would be great." \
 "https://my-swarm-host/api/v9/comments"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "comment": {
 "id": 43,
 "attachments": [],
 "body": "If you could go ahead and attach a cover page to your TPS report, that would be great.",
 "context": [],
 "edited": null,
 "flags": [],
 "likes": [],
 "taskState": "open",
 "time": 123456789,
 "topic": "reviews/2",
 "updated": 123456790,
 "user": "username"
 }
}
```

## Edit a Comment

### Summary

Edit a Comment

PATCH /api/v9/comments/{id}

### Description

Edit a comment.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	ID of the comment to be edited	integer	path	Yes
topic	Topic to comment on. Examples: reviews/1234, changes/1234 or jobs/job001234.	string	form	No
body	Content of the comment.	string	form	Yes

Parameter	Description	Type	Parameter Type	Required
taskState	Optional task state of the comment. Note that certain transitions (such as moving from open to verified) are not possible without an intermediate step (addressed, in this case). Examples: comment (not a task), open, addressed, verified.	string	form	No
flags[]	Optional flags on the comment. Typically set to closed to archive a comment.	array (of strings)	form	No
silenceNotification	If set to 'true' no notifications will ever be sent for this edited comment	string	form	No
delayNotification	If set to 'true' notifications will be delayed	string	form	No

#### Example response

Successful Response contains Comment entity:

HTTP/1.1 200 OK

```
{
 "comment": {
 "id": 1,
 "attachments": [],
 "body": "Best. Comment. EVER!",
 "context": [],
 "edited": 123466790,
 "flags": [],
 "likes": [],
 "taskState": "comment",
 "time": 123456789,
 "topic": "reviews/42",
 "updated": 123456790,
 "user": "bruno"
 }
}
```

### Examples of usage

#### Edit and archive a comment on a review

To edit and archive a comment on a review:

```
curl -u "username:password" \
 -X PATCH \
 -d "flags[]=closed" \
 -d "body=This comment wasn't as excellent as I may have lead you to believe. A thousand
 apologies." \
 "https://my-swarm-host/api/v9/comments/42"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "comment": {
 "id": 42,
 "attachments": [],
 "body": "This comment wasn't as excellent as I may have lead you to believe. A thousand
 apologies.",
 "context": [],
 "edited": 123466790,
 "flags": ["closed"],
 "likes": [],
 "taskState": "comment",
 "time": 123456789,
 "topic": "reviews/2",
 "updated": 123456790,
 }
}
```

```
"user": "username"
}
```

#### Flag a task as addressed on a review

To flag an open task as addressed on a review:

```
curl -u "username:password" \
-X PATCH \
-d "taskState=addressed" \
"https://my-swarm-host/api/v9/comments/43"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "comment": {
 "id": 43,
 "attachments": [],
 "body": "If you could go ahead and attach a cover page to your TPS report, that would be great.",
 "context": [],
 "edited": 123466790,
 "flags": ["closed"],
 "likes": [],
 "taskState": "addressed",
 "time": 123456789,
 "topic": "reviews/2",
 "updated": 123456790,
 "user": "username"
 }
}
```

## Groups : P4 Code Review Groups

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Get List of Groups

### Summary

Get List of Groups

GET /api/v9/groups/

**Description**

Returns the complete list of groups in P4 Code Review.

**Parameters**

Parameter	Description	Type	Parameter Type	Required	Default Value
after	A group ID to seek to. Groups prior to and including the specified ID are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.	string	query	No	

---

Parameter	Description	Type	Parameter Type	Required	Default Value
max	Maximum number of groups to return. This does not guarantee that max groups are returned. It does guarantee that the number of groups returned won't exceed max.	integer	query	No	100
fields	An optional comma-separated list (or array) of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.	string	query	No	



Parameter	Description	Type	Parameter Type	Required	Default Value
keywords	Keywords to limit groups on. Only groups where the group ID, group name (if set), or description contain the specified keywords are returned.	string	query	No	
ignoreExcludeList	Determines if the list of groups has the <code>group_exclude_list</code> filter applied or not. Add the parameter to ignore the <code>group_exclude_list</code> filter.	boolean	query	No	

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "groups": [
 {
 "Group": "test-group",
 "MaxLockTime": null,
 "MaxResults": null,
 "MaxScanRows": null,

```

```
"Owners": [],
"PasswordTimeout": null,
"Subgroups": [],
"Timeout": 43200,
"Users": ["bruno"],
"config": {
 "description": "Our testing group",
 "emailAddress": "test-group3@host.domain",
 "emailFlags": [],
 "name": "Test Group",
 "useMailingList": true
}
}
```

## Examples of usage

### Listing groups

To list groups:

```
curl -u "username:password" "https://myswarm.url/api/v9/groups?keywords=test-
group&fields=Group,Owners,Users,config&max=2"
```

P4 Code Review responds with a list of groups:

HTTP/1.1 200 OK

```
{
 "groups": [
 {
 "Group": "test-group",
 "Owners": [],
 "Users": ["bruno"],
 "config": {
 "description": "Our testing group",
 "emailAddress": "test-group@host.domain",
 "emailFlags": {
 "reviews": "1",
 "commits": "0"
 },
 "name": "Test Group",
 "useMailingList": true
 }
 },
 {
 "Group": "test-group2",
```

```

 "Owners": [],
 "Users": ["bruno"],
 "config": {
 "description": "Our second testing group",
 "emailAddress": "test-group2@host.domain",
 "emailFlags": [],
 "name": "Test Group 2",
 "useMailingList": true
 }
 },
 "lastSeen": "test-group2"
}

```

#### Paginating the groups list

Based on the previous example, we can pass a lastSeen value of `test-group2` to see if there are any subsequent groups in P4 Code Review.

```
curl -u "username:password" "https://myswarm.url/api/v9/groups?keywords=test-group&fields=Group,config&max=2&lastSeen=test-group2"
```

P4 Code Review responds with a list of groups (minus the Owners and Users fields, as we haven't requested them):

HTTP/1.1 200 OK

```

{
 "groups": [
 {
 "Group": "test-group3",
 "config": {
 "description": "Our 3rd testing group",
 "emailAddress": "test-group3@host.domain",
 "emailFlags": [],
 "name": "Test Group 3",
 "useMailingList": true
 }
 }
],
 "lastSeen": "test-group3"
}

```

## Get Group Information

### Summary

Get Group Information

GET /api/v9/groups/{id}

**Description**

Retrieve information about a group.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Group ID	string	path	Yes
fields	An optional comma-separated list (or array) of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.	string	query	No

---

Parameter	Description	Type	Parameter Type	Required
ignoreExcludeList	Determines if the <code>group_exclude_list</code> filter is applied or not when getting the group information. If the filter is applied and the group specified is present in the <code>group_exclude_list</code> the group information is not included in the response. Add the parameter to ignore the <code>group_exclude_list</code> filter.	boolean	query	No

#### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "group": {
 "Group": "test-group",
 "MaxLockTime": null,
 "MaxResults": null,
 "MaxScanRows": null,
 "Owners": [],
 "PasswordTimeout": null,
 "Subgroups": [],
 "Timeout": 43200,
 "Users": ["bruno"],
 "config": {
 "description": "Our testing group",
```

```
 "emailFlags": [],
 "name": "Test Group"
 }
}
```

## Examples of usage

### Fetching a group

To fetch an individual group:

```
curl -u "username:password" "https://myswarm.url/api/v9/groups/my-group"
```

P4 Code Review responds with the group entity:

HTTP/1.1 200 OK

```
{
 "group": {
 "Group": "test-group",
 "LdapConfig": null,
 "LdapSearchQuery": null,
 "LdapUserAttribute": null,
 "MaxLockTime": null,
 "MaxResults": null,
 "MaxScanRows": null,
 "Owners": [],
 "Users": ["bruno"],
 "config": {
 "description": "Our testing group",
 "emailAddress": "test-group@host.domain",
 "emailFlags": {
 "reviews": "1",
 "commits": "0"
 },
 "name": "Test Group",
 "useMailingList": true
 }
 }
}
```

### Limiting returned fields

To limit the returned fields when fetching an individual group:

```
curl -u "username:password" "https://myswarm.url/api/v9/groups/my-group?fields=Group,Owners,Users,config"
```

P4 Code Review responds with the group entity:

HTTP/1.1 200 OK

```
{
 "group": {
 "Group": "test-group",
 "Owners": [],
 "Users": ["bruno"],
 "config": {
 "description": "Our testing group",
 "emailFlags": [],
 "name": "Test Group"
 }
 }
}
```

## Create a new Group

### Summary

Create a new Group

POST /api/v9/groups/

### Description

Creates a new group in P4 Code Review.

### Parameters

Parameter	Description	Type	Parameter Type	Required
Group	Group identifier string.	string	form	Yes

Parameter	Description	Type	Parameter Type	Required
Users	An optional array of group users.  *At least one of Users, Owners, or Subgroups is required.	array	form	No *See Description
Owners	An optional array of group owners.  *At least one of Users, Owners, or Subgroups is required.	array	form	No *See Description
Subgroups	An optional array of subgroups.  *At least one of Users, Owners, or Subgroups is required.	array	form	No *See Description
config[name]	An optional full name for the group.	string	form	No
config [description]	An optional group description.	string	form	No
config [emailAddress]	The email address for this group.	string	form	No



Parameter	Description	Type	Parameter Type	Required
config [emailFlags] [reviews]	Email members when a new review is requested.	boolean	form	No
config [emailFlags] [commits]	Email members when a change is committed.	boolean	form	No
config [useMailingList]	Whether to use the configured email address or expand individual members addresses.	boolean	form	No

**Example response****Successful Response:**

HTTP/1.1 200 OK

```
{
 "group": {
 "Group": "test-group",
 "MaxLockTime": null,
 "MaxResults": null,
 "MaxScanRows": null,
 "Owners": [],
 "PasswordTimeout": null,
 "Subgroups": [],
 "Timeout": null,
 "Users": ["alice"],
 "config": {
 "description": "Test test test",
 "emailAddress": "test-group@host.domain",
 "emailFlags": [],
 "name": "TestGroup",
 "useMailingList": true
 }
 }
}
```

```
}
}
```

## Example of usage

### Create a group

#### Important:

- Only users with *super* privileges in the P4 Server (**p4d**), or users with *admin* privileges in **p4d** versions 2012.1 or newer, can create groups.
- This API version is only capable of setting specific fields: Group, Users, Owners, Subgroups, config. Any other fields specified in the creation request are ignored.

To create a new group:

```
curl -u "username:password" \
 -X POST \
 -d "Group=my-group" \
 -d "Owners[]=alice" \
 -d "Owners[]=bob" \
 -d "Users[]=bruno" \
 -d "Users[]=user2" \
 -d "config[description]=This group is special to me." \
 -d "config[name]=My Group" \
 -d "config[emailFlags][reviews]=1" \
 -d "config[emailFlags][commits]=0" \
 -d "config[emailAddress]=my-group@host.domain" \
 -d "config[useMailingList]=false" \
 "https://myswarm.url/api/v9/groups"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the new group entity:

HTTP/1.1 200 OK

```
{
 "group": {
 "Group": "my-group",
 "MaxLockTime": null,
 "MaxResults": null,
 "MaxScanRows": null,
 "Owners": ["username"],
 "PasswordTimeout": null,
 "Subgroups": [],
 "Timeout": null,
 "Users": [],
 }
}
```

```
"config": {
 "description": "This group is special to me.",
 "emailFlags": {
 "reviews": "1",
 "commits": "0"
 },
 "name": "My Group",
 "useMailingList": true
}
```

## Edit a Group

### Summary

Edit a Group

PATCH /api/v9/groups/{id}

### Description

Change the settings of a group in P4 Code Review. Only super users and group owners can perform this action.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Group ID	string	path	Yes
Users	An optional array of group users.	array	form	No
Owners	An optional array of group owners.	array	form	No
Subgroups	An optional array of group subgroups.	array	form	No

Parameter	Description	Type	Parameter Type	Required
<code>config[name]</code>	An optional full name for the group.	string	form	No
<code>config[description]</code>	An optional group description.	string	form	No
<code>config[emailAddress]</code>	The email address for this group.	string	form	No
<code>config[emailFlags][commits]</code>	Email members when a change is committed.	boolean	form	No
<code>config[emailFlags][reviews]</code>	Email members when a new review is requested.	boolean	form	No
<code>config[useMailingList]</code>	Whether to use the configured email address or expand individual members addresses.	boolean	form	No

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "group": {
 "Group": "test-group",
```

```

"Users": [],
"Owners": [],
"Subgroups": [],
"config": {
 "description": "New Group Description",
 "name": "TestGroup",
 "useMailingList": true
}
}
}

```

### Example of usage

#### Editing a group

##### Important:

- Only users with *super* privileges in the P4 Server, or group owners, can edit groups.
- This API version is only capable of modifying specific fields: **Users**, **Owners**, **Subgroups**, **config**. Any other fields specified in the edit request are ignored.

Here is how to update the **name**, **description**, and **emailFlags** configuration of the group **my-group**:

```

curl -u "username:password" \
 -X PATCH \
 -d "config[description]=This group is special to me." \
 -d "config[name]=My Group" \
 -d "config[emailFlags][commits]=1" \
 "https://myswarm.url/api/v9/groups/my-group"

```

Assuming that the authenticated user has permission, P4 Code Review responds with the modified group entity:

HTTP/1.1 200 OK

```

{
 "group": {
 "Group": "my-group",
 "Users": [],
 "Owners": [],
 "Subgroups": [],
 "config": {
 "description": "This group is special to me.",
 "emailAddress": "test-group@host.domain",
 "emailFlags": {
 "reviews": "1",

```

```
 "commits": "1"
 },
 "name": "My Group",
 "useMailingList": true
}
}
```

## Delete a Group

### Summary

Delete a Group

DELETE /api/v9/groups/{id}

### Description

Delete a group. Only super users and group owners can perform this action.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Group ID.	string	path	Yes

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "id": "test-group"
}
```

## Example of usage

### Deleting a group

**Important:**

Only *super* users and group owners can delete groups.

```
curl -u "username:password" -X DELETE "https://myswarm.url/api/v9/groups/my-group"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the `id` of the deleted group:

HTTP/1.1 200 OK

```
{
 "id": "my-group"
}
```

## Login : P4 Code Review Login API

**Important:**

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Checking the 2FA authentication

### Summary

Checking the 2FA authentication

GET /api/v9/checkauth/

### Description

Checking the 2FA authentication

### Example response

**Successful Response:**

HTTP/1.1 200 OK

```
{
 "results": {
 "trigger": "GAAuth says yes!",
 "successMsg": "Second factor authentication approved."
 }
}
```

```
},
"code": 200
}
```

### Usage example

#### Checking the 2FA authentication

To Check User prompt input for 2FA

```
curl -u "username:password" "https://myswarm.url/api/v9/login/checkauth"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the next step in the 2FA login:

HTTP/1.1 200 OK

```
{
 "results": {
 "trigger": "GAuth says yes!",
 "successMsg": "Second factor authentication approved."
 },
 "code": 200
}
```

## Get List of 2FA Methods

### Summary

Get List of 2FA Methods

GET /api/v9/listmethods/

### Description

Returns the complete list of methods of 2FA.

### Example response

**Successful Response:**

HTTP/1.1 200 OK

```
{
 "results": {
 "methods": {
```



```

 "1": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
 },
 "2": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
 },
 "3": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
 },
 "4": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
 }
 },
 "option": {
 "persist": "option",
 "nextState": "init-auth"
 },
 "code": 200
}

```

### Usage example

#### Listing 2FA Methods

To list the 2FA methods:

```
curl -u "username:password" "https://myswarm.url/api/v9/login/listmethods"
```

P4 Code Review responds with a list of 2FA methods:

HTTP/1.1 200 OK

```

{
 "results": {
 "methods": {
 "1": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
 },
 "2": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
 },
 },
 },
}

```

```
"3": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
},
"4": {
 "methodName": "Method Name will be here",
 "methodDesc": "Method Description will be here"
}
},
"option": {
 "persist": "option",
 "nextState": "init-auth"
},
"code": 200
}
```

## Get the current effective user details

### Summary

Get the current effective user details

GET /api/v9/session

### Description

Get user logged in

### Example response

**Successful Response:**

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "messages": [],
 "user": {
 "User": "reviewer",
 "FullName": "Code Reviewer",
 "Email": "reviewer@swarm.local",
 "Type": "standard",
 "Password": "enabled"
 }
}
```

### Usage example

#### Getting the currently effective user details

To get session details:

```
curl -u "<username>:<ticket>" "http://myswarm.url/api/v9/session"
```

## Checking the 2FA authentication

POST /api/v9/checkauth/

### Summary

Checking the 2FA authentication

### Description

Checking the 2FA authentication

### Parameters

Parameter	Description	Type	Parameter Type	Required
token	The token from the user for their 2FA prompt.	string	form	Yes

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "results": {
 "trigger": "otp-generated|||GAuth says yes!",
 "successMsg": "Second factor authentication approved."
 },
 "code": 200
}
```

### Example usage

#### Checking the 2FA authentication

To Check User prompt input for 2FA

```
curl -u "username:password" \
-X POST \
-d "token=TOKEN" \
"https://myswarm.url/api/v9/login/checkauth"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the next step in the 2FA login:

HTTP/1.1 200 OK

```
{
 "results": {
 "trigger": "GAuth says yes!",
 "successMsg": "Second factor authentication approved."
 },
 "code": 200
}
```

### Initiating the 2FA authentication

#### Summary

Initiating the 2FA authentication

POST /api/v9/initauth/

#### Description

Initiating the 2FA authentication

#### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>method</code>	The Method in which you want to use.	string	form	Yes

---

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "results": {
 "trigger": "TriggerName",
 "successMsg": "Message from Authentication method"
 },
 "option": {
 "prompt": true,
 "nextState": "check-auth"
 },
 "code": 200
}
```

### Example usage

#### Initiating the 2FA authentication

To Initiate the user 2FA login:

```
curl -u "username:password" \
 -X POST
 -d "method=METHOD" \
 "https://myswarm.url/api/v9/login/initauth"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the next step in the 2FA login:

HTTP/1.1 200 OK

```
{
 "results": {
 "trigger": "TriggerName",
 "successMsg": "Message from Authentication method"
 },
 "option": {
 "prompt": true,
 "nextState": "check-auth"
 },
 "code": 200
}
```

## Login to P4 Code Review

### Summary

Login to P4 Code Review

POST /api/v9/login/

### Description

Login to P4 Code Review

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "messages": [],
 "user": {
 "User": "swarm.user",
 "FullName": "Swarm User",
 "Email": "swarm.user@mydomain.com",
 "Type": "standard",
 "Password": "enabled",
 "isAdmin": false,
 "isSuper": false
 }
}
```

#### Note:

In the event of a failed login attempt P4 Code Review responds with:

### Example usage

#### Logging in to swarm

To login:

```
curl -H "Content-Type: application/json" \
 -X POST \
 -u "super:<ticket>" \
 -d '{"username":"swarm.user","password":"1234"}' "http://myswarm.url/api/v9/login"
```

# Login to P4 Code Review with SAML

**Important:**  
P4 Code Review now supports P4 AS (HAS) as a Single Sign-On (SSO) provider. This helps to simplify configuration and create a more robust SSO solution.

## Summary

Login to P4 Code Review with SAML  
POST /api/v9/login/saml

## Description

Login to P4 Code Review with SAML

## Parameters

Parameter	Description	Type	Parameter Type	Required
redirect	<p>Options are:</p> <ul style="list-style-type: none"><li>■ redirect=true or not specified: P4 Code Review redirects the user to the HTTP_REFERER url or to the specified custom "logout_url" on page 605 if it has been set.</li><li>■ redirect=false : P4 Code Review does not redirect the user.</li></ul>	string	query	No

## Example usage

### Logging in to swarm with SAML

```
curl -u "super:<ticket>" \
-X POST \
"http://myswarm.url/api/v9/login/saml"
```

JSON response:

HTTP/1.1 302 OK

```
{
 "isValid": "true"
 "url": "<url to redirect to>"
}
```

### Logging in to swarm with SAML and redirect=false

```
curl -u "super:<ticket>" \
-X POST \
"http://myswarm.url/api/v9/login/saml?redirect=false"
```

JSON response:

HTTP/1.1 200 OK

```
{
 "isValid": "true",
}
```

## Logout of P4 Code Review with optional redirect

### Summary

Logout of P4 Code Review with optional redirect

POST /api/v9/logout/

### Description

Logout of P4 Code Review



## Examples responses

### Successful Response:

HTTP/1.1 302 OK

```
{
 "isValid": true,
 "messages": []
}
```

### Successful Response:

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "messages": []
}
```

**Note:**

In the event of a failed login attempt P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "isValid": false,
 "messages": ["Error message."]
}
```

## Example usage

### Logout of P4 Code Review

To logout:

```
curl -X POST \
 -u "super:<ticket>" \
 "http://myswarm.url/api/v9/logout"
```

### Logout of P4 Code Review without redirect

To logout without any redirect:

```
curl -X POST \
-u "super:<ticket>" \
"http://myswarm.url/api/v9/logout?stay=true"
```

## Create a new P4 Code Review session using the given credentials

### Summary

Create a new P4 Code Review session using the given credentials

POST /api/v9/session

### Description

Login to swarm

### Example response

Successful Response:

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "messages": [],
 "user": {
 "User": "reviewer",
 "FullName": "Code Reviewer",
 "Email": "reviewer@swarm.local",
 "Type": "standard",
 "Password": "enabled"
 }
}
```

### Example usage

#### Logging in to P4 Code Review

To login:

```
curl -H "Content-Type: application/json" \
-X POST \
-d '{"username": "<username>", "password": "<password>", "remember": "false"}' \
-X POST http://localhost/api/v9/session
```

## Destroy the current session, for instance logout

### Summary

Destroy the current session, for instance logout

DELETE /api/v9/session

### Description

Logout of P4 Code Review

### Example response

**Successful Response:**

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "messages": []
}
```

### Example usage

Logging out of swarm

To login:

```
curl -u "<username>:<ticket>" -X DELETE "http://myswarm.url/api/v9/session"
```

## Projects : P4 Code Review Projects

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Get List of Projects

### Summary

Get List of Projects

GET /api/v9/projects/

## Description

Returns a list of projects in P4 Code Review that are visible to the current user. Administrators will see all projects, including private ones.

### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

## Parameters

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show for each project. Omitting this parameter or passing an empty value shows all fields.	string	query	No
workflow	An optional parameter to only list projects using a workflow.	string	query	No

## Example response

### Successful Response:

HTTP/1.1 200 OK

```
{
 "project": {
```

```

 "id": "jplugin",
 "branches": [],
 "defaults": {
 "reviewers": {
 "users": {
 "allison.clayborne": {
 "required": true
 },
 "jack.boone": [],
 "steve.russell": []
 }
 }
 },
 "deleted": false,
 "deploy": {
 "enabled": false,
 "url": ""
 },
 "description": "A Java plugin for continuous integration.",
 "emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
 },
 "jobview": "",
 "members": [
 "allison.clayborne",
 "jack.boone",
 "steve.russell"
],
 "minimumUpVotes": 1,
 "name": "JPlugin",
 "owners": [
 "allison.clayborne"
],
 "private": false,
 "retainDefaultReviewers": false,
 "subgroups": [],
 "tests": {
 "enabled": false,
 "url": "",
 "postBody": "",
 "postFormat": "URL"
 },
 "workflow": null
 },
 "readme": "<h1>P4 Plugin</h1>\n<p>Jenkins plugin for a Perforce Helix Versioning Engine (P4D).</p>\n<h2>Contents</h2>\n\nRelease notes\n\>"

```

}

### Example usage

## Listing projects

To list all projects:

```
curl -u "username:password" "https://my-swarm-
host/api/v9/projects?fields=id,description,members,name"
```

Pagination is not currently supported by this endpoint. P4 Code Review responds with a list of all projects:

HTTP/1.1 200 OK

```
{
 "projects": [
 {
 "id": "blue-book",
 "description": "This is a private project for use only by users with sufficient clearance.",
 "members": [
 "alex.randolph",
 "allison.clayborne"
],
 "name": "Blue Book"
 },
 {
 "id": "jplugin",
 "description": "A Java plugin for continuous integration.",
 "members": [
 "allison.clayborne",
 "jack.boone",
 "steve.russell"
],
 "name": "JPlugin"
 },
 {
 "id": "mercury",
 "description": "Mercury project for code management.",
 "members": [
 "alex.randolph",
 "claire.brevia",
 "jack.boone"
],
 "name": "Mercury"
 }
]
}
```

```

 "id": "test-data",
 "description": "",
 "members": [
 "steve.russell"
],
 "name": "Test Data"
 },
 {
 "id": "test-project-for-workflow",
 "description": "This project has been created to test Swarm workflow.",
 "members": [
 "alex.randolph",
 "allison.clayborne",
 "claire.brevia",
 "jack.boone",
 "paula.boyle",
 "steve.russell"
],
 "name": "Test project for workflow"
 }
]
}

```

Project administrators wishing to see the `tests` and `deploy` fields must fetch projects individually.

## Get Project Information

### Summary

Get Project Information

GET /api/v9/projects/{id}

### Description

Retrieve information about a project.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Project ID	string	path	Yes
fields	An optional comma-separated list (or array) of fields to show for each project. Omitting this parameter or passing an empty value shows all fields.	string	query	No

**Example response****Successful Response:**

HTTP/1.1 200 OK

```
{
 "project": {
 "id": "jplugin",
 "branches": [
 {
 "id": "main",
 "name": "MAIN",
 "workflow": null,
 "paths": [
 "//projects/jplugin/main/..."
],
 "defaults": {
 "reviewers": {
 "users": {
 "steve.russell": []
 }
 }
 }
 },
],
 "minimumUpVotes": null,
 "retainDefaultReviewers": false,
 }
}
```



```

 "moderators": [
 "claire.brevia"
],
 "moderators-groups": []
 },
 {
 "id": "candidate",
 "name": "CANDIDATE",
 "workflow": null,
 "paths": [
 "//projects/jplugin/candidate/..."
],
 "defaults": {
 "reviewers": []
 },
 "minimumUpVotes": null,
 "retainDefaultReviewers": false,
 "moderators": [],
 "moderators-groups": []
 }
],
"defaults": {
 "reviewers": {
 "users": {
 "allison.clayborne": {
 "required": true
 },
 "jack.boone": [],
 "steve.russell": []
 }
 }
},
"deleted": false,
"deploy": {
 "enabled": false,
 "url": ""
},
"description": "A Java plugin for continuous integration.",
"emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
},
"jobview": "",
"members": [
 "allison.clayborne",
 "jack.boone",
 "steve.russell"
],

```

```
"minimumUpVotes": 1,
"name": "JPlugin",
"owners": [
 "allison.clayborne"
],
"private": false,
"retainDefaultReviewers": false,
"subgroups": [],
"tests": {
 "enabled": false,
 "url": "",
 "postBody": "",
 "postFormat": "URL"
},
"workflow": null
},
"readme": "<h1>P4 Plugin</h1>\n<p>Jenkins plugin for a Perforce Helix Versioning Engine (P4D).</p>\n<h2>Contents</h2>\n\nRelease notes\n\n"
```

### Example usage

#### Fetching a project

To fetch an individual project:

```
curl -u "username:password" "https://my-swarm-host/api/v9/projects/testproject2?fields=id,description,members,name"
```

P4 Code Review responds with a project entity:

HTTP/1.1 200 OK

```
{
 "project": {
 "id": "jplugin",
 "description": "A Java plugin for continuous integration.",
 "members": [
 "allison.clayborne",
 "jack.boone",
 "steve.russell"
],
 "name": "JPlugin"
 }
}
```

Project administrators have access to additional fields (`tests` and `deploy`) when fetching individual projects using this endpoint.

## Create a new Project

### Summary

Create a new Project

POST `/api/v9/projects/`

### Description

Creates a new project in P4 Code Review.

### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>name</code>	Project Name (is also used to generate the Project ID)	string	form	Yes
<code>members</code>	An array of project members.	array	form	Yes
<code>subgroups</code>	An optional array of project subgroups.	array	form	No
<code>owners</code>	An optional array of project owners.	array	form	No
<code>description</code>	An optional project description.	string	form	No
<code>private</code>	Private projects are visible only to Members, Moderators, Owners, and Administrators. (Default: false)	boolean	form	No

Parameter	Description	Type	Parameter Type	Required
<code>deploy</code>	Configuration for automated deployment. Example: {"enabled": true, "url": "http://localhost/?change={change}"}	array	form	No
<code>tests</code>	Configuration for testing/continuous integration.	array	form	No
<code>branches</code>	Optional branch definitions for this project.	array	form	No
<code>jobview</code>	An optional jobview for associating certain jobs with this project.	string	form	No
<code>emailFlags[change_email_project_users]</code>	Email members, moderators and followers when a change is committed.	boolean	form	No
<code>emailFlags[review_email_project_members]</code>	Email members and moderators when a new review is requested.	boolean	form	No
<code>defaults</code>	An optional array of defaults at a project level (for example default reviewers).	array	form	No

Parameter	Description	Type	Parameter Type	Required
retainDefaultReviewers	An optional to retain the default reviewers.	boolean	form	No
minimumUpVotes	An optional to set the minimum number of up votes required.	string	form	No

### Example responses

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "project": {
 "id": "testproject",
 "branches": [],
 "defaults": {
 "reviewers": []
 },
 "deleted": false,
 "deploy": {
 "enabled": false,
 "url": null
 },
 "description": "My project is great",
 "emailFlags": [],
 "jobview": null,
 "members": [
 "bruno"
],
 "minimumUpVotes": null,
 "name": "testProject",
 "owners": [],
 "private": false,
 "retainDefaultReviewers": false,
 "subgroups": [],
 "tests": {
 "enabled": false,
 "url": null
 }
 },
}
```

```
"workflow": null
},
"readme": "",
"mode": "add"
}
```

**Successful Response:**

HTTP/1.1 200 OK

```
{
 "project": {
 "id": "project-x",
 "branches": [],
 "defaults": {
 "reviewers": {
 "users": {
 "bruno": {
 "required": "1"
 }
 }
 }
 },
 "deleted": false,
 "deploy": {
 "enabled": false,
 "url": null
 },
 "description": "This project is temporary until project Z replaces it",
 "emailFlags": [],
 "jobview": null,
 "members": [
 "swarm"
],
 "minimumUpVotes": null,
 "name": "project x",
 "owners": [],
 "private": false,
 "retainDefaultReviewers": false,
 "subgroups": [],
 "tests": {
 "enabled": false,
 "url": null
 },
 "workflow": null
 },
 "readme": "",
}
```

```
"mode": "add"
}
```

### Example usage

#### Creating a new project

To create a project:

```
curl -u "username:password" \
 -X POST \
 -d "name=TestProject 3" \
 -d "description=The third iteration of our test project." \
 -d "members[]=alice" \
 -d "members[]=bob" \
 "https://my-swarm-host/api/v9/projects/"
```

P4 Code Review responds with the new project entity:

HTTP/1.1 200 OK

```
{
 "project": {
 "id": "testproject-3",
 "branches": [],
 "defaults": {
 "reviewers": []
 },
 "deleted": false,
 "deploy": {
 "enabled": false,
 "url": null
 },
 "description": "The third iteration of our test project.",
 "emailFlags": [],
 "jobview": null,
 "members": [
 "bruno",
 "swarm"
],
 "minimumUpVotes": null,
 "name": "TestProject 3",
 "owners": [],
 "private": false,
 "retainDefaultReviewers": false,
 "subgroups": [],
 "tests": {
 "enabled": false,
```

```
"url": null
},
"workflow": null
},
"readme": "",
"mode": "add"
}
```

#### Creating a private project with branches

Specifying a branch requires using array notation and providing at least two fields (**name** and **paths**) for each branch you wish to create. Creating more than one branch involves incrementing the **branches[0]** specifier for each branch - an example of this accompanies the PATCH endpoint documentation.

Projects are public by default. To mark a project as private, set the **private** parameter to **true**.

```
curl -u "username:password" \
-X POST \
-d "name=TestProject 4" \
-d "description=The 4th iteration of our test project." \
-d "private=true" \
-d "members[]=bob" \
-d "branches[0][name]=Branch One" \
-d "branches[0][paths][]=//depot/main/TestProject/..." \
"https://my-swarm-host/api/v9/projects"
```

P4 Code Review responds with the new project entity:

HTTP/1.1 200 OK

```
{
 "project": {
 "id": "testproject-4",
 "branches": [],
 "defaults": {
 "reviewers": []
 },
 "deleted": false,
 "deploy": {
 "enabled": false,
 "url": null
 },
 "description": "The 4th iteration of our test project.",
 "emailFlags": [],
 "jobview": null,
 "members": [
 "bruno"
],
 },
}
```



```

"minimumUpVotes": null,
"name": "TestProject 4",
"owners": [],
"private": true,
"retainDefaultReviewers": false,
"subgroups": [],
"tests": {
 "enabled": false,
 "url": null
},
"workflow": null
},
"readme": "",
"mode": "add"
}

```

## Edit a Project

### Summary

Edit a Project

PATCH /api/v9/projects/{id}

### Description

Change the settings of a project in P4 Code Review. If a project has owners set, only the owners can perform this action.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>id</code>	Project ID	string	path	Yes

Parameter	Description	Type	Parameter Type	Required
name	Project Name (changing the project name does not change the project ID)	string	form	No
members	An array of project members.	array	form	No
subgroups	An optional array of project subgroups.	array	form	No
owners	An optional array of project owners.	array	form	No
description	Your project description.	string	form	No
private	Private projects are visible only to Members, Moderators, Owners, and Administrators. (Default: false)	boolean	form	No
deploy	Configuration for automated deployment. Example: {"enabled": true, "url": "http://localhost/?change={change}"}	array	form	No
tests	Configuration for testing/continuous integration.	array	form	No

Parameter	Description	Type	Parameter Type	Required
branches	Optional branch definitions for this project.	array	form	No
jobview	A jobview for associating certain jobs with this project.	string	form	No
emailFlags [change_email_ project_users]	Email members, moderators and followers when a change is committed.	boolean	form	No
emailFlags[review_ email_project_ members]	Email members and moderators when a new review is requested.	boolean	form	No
defaults	An optional array of defaults at a project level (for example default reviewers).	array	form	No
retainDefaultReviewers	An optional to retain the default reviewers.	boolean	form	No
minimumUpVotes	An optional to set the minimum number of up votes required.	string	form	No

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
```

```
"project":{
 "id":"project1",
 "branches":[
 {
 "name":"main",
 "paths":[
 "//depot/dev/APIProject/..."
],
 "id":"main",
 "moderators":[],
 "moderators-groups":[],
 "defaults":{
 "reviewers":[]
 },
 "workflow":null,
 "retainDefaultReviewers":false,
 "minimumUpVotes":null
 }
],
 "defaults":{
 "reviewers":[]
 },
 "deleted":false,
 "deploy":{
 "enabled":false,
 "url":""
 },
 "description":"Updated description",
 "emailFlags":[],
 "jobview": "",
 "members":[
 "swarm-admin"
],
 "minimumUpVotes":null,
 "name":"API Project",
 "owners":[],
 "private":false,
 "retainDefaultReviewers":false,
 "subgroups":[],
 "tests":{
 "enabled":false,
 "url":""
 },
 "workflow":null
},
"readme": "",
"mode":"edit"
}
```

## Example usage

### Editing a project

To edit a project:

**Note:**

It is safe to edit a project without specifying branches, but the instructions for adding branches contain important information for modifying branch configuration.

```
curl -u "username:password" \
-X PATCH
-d "description=Witness the power of a fully operational Swarm project." \
"https://my-swarm-host/api/v9/projects/jam"
```

P4 Code Review responds with the updated project entity:

HTTP/1.1 200 OK

```
{
 "project":{
 "id":"jam",
 "branches":[
 {
 "id":"main",
 "name":"Main",
 "workflow":null,
 "paths":[
 "\Vdepot\Jam\MAINV..."
],
 "defaults":{
 "reviewers":[]
 },
 "minimumUpVotes":null,
 "retainDefaultReviewers":false,
 "moderators":[],
 "moderators-groups":[]
 },
 {
 "id":"release-2-1",
 "name":"Release 2.1",
 "workflow":null,
 "paths":[
 "\Vdepot\Jam\VREL2.1V..."
],
 "defaults":{
 "reviewers":[]
 },
 },
]
 }
}
```

```
"minimumUpVotes":null,
"retainDefaultReviewers":false,
"moderators":[],
"moderators-groups":[]
},
{
 "id":"release-2-2",
 "name":"Release 2.2",
 "workflow":null,
 "paths":[
 "VVdepotVJamVREL2.2V..."
],
 "defaults":{
 "reviewers":[]
 },
 "minimumUpVotes":null,
 "retainDefaultReviewers":false,
 "moderators":[],
 "moderators-groups":[]
}
],
"defaults":{
 "reviewers":[]
},
"deleted":false,
"deploy":{
 "enabled":false,
 "url":""
},
"description":"Witness the power of a fully operational Swarm project.",
"emailFlags":{
 "change_email_project_users":"1",
 "review_email_project_members":"1"
},
"jobview":"","
"members":[
 "alex_qc",
 "pubuser"
],
"minimumUpVotes":null,
"name":"Jam",
"owners":[
 "bruno"
],
"private":false,
"retainDefaultReviewers":false,
"subgroups":[
 "Administrators"
```

```

],
 "tests":{
 "enabled":false,
 "url": "",
 "postBody": "",
 "postFormat": "URL"
 },
 "workflow": "1"
 },
 "readme": "",
 "mode": "edit"
}

```

#### Editing a project to add a moderated branch and make the project public

Specifying a branch requires using array notation and providing at least two fields (**name** and **paths**) for each branch you wish to create. Creating more than one branch involves incrementing the **branches[0]** specifier for each branch.

#### Important:

If you have existing branches, you must specify all of them in the query to avoid data loss. This operation sets the value of the entire **branches** property to match the provided input.

To change a private project to public, set the **private** parameter to **false**.

```

curl -u "username:password" \
 -X PATCH \
 -d "private=false" \
 -d "branches[0][name]=Branch One" \
 -d "branches[0][paths][]=//depot/main/TestProject/..." \
 -d "branches[1][name]=Branch Two" \
 -d "branches[1][paths][]=//depot/main/SecondBranch/..." \
 -d "branches[1][moderators][]=bob" \
 -d "branches[1][moderators-groups][]=group1" \
 "https://my-swarm-host/api/v9/projects/testproject-4"

```

P4 Code Review responds with the new project entity:

HTTP/1.1 200 OK

```

{
 "project":{
 "id":"testproject-4",
 "branches":[
 {
 "name":"Branch One",
 "paths":[
 "///depot/main/TestProject/..."

```

```
],
 "id": "branch-one",
 "moderators": [],
 "moderators-groups": [],
 "defaults": {
 "reviewers": []
 },
 "workflow": null,
 "retainDefaultReviewers": false,
 "minimumUpVotes": null
 },
 {
 "name": "Branch Two",
 "paths": [
 "//depot/main/SecondBranch/..."
],
 "moderators": [
 "non-admin"
],
 "moderators-groups": [
 "group1"
],
 "id": "branch-two",
 "defaults": {
 "reviewers": []
 },
 "workflow": null,
 "retainDefaultReviewers": false,
 "minimumUpVotes": null
 }
],
"defaults": {
 "reviewers": []
},
"deleted": false,
"deploy": {
 "enabled": false,
 "url": ""
},
"description": "Updated description",
"emailFlags": [],
"jobview": null,
"members": [
 "non-admin"
],
"minimumUpVotes": null,
"name": "testproject-4",
"owners": [],
```



```

 "private":false,
 "retainDefaultReviewers":false,
 "subgroups":[],
 "tests":{
 "enabled":false,
 "url":""
 },
 "workflow":null
 },
 "readme":"","
 "mode":"edit"
}

```

#### Editing a project to add default reviewers

```

curl -u "<user>:<password>" \
 -H "Content-Type: application/json" \
 -X PATCH \
 -d '{"defaults":{"reviewers":{"swarm_admin":{"required":true}} \
 "branches":[{" \
 "name":"dev", \
 "paths":["//depot/dev/API Project/..."] \
 "defaults": { \
 "reviewers":{" \
 "groups":{"group1":{"required":"1"}}, \
 "users":{"non_admin":{"required":true}} \
 } \
 }}] \
 "https://my-swarm-host/api/v9/projects/API Project"

```

Or without JSON content:

```

curl -u "username:password" \
 -X PATCH \
 -d "name=Udated description" \
 -d "defaults[reviewers][swarm_admin][required]=true" \
 -d "branches[0][name]=dev" \
 -d "branches[0][paths][]="//depot/dev/API Project/..." \
 -d "branches[0][defaults][reviewers][groups][group1][required]=1" \
 -d "branches[0][defaults][reviewers][users][non_admin][required]=true" \
 "https://my-swarm-host/api/v9/projects/API Project"

```

P4 Code Review responds with project entity similar to:

HTTP/1.1 200 OK

```

{
 "project":{

```

```
"id": "api-project",
"branches": [
 {
 "name": "dev",
 "paths": [
 "//depot/dev/APIProject/..."
],
 "defaults": {
 "reviewers": {
 "users": {
 "non-admin": {
 "required": true
 }
 },
 "groups": {
 "group1": {
 "required": "1"
 }
 }
 }
 },
 "id": "dev",
 "moderators": [],
 "moderators-groups": [],
 "workflow": null,
 "retainDefaultReviewers": false,
 "minimumUpVotes": null
 }
],
"defaults": {
 "reviewers": {
 "users": {
 "swarm-admin": {
 "required": true
 }
 }
 }
},
"deleted": false,
"deploy": {
 "enabled": false,
 "url": ""
},
"description": "Updated description",
"emailFlags": [],
"jobview": null,
"members": [
 "swarm-admin"
```

```

],
"minimumUpVotes":null,
"name":"API Project",
"owners":[],
"private":false,
"retainDefaultReviewers":false,
"subgroups":[],
"tests":{
 "enabled":false,
 "url":""
},
"workflow":null
},
"readme":"","
"mode":"edit"
}

```

## Delete a Project

### Summary

Delete a Project

DELETE /api/v9/projects/{id}

### Description

Mark a P4 Code Review project as deleted. The project ID and name cannot be reused. If a project has owners set, only the owners can perform this action.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>id</code>	Project ID	string	path	Yes

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "id": "testproject"
}
```

### Example usage

#### Deleting a project:

Super users, administrators, and owners can delete projects. Members can delete projects that have no owners set.

```
curl -u "username:password" -X DELETE "https://my-swarm-host/api/v9/projects/testproject3"
```

Assuming that the authenticated user has permission, P4 Code Review responds with the id of the deleted project:

HTTP/1.1 200 OK

```
{
 "id": "testproject3"
}
```

## Reviews : P4 Code Reviews

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Get reviews for action dashboard

### Summary

Get reviews for action dashboard

GET /api/v9/dashboards/action

### Description

Gets reviews for the action dashboard for the authenticated user.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Example response**

Successful Commit contains Review and Commit Entities:

HTTP/1.1 200 OK

```
{
 "lastSeen": 120,
 "reviews": [
 {
 "id": 7,
 "author": "swarm_admin",
 "changes": [6],
 "comments": [0,0],
 "commits": [6],
 "commitStatus": [],
 "created": 1485793976,
 "deployDetails": [],
 "deployStatus": null,
 "description": "test\n",
 "groups": ["swarm-project-test"],
 "participants": {"swarm_admin":[]},
 "pending": false,
 "projects": {"test":["test"]},
 "roles": ["moderator|reviewer|required_reviewer|author"],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1485958875,
 "updateDate": "2017-02-01T06:21:15-08:00"
 }
],
 "totalCount": null
}
```

## Example usage

### Getting reviews for the action dashboard

To list reviews:

```
curl -u "username:password" "http://my-swarm-host/api/v9/dashboards/action"
```

P4 Code Review responds with a list of the latest reviews, a `totalCount` field, and a `lastSeen` value for pagination:

HTTP/1.1 200 OK

```
{
 "lastSeen": 120,
 "reviews": [
 {
 "id": 7,
 "author": "swarm_admin",
 "changes": [6],
 "comments": [0,0],
 "commits": [6],
 "commitStatus": [],
 "created": 1485793976,
 "deployDetails": [],
 "deployStatus": null,
 "description": "test\n",
 "groups": ["swarm-project-test"],
 "participants": {"swarm_admin":[]},
 "pending": false,
 "projects": {"test":["test"]},
 "roles": ["moderator|reviewer|required_reviewer|author"],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1485958875,
 "updateDate": "2017-02-01T06:21:15-08:00"
 }
],
 "totalCount": null
}
```

## Get List of Reviews

### Summary

Get List of Reviews

GET /api/v9/reviews/

### Description

List and optionally filter reviews.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required	Default Value
after	<p>A review ID to seek to. Reviews up to and including the specified id are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.</p> <p>Reviews are returned in the order they were created in, starting with the most recently created review.</p>	integer	query	No	

---



Parameter	Description	Type	Parameter Type	Required	Default Value
max	Maximum number of reviews to return. This does not guarantee that max reviews are returned. It does guarantee that the number of reviews returned won't exceed max. Server-side filtering may exclude some reviews for permissions reasons.	integer	query	No	1000
fields	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
author[]	One or more authors to limit reviews by. Reviews with any of the specified authors are returned. (v1.2+)	array (of strings)	query	No	
change[]	One or more change IDs to limit reviews by. Reviews associated with any of the specified changes are returned.	array (of integers)	query	No	

---

Parameter	Description	Type	Parameter Type	Required	Default Value
hasReviewers	Boolean option to limit to reviews to those with or without reviewers. Use true or false for JSON-encoded data, 1 for true and or 0 for false for form-encoded data. The presence of the parameter without a value is evaluated as true.	boolean	query	No	
ids[]	One or more review IDs to fetch. Only the specified reviews are returned. This filter cannot be combined with the max parameter.	array (of integers)	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
keywords	Keywords to limit reviews by. Only reviews where the description, participants list or project list contain the specified keywords are returned.	string	query	No	
participants[]	One or more participants to limit reviews by. Reviews with any of the specified participants are returned.	array (of strings)	query	No	
project[]	One or more projects to limit reviews by. Reviews affecting any of the specified projects are returned.	array (of strings)	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
state[]	One or more states to limit reviews by. Reviews in any of the specified states are returned.	array (of strings)	query	No	
passesTests	Boolean option to limit reviews by tests passing or failing. Use <code>true</code> or <code>false</code> for JSON-encoded data, <code>1</code> for true and <code>0</code> for false for form-encoded data. The presence of the parameter without a value is evaluated as true.	string	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
notUpdatedSince	Option to fetch unchanged reviews. Requires the date to be in the format YYYY-mm-dd, for example 2017-01-01. Reviews to be returned are determined by looking at the last updated date of the review.	string	query	No	
hasVoted	Should have the value 'up' or 'down' to filter reviews that have been voted up or down by the current authenticated user.	string	query	No	

---

Parameter	Description	Type	Parameter Type	Required	Default Value
myComments	True or false to support filtering reviews that include comments by the current authenticated user.	boolean	query	No	

### Examples responses

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "lastSeen": 12209,
 "reviews": [
 {
 "id": 12206,
 "author": "swarm",
 "changes": [12205],
 "comments": 0,
 "commits": [],
 "commitStatus": [],
 "created": 1402507043,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Review Description\n",
 "participants": {
 "swarm": []
 },
 "pending": true,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1402518492
 }
]
}
```

```
}
],
"totalCount": 1
}
```

**Note:**

P4 Code Review returns `null` for `totalCount` if no search filters were provided. `lastSeen` can often be used as an offset for pagination, by using the value in the `after` parameter of subsequent requests.

When no results are found, the `reviews` array is empty:

HTTP/1.1 200 OK

```
{
 "lastSeen": null,
 "reviews": [],
 "totalCount": 0
}
```

**Example usage****Listing reviews**

To list reviews:

```
curl -u "username:password" "https://my-swarm-
host/api/v9/reviews?max=2&fields=id,description,author,state"
```

P4 Code Review responds with a list of the latest reviews, a `totalCount` field, and a `lastSeen` value for pagination:

HTTP/1.1 200 OK

```
{
 "lastSeen": 120,
 "reviews": [
 {
 "id": 123,
 "author": "bruno",
 "description": "Adding .jar that should have been included in r110\n",
 "state": "needsReview"
 },
 {
 "id": 120,
 "author": "bruno",
```



```
 "description": "Fixing a typo.\n",
 "state": "needsReview"
 },
],
"totalCount": null
}
```

The `totalCount` field is populated when keywords are supplied. It indicates how many total matches there are. If keywords are not supplied the `totalCount` field remains `null`, indicating that the list of all reviews is being queried.

#### Paginating a review listing

To obtain the next page of a reviews list (based on the previous example):

```
curl -u "username:password" "https://my-swarm-
host/api/v9/reviews?max=2&fields=id,description,author,state&after=120"
```

P4 Code Review responds with the second page of results, if any reviews are present after the last seen review:

HTTP/1.1 200 OK

```
{
 "lastSeen": 100,
 "reviews": [
 {
 "id": 110,
 "author": "bruno",
 "description": "Updating Java files\n",
 "state": "needsReview"
 },
 {
 "id": 100,
 "author": "bruno",
 "description": "Marketing materials for our new cutting-edge product\n",
 "state": "needsReview"
 }
],
 "totalCount": null
}
```

#### Finding reviews for a change or a list of changes

Given a list of change IDs (5, 6, 7), here is how to check if any of them have reviews attached:

```
curl -u "username:password" "https://my-swarm-
host/api/v9/reviews?max=2&fields=id,changes,description,author,state&change\
\[]=5&change\[]=6&change\[]=7"
```

P4 Code Review responds with a list of reviews that include these changes:

HTTP/1.1 200 OK

```
{
 "lastSeen": 100,
 "reviews": [
 {
 "id": 110,
 "author": "bruno",
 "changes": [5],
 "description": "Updating Java files\n",
 "state": "needsReview"
 },
 {
 "id": 100,
 "author": "bruno",
 "changes": [6,7],
 "description": "Marketing materials for our new cutting-edge product\n",
 "state": "needsReview"
 }
],
 "totalCount": 2
}
```

If no corresponding reviews are found, P4 Code Review responds with an empty reviews list:

HTTP/1.1 200 OK

```
{
 "lastSeen": null,
 "reviews": [],
 "totalCount": 0
}
```

**Finding inactive reviews (by checking the last updated date)**

```
curl -u "username:password" "https://my-swarm-
host/api/v9
/reviews?max=2&fields=id,changes,description,author,state¬UpdatedSince=2017-01-01"
```

P4 Code Review responds with a list of reviews that have not been updated since the notUpdatedSince date:

HTTP/1.1 200 OK

```
{
 "lastSeen": 100,
 "reviews": [
 {
 "id": 110,
 "author": "bruno",
 "changes": [5],
 "description": "Updating Java files\n",
 "state": "needsReview"
 },
 {
 "id": 100,
 "author": "bruno",
 "changes": [6,7],
 "description": "Marketing materials for our new cutting-edge product\n",
 "state": "needsReview"
 }
],
 "totalCount": 2
}
```

#### Finding reviews I have voted up

```
curl -u "username:password" "https://my-swarm-
host/api/v9/reviews?max=2&fields=id,changes,description,author,state&hasVoted=up"
```

P4 Code Review responds with a list of reviews that include these changes:

HTTP/1.1 200 OK

```
{
 "lastSeen": 100,
 "reviews": [
 {
 "id": 110,
 "author": "bruno",
 "changes": [5],
 "description": "Updating Java files\n",
 "state": "needsReview"
 },
 {
 "id": 100,
 "author": "bruno",
 "changes": [6,7],
 "description": "Marketing materials for our new cutting-edge product\n",
 "state": "needsReview"
 }
]
}
```

```
}
],
"totalCount": 2
}
```

If no corresponding reviews are found, P4 Code Review responds with an empty reviews list:

```
{
 "lastSeen": null,
 "reviews": [],
 "totalCount": 0
}
```

#### Finding reviews I have commented on (current authenticated user)

```
curl -u "username:password" "https://my-swarm-
host/api/v9/reviews?max=2&fields=id,changes,description,author,state&myComments=true"
```

P4 Code Review responds with a list of reviews that include these changes:

HTTP/1.1 200 OK

```
{
 "lastSeen": 100,
 "reviews": [
 {
 "id": 110,
 "author": "bruno",
 "changes": [5],
 "description": "Updating Java files\n",
 "state": "needsReview"
 },
 {
 "id": 100,
 "author": "bruno",
 "changes": [6,7],
 "description": "Marketing materials for our new cutting-edge product\n",
 "state": "needsReview"
 }
],
 "totalCount": 2
}
```

If no corresponding reviews are found, P4 Code Review responds with an empty reviews list:

HTTP/1.1 200 OK

```
{
 "lastSeen": null,
 "reviews": [],
 "totalCount": 0
}
```

## Get Review Information

### Summary

Get Review Information

GET /api/v9/reviews/{id}

### Description

Retrieve information about a review.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes
fields	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No

## Example usage

### Fetching a review

To fetch a review:

```
curl -u "username:password" "https://my-swarm-host/api/v9/reviews/123"
```

P4 Code Review responds with a review entity:

HTTP/1.1 200 OK

```
{
 "review": {
 "id": 123,
 "author": "bruno",
 "changes": [122,124],
 "commits": [124],
 "commitStatus": [],
 "created": 1399325913,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Adding .jar that should have been included in r110\n",
 "groups": [],
 "participants": {
 "alex_qc": [],
 "bruno": {
 "vote": 1,
 "required": true
 },
 "vera": []
 },
 "reviewerGroups": {
 "group1": [],
 "group2": {
 "required": true
 },
 "group3": {
 "required": true,
 "quorum": "1"
 }
 },
 "pending": false,
 "projects": {
 "swarm": ["main"]
 },
 "state": "archived",
 "stateLabel": "Archived",
 "testDetails": {
```

```

 "url": "http://jenkins.example.com/job/project_ci/123/"
 },
 "testStatus": null,
 "type": "default",
 "updated": 1399325913,
 "versions": [
 {
 "difference": 1,
 "stream": "//jam/main",
 "streamSpecDifference": 0,
 "change": 124,
 "user": "bruno",
 "time": 1568115902,
 "pending": true,
 "addChangeMode": "replace",
 "testRuns": [
 8
]
 },
 {
 "difference": 1,
 "stream": "//jam/main",
 "streamSpecDifference": 0,
 "change": 12157,
 "user": "bruno",
 "time": 1568115918,
 "pending": true,
 "addChangeMode": "replace",
 "archiveChange": 122,
 "testRuns": [
 1,
 2,
 3,
 4,
 5,
 6,
 7
]
 }
]
}

```

#### Fetching a review that does not exist

If you try to fetch a review that does not exist:

```
curl -u "username:password" "https://my-swarm-host/api/v9/reviews/9999999"
```

P4 Code Review responds with a 404 response:

HTTP/1.1 404 Not Found

```
{
 "error": "Not Found"
}
```

## Get transitions for a review

### Summary

Get transitions for a review

GET /api/v9/reviews/{id}/transitions

### Description

Get the allowed transitions for a review.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.



## Parameters

Parameter	Description	Type	Parameter Type	Required
upVoters	A list of users whose vote up will be assumed when determining the transitions. For example if a user has not yet voted but would be the last required vote and asked for possible transitions we would want to include 'approve'	string	query	No

## Example response

### Successful Response:

HTTP/1.1 200 OK

```
{
 "isValid": "true",
 "transitions": {
 "needsRevision": "Needs Revision",
 "approved": "Approve",
 "approved:commit": "Approve and Commit",
 "rejected": "Reject",
 "archived": "Archive"
 }
}
```

## Example usage

Getting transitions that a review with id 1 can transition to:

```
curl -u "username:password" "https://my-swarm-host/api/v9/reviews/1/transitions?upVoters=bruno"
```

The transitions that are returned depend on the current review state, P4 Code Review setup and the configuration of associated projects and branches

HTTP/1.1 200 OK

```
{
 "isValid": "true"
 "transitions": {
 "needsRevision": "Needs Revision",
 "approved": "Approve",
 "approved:commit": "Approve and Commit",
 "rejected": "Reject",
 "archived": "Archive"
 }
}
```

## Create a Review

### Summary

Create a Review

POST /api/v9/reviews/

### Description

Pass in a changelist ID to create a review. Optionally, you can also provide a description and a list of reviewers.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
change	Change ID to create a review from	integer	form	Yes
description	Description for the new review (defaults to change description)	string	form	No
reviewers	A list of reviewers for the new review	array (of strings)	form	No
requiredReviewers	A list of required reviewers for the new review (v1.1+)	array (of strings)	form	No
reviewerGroups	A list of required reviewers for the new review (v7+)	array	form	No

**Example response**

Successful Response contains Review Entity:

HTTP/1.1 200 OK

```
{
 "review": {
 "id": 12205,
 "author": "bruno",
 "changes": [10667],
 "commits": [10667],
 "commitStatus": [],
```

```
"created": 1399325913,
"deployDetails": [],
"deployStatus": null,
"description": "Adding .jar that should have been included in r10145\n",
"participants": {
 "bruno": []
},
"reviewerGroups": {
 "group1" : [],
 "group2" : {
 "required" : true
 },
 "group3" : {
 "required" : true,
 "quorum": "1"
 }
},
"pending": false,
"projects": [],
"state": "archived",
"stateLabel": "Archived",
"testDetails": [],
"testStatus": null,
"type": "default",
"updated": 1399325913
}
}
```

### Example usage

#### Starting a review

To start a review for a committed change or a non-empty shelved changelist specifying reviewer groups:

```
curl -u "username:password" \
-X POST \
-d "change=122" \
-d "reviewerGroups[0][name]=group1" \
-d "reviewerGroups[1][name]=group2" \
-d "reviewerGroups[1][required]=true" \
-d "reviewerGroups[2][name]=group3" \
-d "reviewerGroups[2][required]=true" \
-d "reviewerGroups[2][quorum]=1" \
"https://my-swarm-host/api/v9/reviews/"
```

P4 Code Review responds with the new review entity:

HTTP/1.1 200 OK

```
{
 "review": {
 "id": 123,
 "author": "bruno",
 "changes": [122],
 "commits": [],
 "commitStatus": [],
 "created": 1399325913,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Adding .jar that should have been included in r110\n",
 "groups": [],
 "participants": {
 "bruno": []
 },
 },
 "reviewerGroups": {
 "group1" : [],
 "group2" : {
 "required" : true
 },
 "group3" : {
 "required" : true,
 "quorum": "1"
 }
 },
 "pending": true,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1399325913,
 "versions": []
}
```

## Archiving the inactive reviews

### Summary

Archiving the inactive reviews (v6+)

POST /api/v9/reviews/archive/

### Description

Archiving reviews not updated since the date (v6+).

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
notUpdatedSince	Updated since date. Requires the date to be in the format YYYY-mm-dd, for example 2017-01-01	string	form	Yes
description	A description that is posted as a comment for archiving.	string	form	Yes

**Example response****Successful Response:**

HTTP/1.1 200 OK

```
{
 "archivedReviews": [
 {
 "id": 836,
 "author": "swarm",
 "changes": [789],
 "commits": [],
 "commitStatus": [],
 "created": 1461164339,
 "deployDetails": [],
```

```

 "deployStatus": null,
 "description": "Review description\n",
 "groups": [],
 "participants": {
 "swarm": []
 },
 "pending": false,
 "projects": [],
 "state": "archived",
 "stateLabel": "Archived",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1478191607
 }
],
"failedReviews": []
}

```

### Example usage

#### Archiving reviews inactive since 2016/06/30

To archive reviews not updated since 2016/06/30 inclusive:

```

curl -u "username:password" \
 -X POST \
 -d "notUpdatedSince=2016-06-30" \
 "https://my-swarm-host/api/v9/reviews/archive/"

```

P4 Code Review responds with the list of archived reviews and failed reviews if there are any:

HTTP/1.1 200 OK

```

{
 "archivedReviews": [
 {
 "id": 911,
 "author": "swarm",
 "changes": [601],
 "commits": [],
 "commitStatus": [],
 "created": 1461164344,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Touch up references on html pages.\n",
 "groups": [],
 "participants": {

```

```
 "swarm": [],
 },
 "pending": false,
 "projects": [],
 "state": "archived",
 "stateLabel": "Archived",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1478191605
},
{
 "id": 908,
 "author": "earl",
 "changes": [605],
 "commits": [],
 "commitStatus": [],
 "created": 1461947794,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Remove (attempted) installation of now deleted man pages.\n",
 "groups": [],
 "participants": {
 "swarm": []
 },
 "pending": false,
 "projects": [],
 "state": "archived",
 "stateLabel": "Archived",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1478191605
}
],
"failedReviews": [
 {
 }
]
}
```

If no reviews are archived, P4 Code Review responds with an empty reviews list:

HTTP/1.1 200 OK

```
{
 "archivedReviews": [],
 "failedReviews": []
}
```



## Add Change to Review

### Summary

Add Change to Review

POST /api/v9/reviews/{id}/changes/

### Description

Links the given change to the review and schedules an update.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
id	Review ID	integer	path	Yes	
change	Change ID	integer	form	Yes	
mode	The mode of operation, currently 'replace' or 'append'	string	form	No	replace

### Example response

Successful Response contains Review Entity:

HTTP/1.1 200 OK

```
{
```

```
"review": {
 "id": 12206,
 "author": "bruno",
 "changes": [10667, 12000],
 "commits": [10667, 12000],
 "commitStatus": [],
 "created": 1399325913,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Adding .jar that should have been included in r10145\n",
 "participants": {
 "bruno": []
 },
 "pending": false,
 "projects": [],
 "state": "archived",
 "stateLabel": "Archived",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1399325913
}
```

### Example usage

#### Adding a change to a review

You may want to update a review from a shelved or committed change that is different from the initiating change. This is done by adding a change to the review.

To replace a changelist:

```
curl -u "username:password" \
 -X POST \
 -d "change=124" \
 "https://my-swarm-host/api/v9/reviews/123/changes/"
```

To append a changelist:

```
curl -u "username:password" -X POST -d "change=124" -d "mode=append" https://my-swarm-host/api/v9/reviews/123/changes/
```

P4 Code Review responds with the updated review entity:

HTTP/1.1 200 OK

```
{
```

```

"review": {
 "id": 123,
 "author": "bruno",
 "changes": [122, 124],
 "commits": [],
 "commitStatus": [],
 "created": 1399325913,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Adding .jar that should have been included in r110\n",
 "groups": [],
 "participants": {
 "bruno": []
 },
 "pending": true,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1399325913,
 "versions": [
 {
 "difference": 1,
 "stream": null,
 "change": 124,
 "user": "bruno",
 "time": 1399330003,
 "pending": true,
 "archiveChange": 124
 }
]
}

```

## Clean up a review

### Summary

Clean up a review (v6+)

POST /api/v9/reviews/{id}/cleanup

### Description

Clean up a review for the given id.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
reopen	Expected to be a boolean (defaulting to false). If true then an attempt will be made to reopen files into a default changelist	boolean	form	No

**Example response****Successful Response:**

HTTP/1.1 200 OK

```
{
 "complete": [
 {
 "1": ["2"]
 }
],
 "incomplete": []
}
```

**Example usage****Cleaning up a review with id 1**

Cleanup review number 1, reopening any files into the default changelist.

```
curl -u "username:password" \
 -X POST \
 -d "reopen=true" \
 "https://my-swarm-host/api/v9/reviews/1/cleanup"
```

P4 Code Review responds with the review and the changelists cleaned. Depending on the completion they will be either detailed in 'complete' or 'incomplete'. Incomplete changelists will have messages indicating why it was not possible to complete:

HTTP/1.1 200 OK

```
{
 "complete": [
 {
 "1": ["2"]
 }
],
 "incomplete": []
}
```

## Obliterate a review

### Summary

Obliterate a review

POST /api/v9/reviews/{id}/obliterate

### Description

Obliterate a review for the given id.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "message": "The review with id [1] has been obliterated.",
 "code": 200
}
```

### Example usage

#### Obliterating a review with id 1

Obliterating review number 1.

```
curl -u "username:password" \
 -X POST \
 "https://my-swarm-host/api/v9/reviews/1/obliterate"
```

P4 Code Review responds with a message informing you that is has successfully Obliterated the review:

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "message": "The review with id [1] has been obliterated.",
 "code": 200
}
```

## Set vote for the authenticated user to up, down, or cleared

### Summary

Set the vote for the authenticated user to be up, down or cleared

POST /api/v9/reviews/{id}/vote/

### Description

Set the vote for the authenticated user to be up, down or clear.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
version	Expected to be a valid review version to vote on if supplied, ignored if the version does not exist and the vote will apply to the latest version	string	form	No
vote	Expected to be a valid vote. Valid votes are 'up', 'down' and 'clear	string	form	Yes

**Example response****Successful Response:**

HTTP/1.1 200 OK

```
{
 "isValid": "true",
 "messages": ["User username set vote to up on review 1"]
}
```

## Example usage

Voting up for review with id 1

```
curl -u "username:password" \
 -X POST \
 -d "vote[value]=up" -d "vote[version]=1" \
 "https://my-swarm-host/api/v9/reviews/1/vote/"
```

P4 Code Review responds with

HTTP/1.1 200 OK

```
{
 "isValid": "true",
 "messages": ["User username set vote to up on review 1"]
}
```

## Transition the Review State

### Summary

Transition the Review State (v2+)

PATCH /api/v9/reviews/{id}/state/

### Description

Transition the review to a new state. When transitioning to approved, you can optionally commit the review. (v2+).

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.



## Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes
state	Review State. Valid options: needsReview, needsRevision, approved, archived, rejected	string	form	Yes
description	An optional description that is posted as a comment for non-commit transitions. Commits that do not include a description default to using the Review description in the resulting change description.	string	form	No
commit	Set this flag to true and provide a state of <b>approved</b> in order to trigger the <b>Approve and Commit</b> action in P4 Code Review.	boolean	form	No

Parameter	Description	Type	Parameter Type	Required
<code>wait</code>	Instruct P4 Code Review to wait for a commit to finish before returning.	boolean	form	No
<code>jobs[]</code>	When performing an 'Approve and Commit', one or more jobs can be attached to the review as part of the commit process.	stringArray	form	No
<code>fixStatus</code>	Provide a fix status for the attached job(s) when performing an 'Approve and Commit'. Possible status values vary by job specification, but often include: open, suspended, closed, review, fixed.	string	form	No

### Example responses

Successful Response contains Review Entity:

HTTP/1.1 200 OK

```
{
 "review": {
 "id": 12207,
 "author": "bruno",
```

```

 "changes": [10667, 12000],
 "commits": [],
 "commitStatus": [],
 "created": 1399325913,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Adding .jar that should have been included in r10145\n",
 "participants": {
 "bruno": []
 },
 "pending": false,
 "projects": [],
 "state": "needsRevision",
 "stateLabel": "Needs Revision",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1399325913
 },
 "transitions": {
 "needsReview": "Needs Review",
 "approved": "Approve",
 "rejected": "Reject",
 "archived": "Archive"
 }
}

```

Successful Commit contains Review and Commit Entities:

HTTP/1.1 200 OK

```

{
 "review": {
 "id": 12208,
 "author": "bruno",
 "changes": [10667, 12000, 12006],
 "commits": [12006],
 "commitStatus": {
 "start": 1399326910,
 "change": 12006,
 "status": "Committed",
 "committer": "bruno",
 "end": 1399326911
 },
 "created": 1399325900,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Adding .jar that should have been included in r10145\n",

```

```
"participants": {
 "bruno": []
},
"pending": false,
"projects": [],
"state": "needsRevision",
"stateLabel": "Needs Revision",
"testDetails": [],
"testStatus": null,
"type": "default",
"updated": 1399325905
},
"transitions": {
 "needsReview": "Needs Review",
 "needsRevision": "Needs Revision",
 "rejected": "Reject",
 "archived": "Archive"
},
"commit": 12006
}
```

### Example usage

#### Committing a review

To commit a review:

```
curl -u "username:password" \
-X PATCH \
-d "state=approved" -d "commit=1" \
"https://my-swarm-host/api/v9/reviews/123/state/"
```

P4 Code Review responds with the updated review entity, as well as a list of possible transitions for the review:

HTTP/1.1 200 OK

```
{
 "review": {
 "id": 123,
 "author": "bruno",
 "changes": [122, 124],
 "commits": [124],
 "commitStatus": {
 "start": 1399326910,
 "change": 124,
 "status": "Committed",
 "committer": "bruno",

```

```

 "end": 1399326911
 },
 "created": 1399325913,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Adding .jar that should have been included in r110\n",
 "groups": [],
 "participants": {
 "bruno": []
 },
 "pending": false,
 "projects": [],
 "state": "approved",
 "stateLabel": "Approved",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1399325913,
 "versions": []
},
"transitions": {
 "needsReview": "Needs Review",
 "approved": "Approve",
 "rejected": "Reject",
 "archived": "Archive"
}
}

```

## Update Review Description

### Summary

Update Review Description

PATCH /api/v9/reviews/{review\_id}

### Description

Update the description field of a review.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).

- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

#### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>review_id</code>	Review ID	integer	path	Yes
<code>author</code>	The new author for the specified review. (At least one of Author or Description are required.)	string	form	No
<code>description</code>	The new description for the specified review. (At least one of Description or Author are required.)	string	form	No
<code>_method</code>	Method Override. If your client cannot submit HTTP PATCH, use an HTTP POST with the parameter <code>?_method=PATCH</code> to override.	string	query	No

#### Example responses

##### Successful Response:

HTTP/1.1 200 OK

```

{
 "review": {
 "id": 12306,
 "author": "swarm",
 "changes": [12205],
 "comments": 0,
 "commits": [],
 "commitStatus": [],
 "created": 1402507043,
 "deployDetails": [],
 "deployStatus": null,
 "description": "Updated Review Description\n",
 "participants": {
 "swarm": []
 },
 "pending": true,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testDetails": [],
 "testStatus": null,
 "type": "default",
 "updated": 1402518492
 },
 "transitions": {
 "needsRevision": "Needs Revision",
 "approved": "Approve",
 "rejected": "Reject",
 "archived": "Archive"
 },
 "canEditAuthor": true
}

```

**Note:**

P4 Code Review returns null for `totalCount` if no search filters were provided. `lastSeen` can often be used as an offset for pagination, by using the value in the `after` parameter of subsequent requests.

When no results are found, the **reviews** array is empty:

HTTP/1.1 200 OK

```

{
 "lastSeen": null,
 "reviews": [],
 "totalCount": 0
}

```

## Servers : P4 Code Review Servers API

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Get a list of servers

### Summary

Gets a list of servers

GET /api/v9/servers/

### Description

Gets a list of servers

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "servers": {
 "Main": {
 "port": "ssl:10.33.44.55:1666"
 },
 "Artifacts": {
 "port": "10.55.66.77:1666"
 }
 }
}
```

### Example usage

#### Get a list of servers

Get a list of servers that P4 Code Review is aware of

```
curl -u "username:password" "https://myswarm.url/api/v9/servers/"
```

If multiple p4d are configured it will give an output like

HTTP/1.1 200 OK



```
{
 "servers": {
 "Main": {
 "port": "ssl:10.33.44.55:1666"
 },
 "Artifacts": {
 "port": "10.55.66.77:1666"
 }
 }
}
```

If a single p4d is configured it will give an output like

HTTP/1.1 200 OK

```
{
 "servers": {
 "p4": {
 "port": "ssl:10.33.44.55:1666"
 }
 }
}
```

## Users : P4 Code Review Users

### Important:

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

## Get List of Users

### Summary

Get List of Users

GET /api/v9/users/

### Description

Returns a list of users in P4 Code Review.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show for each user. Omitting this parameter or passing an empty value shows all fields. Be aware, the fields are case sensitive for users. You can use one of the following: User, Type, Email, Updated, Access, FullName, JobView, Password, AuthMethod, Reviews.	string	query	No
users	An optional comma-separated list (or array) of users to display. Omitting this parameter or passing an empty value shows all users.	string	query	No

Parameter	Description	Type	Parameter Type	Required
group	An optional to get users from a group. Cannot be used with users parameter	string	query	No
ignoreExcludeList	Determines if the list of users has the <a href="#">user_exclude_list</a> filter applied or not. Add the parameter to ignore the <a href="#">user_exclude_list</a> filter.	boolean	query	No

### Example response

#### Successful Response:

HTTP/1.1 200 OK

```
{
 "User": "bruno",
 "Type": "standard",
 "Email": "bruno@perforce.com",
 "Updated": "2005/10/11 21:05:15",
 "Access": "2018/04/12 14:55:10",
 "FullName": "Bruno First",
 "JobView": null,
 "Password": null,
 "AuthMethod": "perforce",
 "Reviews": []
}
```

### Example usage

#### Listing users

To list all users:

```
curl -u "username:password" "https://my-swarm-
host/api/v9/users?fields=User,FullName&users=super,bruno"
```

Pagination is not currently supported by this endpoint. P4 Code Review responds with a list of all users:

HTTP/1.1 200 OK

```
{
 {
 "User": "bruno",
 "FullName": "Bruno First"
 },
 {
 "User": "super",
 "FullName": "Super Second"
 }
}
```

## Unfollow all Users and Projects

### Summary

Unfollow all Users and Projects

POST /api/v9/users/{user}/unfollowall

### Description

Admin and super users are permitted to execute unfollow all against any target user. Other users are only permitted to execute the call if they themselves are the target user

### Example response

Successful Response:

HTTP/1.1 200 OK

```
{
 "isValid": true,
 "messages": "User {user} is no longer following any Projects or Users."
}
```

## Workflows : Controller for querying, creating and updating workflows

**Important:**

From P4 Code Review 2022.2, P4 Code Review no longer supports APIs older than v9.

### Get workflows

**Summary**

Gets workflows

GET /api/v9/workflows/

**Description**

Get all workflows

**Parameters**

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show for each workflow. Omitting this parameter or passing an empty value shows all fields.	string	query	No

Parameter	Description	Type	Parameter Type	Required
noCache	If provided and has a value of 'true' a query will always be performed and the cache of workflows is ignored. Otherwise the cache will be used if it exists.	boolean	query	No

### Usage example

#### Get a list of workflows

```
curl -u "username:password" "https://my-swarm-host/api/v9/workflows"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflows": [
 {
 "on_submit":{
 "without_review":{
 "rule": "no_checking"
 },
 "with_review":{
 "rule": "no_checking"
 }
 },
 "name": "myWorkflow",
 "description": "A description",
 "shared": "true",
 "owners": [
 "user1",
 "user2"
],
 "end_rules":{
 "update":{
```

```
 "rule": "no_revision"
 }
 },
 "auto_approve": {
 "rule": "never"
 },
 "counted_votes": {
 "rule": "anyone"
 },
 "group_exclusions": {
 "rule": []
 },
 "user_exclusions": {
 "rule": []
 },
 "id": "1"
 },
 {
 "on_submit": {
 "without_review": {
 "rule": "no_checking"
 },
 "with_review": {
 "rule": "no_checking"
 }
 },
 "name": "myWorkflow 2",
 "description": "Another description",
 "shared": "true",
 "owners": [
 "user3",
 "user4"
],
 "end_rules": {
 "update": {
 "rule": "no_revision"
 }
 },
 "auto_approve": {
 "rule": "votes"
 },
 "counted_votes": {
 "rule": "members"
 },
 "group_exclusions": {
 "rule": []
 },
 "user_exclusions": {
```

```
 "rule": []
 },
 "id": "2"
},
]
```

## Get a workflow by id

### Summary

Gets a workflow by id

GET /api/v9/workflows/{id}

### Description

Gets a workflow by id

### Parameters

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show for each workflow. Omitting this parameter or passing an empty value shows all fields.	string	query	No

---

### Example usage

#### Get a workflows by id

```
curl -u "username:password" "https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:



HTTP/1.1 200 OK

```
{
 "workflows": [
 "on_submit":{
 "without_review":{
 "rule": "no_checking"
 },
 "with_review":{
 "rule": "no_checking"
 }
 },
 "name": "myWorkflow",
 "description": "A description",
 "shared": "true",
 "owners": [
 "user1",
 "user2"
],
 "end_rules":{
 "update":{
 "rule":"no_revision"
 }
 },
 "auto_approve":{
 "rule": "never"
 },
 "counted_votes":{
 "rule": "anyone"
 },
 "group_exclusions":[],
 "rule": []
 },
 "user_exclusions":[],
 "rule": []
},
{id": "1"
}
}
```

## Create a workflow

### Summary

Create a workflow

POST /api/v9/workflows/

**Description**

Create a new workflow.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
name	The workflow name. Will be compared against other workflows and rejected if not unique	string	form	Yes
description	Description for the new workflow	string	form	No
shared	Whether this workflow is shared for other users that do not own it. Defaults to not shared	boolean	form	No
owners	A list owners for the workflow. Can be users or group names (prefixed with swarm-group-). Users and group names must exist or the workflow will be rejected	array (of strings)	form	No

Parameter	Description	Type	Parameter Type	Required
on_submit	Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without review are no_checking, auto_create, reject	array	form	No
end_rules	Data for rules when changes are submitted. Valid values are no_checking, no_revision.	array	form	No
auto_approve	Data for rules when changes are submitted. Valid values are votes, never.	array	form	No
counted_votes	Data for rules when counting votes up. Valid values are anyone, members.	array	form	No

### Example usage

#### Create a workflow

```
curl -u "username:password" \
 -X POST \
 -d "on_submit[with_review][rule]=no_checking" \
```

```
-d "on_submit[without_review][rule]=no_checking" \
-d "name=myWorkflow" \
-d "description=A description" \
-d "shared=true" \
-d "owners[]=Francois_Piccard" \
-d "owners[]=Anna_Schmidt" \
-d "end_rules[update][rule]=no_checking" \
-d "auto_approve:[rule]never" \
-d "counted_votes:[rule]members" \
"https://my-swarm-host/api/v9/workflows"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"myWorkflow",
 "description":"A description",
 "shared":true,
 "owners":[
 "Anna_Schmidt",
 "Francois_Piccard"
],
 "end_rules":{
 "update":{
 "rule":"no_checking"
 }
 },
 "auto_approve":{
 "rule":"never"
 },
 "counted_votes":{
 "rule":"members"
 },
 "group_exclusions":[],
 "rule": []
 },
 "user_exclusions":[],
 "rule": []
},
```

```
{
 "id": 1
}
```

## Patch a workflow

### Summary

Patch a workflow

PATCH /api/v9/workflows/{id}

### Description

Patch a workflow.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	The id of the workflow being patched	string	form	Yes
name	The workflow name. Will be compared against other workflows and rejected if not unique	string	form	No
description	Description for the new workflow	string	form	No
shared	Whether this workflow is shared for other users that do not own it. Defaults to not shared	boolean	form	No

Parameter	Description	Type	Parameter Type	Required
owners	A list owners for the workflow. Can be users or group names (prefixed with swarm-group-). Users and group names must exist or the workflow will be rejected	array (of strings)	form	No
on_submit	Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without review are no_checking, auto_create, reject	array	form	No
end_rules	Data for rules when changes are submitted. Valid values are no_checking, no_revision.	array	form	No
auto_approve	Data for rules when changes are submitted. Valid values are votes, never.	array	form	No

Parameter	Description	Type	Parameter Type	Required
counted_votes	Data for rules when counting votes up. Valid values are anyone, members.	array	form	No

### Example usage

#### Patch workflow name and description

```
curl -u "username:password" \
 -X PATCH \
 -d "name=myWorkflow" \
 -d "description=A description" \
 "https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"myWorkflow",
 "description":"A description",
 "shared":false,
 "owners":[
 "bruno"
],
 "end_rules":{
 "update":{
 "rule":"no_revision"
 }
 }
 },
}
```

```
"auto_approve":{
 "rule":"votes"
},
"counted_votes":{
 "rule":"members"
},
"group_exclusions":[],
"rule":[]
},
"user_exclusions":[],
"rule":[]
},
"id":1
}
}
```

#### Patch on\_submit with\_review

```
curl -u "username:password" \
-X PATCH \
-d "on_submit[with_review][rule]=no_checking" \
"https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"test",
 "description":"test line 2",
 "shared":false,
 "owners":[
 "bruno"
],
 "end_rules":{
 "update":{
 "rule":"no_revision"
 }
 }
 },
}
```



```
"auto_approve":{
 "rule":"votes"
},
"counted_votes":{
 "rule":"members"
},
"group_exclusions":[],
"rule":[]
},
"user_exclusions":[],
"rule":[]
},
"id":1
}
}
```

**Patch on\_submit without\_review**

```
curl -u "username:password" \
-X PATCH \
-d "on_submit[without_review][rule]=no_checking" \
"https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"test",
 "description":"test line 2",
 "shared":false,
 "owners":[
 "bruno"
],
 "end_rules":{
 "update":{
 "rule":"no_revision"
 }
 }
 },
}
```

```

 "auto_approve":{
 "rule":"votes"
 },
 "counted_votes":{
 "rule":"members"
 },
 "group_exclusions":[],
 "rule":[]
 },
 "user_exclusions":[],
 "rule":[]
},
"id":1
}
}

```

#### Patch Auto\_approve

```

curl -u "username:password" \
-X PATCH \
-d "auto_approve[rule]=never" \
"https://my-swarm-host/api/v9/workflows/1"

```

JSON Response:

HTTP/1.1 200 OK

```

{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"test",
 "description":"test line 2",
 "shared":false,
 "owners":[
 "bruno"
],
 "end_rules":{
 "update":{
 "rule":"no_revision"
 }
 }
 },

```

```
"auto_approve":{
 "rule":"never"
},
"counted_votes":{
 "rule":"members"
},
"group_exclusions":[],
"rule":[]
},
"user_exclusions":[],
"rule":[]
},
"id":1
}
}
```

#### Patch counted\_votes

```
curl -u "username:password" \
-X PATCH \
-d "counted_votes[rule]=anyone" \
"https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"test",
 "description":"test line 2",
 "shared":false,
 "owners":[
 "bruno"
],
 "end_rules":{
 "update":{
 "rule":"no_revision"
 }
 }
 },
}
```

```

 "auto_approve":{
 "rule":"never"
 },
 "counted_votes":{
 "rule":"anyone"
 },
 "group_exclusions":[],
 "rule":[]
 },
 "user_exclusions":[],
 "rule":[]
},
"id":1
}
}

```

#### Patch end\_rules

```

curl -u "username:password" \
-X PATCH \
-d "end_rules[update][rule]=no_checking" \
"https://my-swarm-host/api/v9/workflows/1"

```

JSON Response:

HTTP/1.1 200 OK

```

{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"test",
 "description":"test line 2",
 "shared":false,
 "owners":[
 "bruno"
],
 "end_rules":{
 "update":{
 "rule":"no_checking"
 }
 }
 },

```

```
"auto_approve":{
 "rule":"never"
},
"counted_votes":{
 "rule":"members"
},
"group_exclusions":[],
"rule":[]
},
"user_exclusions":[],
"rule":[]
},
"id":1
}
}
```

**Patch shared**

```
curl -u "username:password" \
-X PATCH \
-d "shared=true" \
"https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"test",
 "description":"test line 2",
 "shared":true,
 "owners":[
 "bruno"
],
 "end_rules":{
 "update":{
 "rule":"no_checking"
 }
 }
 },
}
```

```
"auto_approve":{
 "rule":"never"
},
"counted_votes":{
 "rule":"anyone"
},
"group_exclusions":[],
"rule":[]
},
"user_exclusions":[],
"rule":[]
},
"id":1
}
}
```

#### Patch Owners

```
curl -u "username:password" \
-X PATCH \
-d "owners[]=Francois_Piccard" \
-d "owners[]=Anna_Schmidt" \
-d "owners[]=bruno" \
"https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"test",
 "description":"test line 2",
 "shared":true,
 "owners":[
 "Anna_Schmidt",
 "Francois_Piccard",
 "bruno"
],
 "end_rules":{
```

```
 "update":{
 "rule":"no_checking"
 },
 "auto_approve":{
 "rule":"never"
 },
 "counted_votes":{
 "rule":"anyone"
 },
 "group_exclusions":[],
 "rule": []
 },
 "user_exclusions":[],
 "rule": []
},
"id":1
}
```

## Update a workflow

### Summary

Update a workflow

PUT /api/v9/workflows/{id}

### Description

Update a workflow. All values should be provided in the request. If not provided any missing values are reverted to default.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	The id of the workflow being patched	string	form	Yes

---

Parameter	Description	Type	Parameter Type	Required
name	The workflow name. Will be compared against other workflows and rejected if not unique	string	form	Yes
owners	A list owners for the workflow. Can be users or group names (prefixed with swarm-group-). Users and group names must exist or the workflow will be rejected	array (of strings)	form	No
description	Description for the new workflow	string	form	No
shared	Whether this workflow is shared for other users that do not own it. Defaults to not shared	boolean	form	No



Parameter	Description	Type	Parameter Type	Required
on_submit	Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without review are no_checking, auto_create, reject	array	form	No
end_rules	Data for rules when changes are submitted. Valid values are no_checking, no_revision.	array	form	No
auto_approve	Data for rules when changes are submitted. Valid values are votes, never.	array	form	No
counted_votes	Data for rules when counting votes up. Valid values are anyone, members.	array	form	No

### Example usage

#### Update a workflow

```
curl -u "username:password" \
 -X PUT \
 -d "on_submit[with_review][rule]=no_checking" \
```

```
-d "on_submit[without_review][rule]=no_checking" \
-d "name=myWorkflow" \
-d "description=A description" \
-d "shared=true" \
-d "owners[]=Francois_Piccard" \
-d "owners[]=Anna_Schmidt" \
"https://my-swarm-host/api/v9/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "workflow":{
 "on_submit":{
 "with_review":{
 "rule":"no_checking"
 },
 "without_review":{
 "rule":"no_checking"
 }
 },
 "name":"myWorkflow",
 "description":"A description",
 "shared":true,
 "owners":[
 "Anna_Schmidt",
 "Francois_Piccard"
],
 "end_rules":{
 "update":{
 "rule":"no_checking"
 }
 },
 "auto_approve":{
 "rule":"never"
 },
 "counted_votes":{
 "rule":"anyone"
 },
 "group_exclusions":[],
 "rule": []
 },
 "user_exclusions":[],
 "rule": []
},
 "id":1
}
```

## Delete a workflow

### Summary

Delete a workflow

DELETE /api/v9/workflows/{id}

### Description

Delete a workflow for the provided id. This call must be authenticated and the user must have permission to edit the workflow. If the workflow is in use it cannot be deleted and an error message will be returned.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	The id of the workflow being deleted	string	form	Yes

### Example usage

#### Delete a workflow not in use

```
curl -u "username:password" -X DELETE "https://my-swarm-host/api/v9/workflows/1"
```

#### JSON Response

HTTP/1.1 200 OK

```
{
 "isValid": true,
}
```

```
"messages": [
 "Workflow [1] was deleted"
]
}
```

**Delete a workflow that is in use on public projects**

```
curl -u "username:password" -X DELETE "https://my-swarm-host/api/v9/workflows/1"
```

**JSON Response**

HTTP/1.1 200 OK

```
{
 "isValid": false,
 "messages": [
 "Cannot delete workflow [1], it is in use on project [project1, project2]"
]
}
```

**Delete a workflow that is in use on a public and some private projects**

```
curl -u "username:password" -X DELETE "https://my-swarm-host/api/v9/workflows/1"
```

**JSON Response**

HTTP/1.1 200 OK

```
{
 "isValid": false,
 "messages": [
 "Cannot delete workflow [1], it is in use on project [project1] and others"
]
}
```

**Delete a workflow that is in use on only private projects**

```
curl -u "username:password" -X DELETE "https://my-swarm-host/api/v9/workflows/1"
```

**JSON Response**

HTTP/1.1 200 OK

```
{
 "isValid": false,
```

```

 "messages": [
 "Cannot delete workflow [1], it is in use"
]
}

```

## API (v9) examples

This section contains some API (v9) examples.

### Extended API example

This section contains an extended API example, involving multiple API calls to answer a more complicated kind of question than any single API endpoint can provide: which reviews does a specific *userid* need to attend to?

#### The code

```

<?php
/**
 * vim:set ai si et ts=4 sw=4 syntax=php:
 *
 * reviews.php
 *
 * Queries the Swarm API and reports which reviews a specified user
 * needs to attend to.
 *
 * Required attention is determined by the following criteria:
 * - the user is a participant in a review
 * - and the user has not voted on the review
 * - and the user has not commented on the review
 * - or the user's comment on the review is a
 * task that has been addressed and needs verification
 */

if (ini_set('track_errors', 1) === false) {
 echo "Warning: unable to track errors.\n";
}

process command-line arguments
$options = getopt(
 'hs:r:v',
 array('help', 'swarm:', 'reviewer', 'verbose')
);

$swarm = "";
if (isset($options['s'])) {
 $swarm = $options['s'];
}

```

```
}
if (isset($options['swarm'])) {
 $swarm = $options['swarm'];
}
if (!$swarm) {
 usage('Swarm API URL not provided.');
```

```
$reviewer = "";
if (isset($options['r'])) {
 $reviewer = $options['r'];
}
if (isset($options['reviewer'])) {
 $reviewer = $options['reviewer'];
}
if (!$reviewer) {
 usage('Swarm reviewer not provided.');
```

```
$verbose = false;
if (isset($options['v']) || isset($options['verbose'])) {
 $verbose = true;
}
```

```
if (isset($options['h']) || isset($options['help'])) {
 usage();
}
```

```
function usage($message = null)
{
 if ($message) {
 echo "$message\n\n";
 }
}
```

```
$script = basename(__FILE__);
echo <<<EOU
$script: -s <Swarm URL> -u <API userid> -p <API user's password> \
-r <reviewer userid to report on> -h

-s|--swarm Swarm's URL (e.g. https://user@password:myswarm.url/)
-r|--reviewer The reviewer to report on.
-h|--help This help text.
-v|--verbose Verbose output.
```

This script queries the Swarm API and reports on reviews that the specified user needs to attend to.

Note: If your Helix Core Server (p4d) has security level 3 set, you

cannot use a password to authenticate; you must acquire a host-unlocked ticket from p4d, and use the ticket in place of a password when communicating with the Swarm API connected to p4d.

```
EOU;
 exit;
}

function msg($message)
{
 global $verbose;

 if ($verbose) {
 echo $message;
 }
}

function call_api($url, $params)
{
 global $php_errormsg;

 $query = http_build_query($params);
 $request = $url . '?' . $query;
 $response = @file_get_contents($request);
 if ($php_errormsg) {
 echo "Unable to call api: $php_errormsg\n";
 exit;
 }

 $json = @json_decode($response, true);
 if ($php_errormsg) {
 echo "Unable to decode api response: $php_errormsg\n";
 exit;
 }

 return $json;
}

remove trailing / from URL, if it exists
$swarm = rtrim(trim($swarm), '/');

fetch the list of reviews
$reviews = call_api(
 "$swarm/api/v9/reviews",
 array(
 'hasReviewers' => 1, # only reviews with participants
 'participants' => array($reviewer), # only review for this reviewer
 'max' => 9, # get plenty of reviews, if available
)
);
```

```
 'fields' => array('id', 'description', 'commits'), # get these fields
)
);

$report = array();
foreach ($reviews['reviews'] as $review) {
 if (is_null($review)) {
 continue;
 }

 $flag = false;
 msg('Review: ' . $review['id'] . ' ');

 # if the review is already committed, it likely does not need attention
 if (array_key_exists('commits', $review)
 && count($review['commits'])
) {
 msg("is committed, skipping...\n");
 continue;
 }

 # if the review has a vote from the reviewer, they are already aware
 if (array_key_exists('participants', $review)
 && array_key_exists('vote', $review['participants'][$reviewer])
) {
 msg("has vote from reviewer, skipping...\n");
 continue;
 }

 # if there are no open comments on the review, the reviewer's
 # attention is required
 if (array_key_exists('comments', $review)
 && $review['comments'][0] == 0
) {
 msg("has no open comments, skipping...\n");
 continue;
 }

 # fetch the comments for this review
 $comments = call_api(
 "$swarm/api/v9/comments",
 array(
 'topic' => 'reviews/' . $review['id'], # comments for this review
 'max' => 9, # get plenty of comments, if available
)
);

 foreach ($comments['comments'] as $comment) {
```



```

msg("\n Comment: " . $comment['id'] . ' ');

// skip over comments from other reviewers
if (array_key_exists('user', $comment) && $reviewer != $comment['user']) {
 msg("is by another user, carry on...\n");
 continue;
}

skip archived comments
if (array_key_exists('flags', $comment)
 && count($comment['flags']) > 0
 && $comment['flags'][0] == 'closed'
){
 msg("is archived, carry on...\n");
 continue;
}

skip marked tasks
if (array_key_exists('taskState', $comment)
 && ($comment['taskState'] == 'comment'
 || $comment['taskState'] == 'verified'
 || $comment['taskState'] == 'open'
)
){
 msg("reviewer's comment needs attention, carry on...\n");
 continue;
}

// anything else means that the reviewer's comment needs attention
// by the reviewer
$flag = true;
msg("needs attention!\n");
break;
}

// evaluation is complete. Does this review need attention?
if ($flag) {
 $report[] = $review;
}
}

if (count($report)) {
 echo "User '$reviewer' needs to attend to these reviews:\n";
 foreach ($report as $review) {
 $description = trim($review['description']);
 if (strlen($description) > 60) {
 $description = substr($description, 0, 60) . '...';
 }
 }
}

```

```
 echo $review["id"] . ": $description\n";
 }
} else {
 echo "User '$reviewer' has no reviews to attend to.\n";
}
```

#### Executing the example

The example is written in PHP. To use it copy and paste it into a file called `reviews.php`. Then, execute it like this:

```
$ php reviews.php -s https://myswarm.host:port/ -r bob
```

Replace `https://myswarm.host/` with the URL to your P4 Code Review installation. Replace `bob` with the userid you'd like to report on.

To authenticate, insert `username:password@` before the hostname. If your P4 Server security counter is set to 3 or higher, you need to acquire a ticket and use the ticket in place of the password (see [Authentication](#) for details). If your Swarm is installed on a [custom port](#), or is installed in a [sub-folder](#), include those elements in the URL as well. For example:

```
$ php reviews.php -s
https://me:F0FC33068BA244B1BBD8196CC9166F34@my.host:8080/swarm/ -r bob
```

If you do not specify the URL correctly, you might see an error like:

```
Unable to call api: file_get_contents(http://...@my.host:8080/swarm/api/v9
/reviews?hasReviewers=1&participants%5B0%5D=bob&max=9&fields%5B0%5D=id&fields%5B1%
5D=description&fields%5B2%5D=commits): failed to open stream: HTTP request failed! HTTP/1.1
404 Not Found
```

If there are no errors, and the specified userid does have reviews to attend to, the output might look like:

```
1234: Added grapple-grommit support to woozlewoobble class. @bob sh ...
```

1234 is the id of a review that `bob` should attend to, followed by the first 60 characters of the review's description.

## Pending review cleanup API example

This section contains an example script that cleans up pending changelists which are no longer needed. See the [review cleanup](#) options for how this can be done automatically when a review is committed.

For pending changelists which were present before this option was available, or for reviews which have been contributed to by multiple authors and so require super user access to tidy up, there is an API which allows the super user to bulk remove such changelists.

The script demonstrates how this API could be used. It isn't meant as a complete solution, just a guide to demonstrate what is possible.

**The code**

```

<?php
/**
 * Perforce Swarm
 *
 * @copyright 2017 Perforce Software. All rights reserved.
 * @license Please see LICENSE.txt in top-level folder of this distribution.
 * @version <release>/<patch>
 */
/**
 * This example script can be used to clean up (delete) Perforce Server pending changelists
 * automatically when run
 * as a super user. It is able to query for reviews based on parameters to establish which
 * changelists are eligible
 * for clean up. In this way, it can be tailored to run against reviews of a user's choice.

 * Requirements for this script:
 * - MUST be a super user
 * - MUST populate the parameters below
 * - MUST be using Swarm 2017.1 or later
 *
 * Usage of script
 * php superUserReviewCleanUp.php max=10 notUpdatedSince=2017-04-01
 * state=approved
 * php superUserReviewCleanUp.php max=10 author=bruno state=approved
 *
 * Each of the parameters that you can use are documented at the following URL:
 * https://www.perforce.com/perforce/doc.current/manuals/swarm/Content/Swarm/swarm-
 * apidoc_endpoint_reviews.html
 *
 *
 * This returns a JSON object that contains four main objects.
 *
 * {
 * "error": "",
 * "help": "",
 * "results": [],
 * "search_criteria": {
 * "fields": "id",
 * "max": "10",
 * "notUpdatedSince": "2017-04-01"
 * }
 * }
 *
 * Referencing the example above:
 *
 * The error section will contain any errors that have been encountered trying to execute the
 * script.

```

```
*
* The help section will populated if you have run the command php
superUserReviewCleanUp.php help
*
* The search_criteria section indicates which parameters have been used to fetch the reviews
list.
*
* The results section returns a JSON object of each of the reviews it has processed, which
may
* include actions that were incomplete.
*
* Below is an example of results being processed:
* "814": {
* "complete": [],
* "incomplete": {
* "814": {
* "813": [
* "0": "Command failed: No shelved files in changelist to delete.",
*]
* }
* }
* },
* "818": {
* "complete": [],
* "incomplete": []
* }
* "820": {
* "complete": [],
* "incomplete": {
* "820": {
* "821": [
* "0": "Command failed: No shelved files in changelist to delete.",
* "1": "Command failed: Usage: fix [-d] [-s status] -c changelist# jobName
...\nMissing/wrong
* number of arguments."
*]
* }
* }
* },
*
* Some reviews may have no actions or incomplete actions. Incomplete actions indicate that
additional work is required
* and the review could not be entirely cleaned up. In the example above, the first message
indicates a changelist was
* not found. This could be because the end user has already deleted it.
*
* An error with the fix command can indicate that the pending changelist doesn't have any jobs
linked or that the jobs
```

```

* have already been removed.
*
* NOTES:
* To make the output print nicely, you can use the python command like this:
*
* php superUserReviewCleanUp.php max=10 notUpdatedSince=2018-04-01 state=approved
| python -m json.tool
*
*/

/* ***** */
/* These values MUST be set before running the script. */
/* ***** */

// URL to access swarm. Must not include trailing slash.
$swarmURL = 'http://my.swarm.com';
// Username of super user
$username = "";
// Ticket for user, can be created by running "p4 -u $username login -pa"
$ticket = "";

/* ***** */

// If the super user wants to reopen the files of the end user.
$reopen = true;
// Prebuild the the return message helper array.
$help = array("help" => "", "error" => "", "results" => "", "search_criteria" => "");
// @codingStandardsIgnoreEnd
/**
 * function that make the GET or POST requests
 *
 * @param $url Url in which we want to make our request to
 * @param bool $fetch Set to true for a GET request, otherwise will do a POST.
 * @param array $args These are the parameter that we pass the the GET or POST request to
filter the reviews
 * @return JSON We return a JSON object back to be worked on.
 */
function request($url, $fetch = false, $args = array())
{
 // Fetch the settings to allow this function to access them.
 global $username, $ticket, $reopen, $help;
 $curl = "";

 // If GET request fetch should be true and args shouldn't be empty
 if ($fetch === true) {
 // If is args is empty just give the url, otherwise build a http query with args elements

```

```
$curl = empty($args) ? curl_init($url) : curl_init($url."?".http_build_query($args));
} else {
 // Assume fetch is false and build a POST request.
 $curl = curl_init($url."/".$args['id'].'cleanup');
 curl_setopt($curl, CURLOPT_POSTFIELDS, http_build_query(array
('reopened'=>$reopen)));
 curl_setopt($curl, CURLOPT_POST, count($reopen));
}
curl_setopt($curl, CURLOPT_USERPWD, "$username:$ticket");
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
curl_setopt($curl, CURLOPT_HTTPAUTH, CURLAUTH_BASIC);

$result = curl_exec($curl);

// Catch the error code in case Ticket expired or url doesn't return.
$statuscode = curl_getinfo($curl, CURLINFO_HTTP_CODE);
if ($statusCode != 200) {
 $help['error'] = array("HTTP code" => $statusCode);
}

curl_close($curl);
return json_decode($result, true);
}

/**
 * Fetch a list of Reviews based on parameters
 *
 * @param $elements Each of the args passed from command line are treated as elements
 * @return JSON We return a JSON object back to be worked on.
 */
function fetchReviews($elements)
{
 global $swarmURL, $help;
 $parameters = array();

 if ($elements !== null) {
 // Loop though each of the args we want to fetch reviews based on
 foreach ($elements as $key => $value) {
 // Break each of the element into key and value to allow use to build a array to pass to url
 $brokenDown = explode("=", $value);
 if ($key !== 0 && sizeof($brokenDown) === 2) {
 $parameters[$brokenDown[0]] = $brokenDown[1];
 }
 }
 // We only require the field id to limit the amount of data.
 $parameters['fields'] = 'id';
 // Helpful for debugging which parameters we have passed in the queue.
 $help['search_criteria'] = $parameters;
 }
}
```

```

 }
 // Now make the request to the Swarm server with your field options.
 $result = request(
 "$swarmURL/api/v9/reviews",
 true,
 $parameters
);

 // Return the JSON object back to be worked on.
 return $result;
}

/**
 * Loop though each of the Reviews passed in and run clean up for them.
 *
 * @param $reviews JSON object of all the reviews we want to run cleanup on
 * $reviews => array(
 * 'reviews' => array(
 * 0 => array (
 * 'id' => 134,
 *),
 * 1 => array (
 * 'id' => 136,
 *),
 * 2 => array (
 * 'id' => 143,
 *),
 * 3 => array (
 * 'id' => 158,
 *)
 *)
 *)
 */
* @return $array return the array of work that has been carried out.
*/
function runCleanUp($reviews)
{
 global $swarmURL;
 $results = array();
 // Check if there is an reviews element of the array
 if (isset($reviews['reviews'])) {
 foreach ($reviews['reviews'] as $review) {
 // Now make the request to the Swarm for each review.
 $results[$review['id']] = request(
 "$swarmURL/api/v9/reviews",
 false,
 $review
);
 }
 }
}

```

```
 }
 }
 return $results;
}

/**
 * The help function in case a user doesn't set the basic settings.
 *
 * @param $help Pass the help array from main script to append the helpful message.
 * @return $array Return the array that will be presented to the users.
 */
function helpMessage($help)
{
 $help["help"] = array("1" => "", "2" => "", "3" => "");
 $help["help"]["1"] = "Please ensure you have set the Username, Ticket and Swarm URL
before using this script";
 $help["help"]["2"] = "Running the script can be done by using any of the standard Swarm API
fields for reviews";
 $help["help"]["3"] = "Visit
https://www.perforce.com/perforce/doc.current/manuals/swarm/index.html";
 return $help;
}

// check the first argument is not help.
$helpSet = isset($argv[1]) && $argv[1] == "help" ? "help" : null;

// Check if the user has given help as a command to this script.
if (isset($argv[1]) && $helpSet == "help") {
 // Set the $help array with the help message.
 $help = helpMessage($help);
}

// Check if the basic user ticket and swarmurl are set.
if (!empty($username) && !empty($ticket) && !empty($swarmURL) && $helpSet != "help") {
 try {
 $help["results"] = runCleanUp(fetchReviews($argv));
 } catch (Exception $e) {
 $help = helpMessage($help);
 }
} else {
 $message = "Please ensure you have set the below before using this script";
 $dataArray = array("message" => $message, "parameter" => "");

 $missingParameter = array();

 // Now check if the basic settings are empty and show the end user.
 empty($username) ? $missingParameter[] = "Username:";
 empty($ticket) ? $missingParameter[] = "Ticket:";
}
```



```

empty($swarmURL) ? $missingParameter[] = "SwarmURL":"";

$errorArray['parameter'] = $missingParameter;

$help["error"] = $errorArray ;
}
// Output the end result of the what the script does.
echo json_encode($help, JSON_FORCE_OBJECT);
// @codingStandardsIgnoreEnd

```

#### Executing the script

The example is written in PHP, and demonstrates how to make use of the APIs which remove unneeded pending changelists. It **must** be run as a super user.

For a full set of instructions on how to use the example script, see the comments in the script itself.

Abbreviated instructions:

1. Set the value of the `$swarmURL`, `$username` and `$ticket` variables.
2. Run the script by using a command similar to the following:

```
$ php pendingReviewCleanUp.php max=10 author=bruno state=approved
```

## API endpoints (v10)

This section includes coverage for the endpoints provided by the API (v10).

**In this section:**

## Activity: P4 Code Review activity

### Get a list of all activities

#### Summary

Get a list of all activities.

GET /api/v10/activity

#### Description

Get a list of all activities the user has permission to view.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
followed	<p>Set the followed parameter to true to limit results to activities followed by the user making the request.</p> <p>If the followed parameter is set to false or is not included in the request, results are not filtered by followed activities.</p>	Boolean	query	No

Parameter	Description	Type	Parameter Type	Required
after	An activity ID to seek to. Activity entries up to and including the specified ID are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.	integer	query	No
max	Maximum number of activity entries to return. This does not guarantee that max entries are returned. It does guarantee that the number of entries returned won't exceed max. Server-side filtering may exclude some activity entries for permissions reasons.  If the max parameter is not included in the request, only the most recent 100 activities are returned.	integer	query	No

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No

### Example usage

#### Get a list of all activities

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/activity"
```

P4 Code Review responds with:

**Tip:**

By default, this is limited to the 100 most recent activities. This can be changed by adding a max parameter value to the request.

```
{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1646,
 "type": "review",
 "link": [
 "review",
 {
 "review": "1236",
 "version": 4
 }
],
 "user": null
 },
 {
 "action": "Automated tests reported",
 "target": "review 1236",
 "preposition": "failed tests for",
 "description": "Make a review.",
 "details": [],
 }
]
 }
}
```

```

 "topic": "reviews/1236",
 "depotFile": null,
 "time": 1620228301,
 "behalfOf": null,
 "projects": [],
 "followers": [
 "bruno"
 "bob"
],
 "streams": {
 "review-1236",
 "user-",
 "personal-",
 "personal-macy.winter",
 "personal-super"
 },
 "change": 1236
 },
 {
 Other ids formatted as above
 }
],
"totalCount": 100,
"lastSeen": 1646
}
}

```

#### Activity pagination

To get the second page of activity entries limited to a max of 1 and showing specified fields only (based on the previous example):

```

curl -u "username:ticket"
"https://myswarm.url/api/v10/activity?max=1&after=1646&fields=id,action,time,description,type"

```

P4 Code Review again responds with:

```

{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1645,
 "type": "review",
 "action": "Automated tests reported",
 "description": "Make a review.",
 "time": 1626945417
 }
]
 },

```

```
"totalCount": 1,
"lastSeen": 1645
}
}
```

**Get a list of all activities the requester follows**

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/activity?followed=true"
```

P4 Code Review responds with:

```
{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1646,
 "type": "review",
 "link": [
 "review",
 {
 "review": "1236",
 "version": 4
 }
],
 "user": null,
 "action": "Automated tests reported",
 "target": "review 1236",
 "preposition": "failed tests for",
 "description": "Make a review.",
 "details": [],
 "topic": "reviews/1236",
 "depotFile": null,
 "time": 1620228301,
 "behalfOf": null,
 "projects": [],
 "followers": [
 "bruno",
 "bob"
],
 "streams": {
 "review-1236",
 "user-",
 "personal-",
 "personal-macy.winter",
 "personal-super"
 },
 "change": 1236
 }
]
 }
}
```

```

 },
 {
 Other ids formatted as above
 }
],
 "totalCount": 100,
 "lastSeen": 1646
}
}

```

**If a request fails****<error code>:**

- 400 one of the following:
  - the max parameter value must be an integer greater than 0
  - the after parameter must be an integer greater than 0
- 404 record not found
- 500 internal error, request failed

HTTP/1.1 &lt;response error code&gt;

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Get a list of all activities of a specified type

**Summary**

Get a list of all activities of a specified type.

GET /api/v10/activity/{type}

**Description**

Get a list of all activities of a specified type that the user has permission to view.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
followed	<p>Set the followed parameter to true to limit results to activities followed by the user making the request.</p> <p>If the followed parameter is set to false or is not included in the request, results are not filtered by followed activities.</p>	Boolean	query	No



Parameter	Description	Type	Parameter Type	Required
after	An activity ID to seek to. Activity entries up to and including the specified ID are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.	integer	query	No
max	Maximum number of activity entries to return. This does not guarantee that max entries are returned. It does guarantee that the number of entries returned won't exceed max. Server-side filtering may exclude some activity entries for permissions reasons.  If the max parameter is not included in the request, only the most recent 100 activities are returned.	integer	query	No

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No

### Example usage

#### Get a list of all review activities

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/activity/review"
```

P4 Code Review responds with:

**Tip:**

By default, this is limited to the 100 most recent activities. This can be changed by adding a max parameter value to the request.

```
{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1646,
 "type": "review",
 "link": [
 "review",
 {
 "review": "1236",
 "version": 4
 }
],
 "user": null
 },
 {
 "action": "Automated tests reported",
 "target": "review 1236",
 "preposition": "failed tests for",
 "description": "Make a review.",
 "details": [],
 }
]
 }
}
```

```

 "topic": "reviews/1236",
 "depotFile": null,
 "time": 1620228301,
 "behalfOf": null,
 "projects": [],
 "followers": [
 "bruno"
 "bob"
],
 "streams": {
 "review-1236",
 "user-",
 "personal-",
 "personal-macy.winter",
 "personal-super"
 },
 "change": 1236
 },
 {
 Other ids formatted as above
 }
],
"totalCount": 100,
"lastSeen": 1646
}
}

```

#### Activity pagination

To get the second page of review activity entries limited to a max of 1 and showing specified fields only (based on the previous example):

```

curl -u "username:ticket"
"https://myswarm.url/api/v10
/activity/review?max=1&after=1646&fields=id,action,date,description,type"

```

P4 Code Review responds with:

```

{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1645,
 "type": "review",
 "action": "Automated tests reported",
 "description": "Make a review.",
 "time": 1626945417
 }
]
 }
}

```

```
],
 "totalCount": 1,
 "lastSeen": 1645
 }
}
```

**Get a list of all review activities the requester follows**

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/activity/review?followed=true"
```

P4 Code Review responds with:

```
{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1646,
 "type": "review",
 "link": [
 "review",
 {
 "review": "1236",
 "version": 4
 }
],
 },
],
 "user": null,
 "action": "Automated tests reported",
 "target": "review 1236",
 "preposition": "failed tests for",
 "description": "Make a review.",
 "details": [],
 "topic": "reviews/1236",
 "depotFile": null,
 "time": 1620228301,
 "behalfOf": null,
 "projects": [],
 "followers": [
 "bruno"
 "bob"
],
 "streams": {
 "review-1236",
 "user-",
 "personal-",
 "personal-macy.winter",
 "personal-super"
 },
 },
}
```

```

 "change": 1236
 },
 {
 Other ids formatted as above
 }
],
"totalCount": 100,
"lastSeen": 1646
}
}

```

#### If a request fails

<error code>:

- 400 one of the following:
  - the max parameter value must be an integer greater than 0
  - the after parameter must be an integer greater than 0
- 404 record not found
- 500 internal error, request failed

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Get a list of all activities for a specified review

### Summary

Get a list of all activities for a specified review.

GET /api/v10/reviews/{id}/activity

### Description

Get a list of all activities for a specified review that the user has permission to view.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
followed	<p>Set the followed parameter to true to limit results to activities followed by the user making the request.</p> <p>If the followed parameter is set to false or is not included in the request, results are not filtered by followed activities.</p>	Boolean	query	No

Parameter	Description	Type	Parameter Type	Required
after	An activity ID to seek to. Activity entries up to and including the specified ID are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.	integer	query	No
max	Maximum number of activity entries to return. This does not guarantee that max entries are returned. It does guarantee that the number of entries returned won't exceed max. Server-side filtering may exclude some activity entries for permissions reasons.  If the max parameter is not included in the request, only the most recent 100 activities are returned.	integer	query	No

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.	string	query	No

### Example usage

Get a list of all activities for review 12345

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/reviews/12345/activity"
```

P4 Code Review responds with:

**Tip:**

By default, this is limited to the 100 most recent activities. This can be changed by adding a max parameter value to the request.

```
{
 "error": null,
 "messages": [],
 "data": {
 "activity": [
 {
 "id": 790,
 "type": "review",
 "link": [
 "review",
 {
 "review": "12345",
 "version": 1
 }
],
 "user": "bruno",
 "action": "changed author of",
 "target": "review 12345 (revision 1)",
 "preposition": "for",
 "description": "Has comment that is outside of the Diff chunks.",
 "details": {
```



```

 "author": {
 "oldAuthor": "dai",
 "newAuthor": "earl"
 }
 },
 "topic": "reviews/12345",
 "depotFile": null,
 "time": 1640102250,
 "behalfOf": null,
 "projects": {
 "jam": [
 "main"
]
 },
 "followers": [],
 "streams": {
 "0": "review-12345",
 "1": "user-bruno",
 "2": "personal-bruno",
 "3": "project-jam",
 "4": "group-Administrators",
 "6": "personal-dai",
 "7": "personal-earl",
 "8": "personal-ona",
 "9": "personal-raj",
 "10": "personal-swarm"
 },
 "change": 12344
},
{
 Other ids formatted as above
}
],
"totalCount": 20,
"lastSeen": 790
}
}

```

#### Activity pagination

To get the second page of review activity entries limited to a max of 1 and showing specified fields only (based on the previous example):

```

curl -u "username:ticket"
"https://myswarm.url/api/v10
/reviews/12345/activity?max=1&after=790&fields=id,action,date,description,type"

```

P4 Code Review responds with:

```
{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 789,
 "type": "review",
 "action": "Automated tests reported",
 "description": "Make a review.",
 "time": 1626945417
 }
],
 "totalCount": 1,
 "lastSeen": 789
 }
}
```

**If a request fails**

<error code>:

- 400 one of the following:
  - the max parameter value must be an integer greater than 0
  - the after parameter must be an integer greater than 0
- 404 review not found
- 500 internal error, request failed

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## AiAnalysis: P4 Code Review AI code analysis

### Use AI to analyze code

**Summary**

Use AI to generate explanations for a diff when reviewing a code change.

POST /api/v10/AiAnalysis/analyzeCode

### Description

Use AI to generate explanations for a diff when reviewing a code change.

### Parameters

Parameter	Description	Type	Parameter Type	Required
reviewId	Review ID	integer	body	Yes
fileId	<p>This is the full filepath including the filename encoded to <b>URL safe Base64</b>.</p> <p>Various tools are available online to encode the id to <b>URL safe Base64</b>, for example: <a href="#">BASE64 Decode and Encode</a></p>	string	body	Yes
toVersion	<p>Specify a value for toVersion:</p> <ul style="list-style-type: none"><li>■ If omitted, toVersion defaults to the latest version of the review.</li><li>■ Cannot be 0 or -1</li><li>■ Must not be greater than the latest version of the review.</li></ul>	integer	body	Yes

Parameter	Description	Type	Parameter Type	Required
fromVersion	<p>Specify a value for fromVersion:</p> <ul style="list-style-type: none"><li>0 specifies the base version of the review.</li><li>-1 specifies the current #head version of the review files in the P4 Server.</li><li>If omitted, fromVersion defaults to 0 which is the base version of the review.</li><li>Must be less than the latest version of the review.</li></ul>	integer	body	Yes
digest	<p>Specify the file digest.</p>	string	body	No
fileName	<p>Specify the name of the file being sent to the AI vendor to generate an explanation.</p>	string	body	Yes

Parameter	Description	Type	Parameter Type	Required
diffContent	<p>Send the diff content from the P4 Code Review UI to the AI vendor to get the explanation on the diff change.</p> <p>For example, "diffContent": "static void var_mods();",</p>	string	body	Yes
diffStart	<p>Specify the starting line number of the diff.</p>	integer	body	Yes
diffEnd	<p>Specify the ending line number of the diff.</p>	integer	body	Yes
id	<p>This is the P4 Server AI summary record id.</p> <p>If User 1 has already generated an AI summary for a diff, and User 2 attempts to create a summary for the same diff, the UI will include this id parameter from User 1's response in the request body for User 2.</p>	integer	body	Yes
aiPackage	<p>Specify the name of the AI package used for generating AI code explanations for a diff during a code review.</p>	string	body	No

## Example usage

Analyze code for a diff starting at line number 43 and ending at line number 56 using AI

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://my-swarm-host/api/v10/AiAnalysis/analyzeCode"
```

The "mybodyfilename.txt" file contains:

```
{
 "reviewId": 12169,
 "fileId": "Kz8jMNVdC9KYW0vABCDEF3zcmMvZXhwTC7qOpP",
 "toVersion": 1,
 "fromVersion": 0,
 "digest": "5CDD3T83GG14518BC660CAF516066AA0",
 "fileName": "nodeofequal.c",
 "diffContent": "static void IsCorrectBehaviour();",
 "diffStart": 43,
 "diffEnd": 56
}
```

Swarm responds with:

```
{
 "error": null,
 "messages": [],
 "data": {
 "aiSummary": {
 "12169": {
 "0_1": {
 "Kz8jMNVdC9KYW0vABCDEF3zcmMvZXhwTC7qOpP": {
 "43_56": {
 "1": {
 "ai_package_id": "1",
 "users": [
 "dai"
],
 "digest": "5CDD3T83GG14518BC660CAF516066AA0",
 "content": "It appears that there may be a misunderstanding or miscommunication
```

with

the providing of the code. Given the line of code, it could suggest the declaration of a static void function called `IsCorrectBehaviour` within a file named `nodeofequal.c`. In C programming, `static` function means that the function is only visible to other functions in the same file (more precisely, the same translation unit), `void` means this function does not return a value. However, the exact behavior or purpose of the `IsCorrectBehaviour` function can't be determined as the actual implementation of the function is not given. It is likely to hold a series of commands or checks to determine if some form of behavior is correct, due to its name. Your original question may be missing some

## If a request fails

- 400 invalid type given, string expected
- 401 unauthorized

## Summary

DELETE /api/v10/AiAnalysis/discard

Discard the AI summary of a diff for the logged in user.

## Example usage

Discard summary for review 12373 with the P4 Server AI summary record id 2

```
curl -X DELETE -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/AiAnalysis/discard"
```

The "mybodyfilename.txt" file contains:

```
{
 "reviewId": 12373,
 "id": 2
}'
```

Swarm responds with:

```
{
 "error": null,
 "messages": [],
 "data": {
 "aiSummary": {
 "12373": {
 "0_1": {
 "Ly9kZXBvdC9KYW0vUkVMMi4xL3NyYy9NYWtlZmlsZQ": {
 "25_42": {
 "1": {
 "ai_package_id": "1",
 "users": [],
 "digest": "F3477A04DF39FBA82870832E517E981E",
 "content": "This json file describes a diff, or the changes made, to a Makefile, which
 is a script used by the make build automation tool in programming. The diff in
 this file shows
 that new lines of comments and instructions were added to the Makefile. From
 line 20 to 24,
 we see previously commented out (disabled) lines regarding the compiler, the
 flags used in
 compilation, the target, and the libraries to be linked to the program.\n\nStarting
 from
 line 25, the following commented out instruction or note was added to the
 Makefile for
 multiple lines: \n\n```\n# NT (with Microsoft compiler)\n# Use FATFS if building
 on a
 DOS FAT file system\n\nFATFS might denote a library for working with FAT
 file
 systems (commonly used in Microsoft's operating systems).\n\nThe diff also
 shows
 that the rest of the Makefile's instructions remain unchanged, with the
 `install: jam0` target indicating that to install the program, the
 `jam0` target should be built first, and then the command `jam0 -f Jambase
 install` should be
```



```

 executed. The `-f` flag here typically indicates the file to be used by the `jam0`
command,
 in this case, `Jambase`. The purpose and functionality of these commands
would heavily
 depend on their context and the specifics of the project for which this Makefile is
used.",
 "error": false
 }
}
}
}
}
},
 "totalCount": 1
}
}
```

If a request fails

<error code>:

- 400 the input is not an integer greater than '0'
- 401 unauthorized
- 404 cannot fetch entry. Id does not exist.

## Delete AI summaries

### Summary

Permanently delete the AI summaries that are older than a given amount of time. By default, this is 30 days. Change the AI summary retention time via the `data_retention_lifetime` parameter found in the `config.php` file. For more information, see [Configuration overview](#).

The AI summaries are deleted using a cron job which runs at runs at 6:00 PM on the 28th of every month. This cron job runs in the P4 Code Review server's time zone.

Do I need to manually add the cron job to P4 Code Review?

Use the table below to determine if you need to manually add the cron job to the AI summaries.

P4 Code Review version	Installation method	Do I need to manually add cron job?
Upgrading to 2025.2	Package	Yes

P4 Code Review version	Installation method	Do I need to manually add cron job?
Upgrading to 2025.2	Docker	No
Upgrading to 2025.2	Tarball	Yes
Installing 2025.2 and onwards	Package	No
Installing 2025.2 and onwards	Docker	No
Installing 2025.2 and onwards	Tarball	Yes

To manually add a cron job to delete AI summaries, see ["Set up a cron job to delete AI summaries" on page 212](#).

DELETE /api/v10/AiAnalysis/removeAiSummaries

#### Description

Delete the AI summaries that are older than a given amount of time.

#### Example usage

Delete AI summaries that are no longer required (they are older than 30 days).

```
curl -X DELETE -H "Content-Type: application/json" -u "username:ticket" "https://my-swarm-host/api/v10/AiAnalysis/removeAiSummaries"
```

Swarm responds with:

```
{
 "error": null,
 "messages": [],
 "data": [
 {
 "aiSummary": [
 {
 "id": 2,
 "updated": "2025-Mar-30"
 }
]
 }
]
}
```

```
]
 },
 {
 "code": "aiSummary-obliterated",
 "text": "Ai summary of records has been obliterated for records which are older than 30
days"
 }
]
}
```

**If a request fails**

<error code>:

- 401 unauthorized
- 403 AI feature is not enabled or you do not have admin privileges

## Get AI summary for a diff in the review

**Summary**

Get the AI-generated code explanation for a diff between the specified versions of a review.

GET /api/v10/reviews/{reviewid}/aiReviewSummary?fromVersion={x}&toVersion={y}

**Description**

Get the AI-generated code explanation for a diff between the specified versions of a review.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
reviewId	Review ID	integer	query	Yes

Parameter	Description	Type	Parameter Type	Required
toVersion	<p>Specify a value for toVersion:</p> <ul style="list-style-type: none"><li>■ If omitted, toVersion defaults to the latest version of the review.</li><li>■ Cannot be 0 or -1</li><li>■ Must not be greater than the latest version of the review.</li></ul>	integer	query	Yes

Parameter	Description	Type	Parameter Type	Required
fromVersion	<p>Specify a value for fromVersion:</p> <ul style="list-style-type: none"><li>0 specifies the base version of the review.</li><li>-1 specifies the current #head version of the review files in the P4 Server.</li><li>If omitted, fromVersion defaults to 0 which is the base version of the review.</li><li>Must not be greater than the latest version of the review.</li></ul>	integer	query	Yes

### Example usage

Get AI summary for review 12169 from version 0 to 3

```
curl -X GET -H "Content-Type: application/json" -u "username:ticket" "https://my-swarm-host/api/v10/reviews/12169/aiReviewSummary?fromVersion=0&toVersion=3"
```

P4 Code Review responds with:

```
{
 "error": null,
 "messages": [],
 "data": {
 "aiSummaries": {
 "25": {
 "12169": {
```

```

"0_3": {
 "Ly9kZXBvdC9KYW0vTUFJTj9zcmMvZXhwYW5kLmM": {
 "37_39": {
 "1": {
 "ai_package_id": "1",
 "users": [
 "bruno",
 "bruno",
 "me",
 "mei"
],
 "digest": "4AB81C92FF14518BC660CAF516066CC0",
 "content": "The given code is not complete, but it includes a declaration of a function
 named `edit`. This function is marked as `static`, which means that it can
 only be called from within the same file.\n\nHere's some additional insight
 into what we can infer from this partial code snippet:\n\n- `static` in
 C programming: The `static` keyword defines the scope to the same source
 code file. In other words, static functions are functions that are only
 visible to other functions in the same file (more precisely, the same
 translation unit).\n\n- `void` in C programming: The `void` keyword in
 function declaration represents that the function doesn't return any
 value.\n\n- `edit`: This is the name of the function.\n\n- `()` in
 C programming: The parentheses are used to list the arguments that a
 function takes. If there is nothing in the parentheses (like you have here
 in the `edit()` function), that means that the function does not take any
 arguments. \n\nSo, a fleshed out version of this function, and one way
 to use it in code, might look something like:\n\n```\nC\nstatic void edit()\n{\n
 // function body\n // This is where the code to execute when edit() is called
 would go.\n}\n \nint main()\n{\n // call the function\n edit();\n \n
 return 0;\n}\n```\nThis function when called, runs the code inside the edit()
 function. Before calling this function, it must be defined in the same
 file.\n\nIt is recommended to provide a complete code snippet to get the full
 explanations.",
 "error": false
 }
 }
 }
},
"id": 25,
"topic": "review/12169",
"context": [],
"flags": [],
"time": 1730098608,
"updated": 1730098608,
"edited": null,
"isVisible": true
},

```

```

"26": {
 "12169": {
 "0_3": {
 "Ly9kZXBvdC9KYW0vTUFJT9zcmMvZXhwYW5kLmM": {
 "34_36": {
 "1": {
 "ai_package_id": "1",
 "users": [
 "mei",
 "bruno"
],
 "digest": "4AB81C92FF14518BC660CAF516066CC0",
 "content": "The provided code seems incomplete. Here's some general interpretation
 derived from it:\n\n1. `expand.c` is a C source file. The code inside
 relates to functionality contained in a file named `expand.c`.\n\n2.
 `static void new_edit();` is defining a function prototype. \n\n
 - `static` is a storage class specifier. It tells the compiler that the
 function `new_edit` has internal linkage (its scope is limited to its
 containing file - in this case `expand.c`) other than external linkage
 (able to be referenced from other C files). \n\n - `void` indicates
 that `new_edit` does not return any value. \n\n - `new_edit();` is the
 function name followed by an empty parameter list. It suggests that `new_edit`
 function does not take any arguments.\n\nHowever, without more context or
 clarification, I could only provide generic details about what the certain
 parts do. The function as to what it does isn't clear based on what is
 provided.",
 "error": false
 }
 }
 }
 }
 },
 "id": 26,
 "topic": "review/12169",
 "context": [],
 "flags": [],
 "time": 1730102666,
 "updated": 1730102666,
 "edited": null,
 "isVisible": true
},
"27": {
 "12169": {
 "0_3": {
 "Ly9kZXBvdC9KYW0vABCDEF3zcmMvZXhwTC7qOpP": {
 "12_13": {
 "1": {
 "ai_package_id": "1",

```

```
"users": [
 "bruno"
],
"digest": "4AB81C92FF14518BC660CAF516066AA0",
"content": "This code is part of a C program. The 'static' keyword before the function
 declaration means that the function 'modes' is only visible or usable
 within the file it's declared ('explain.c' in this case). It has file
 scope, which means other files that are part of the same project will
 not be able to directly call this function.\n\nThe 'void' keyword means
 this function does not return any value.\n\nSo 'static void modes();'
 is a prototype for a function named 'modes', which does not accept any
 parameters, does not provide a return value and is only accessible within
 the file explain.c. The actual definition of the function where the
 functionality of the function is implemented will be found elsewhere in
 the file.",
"error": false
}
}
}
},
"id": 27,
"topic": "review/12169",
"context": [],
"flags": [],
"time": 1730103198,
"updated": 1730103198,
"edited": null,
"isVisible": true
}
},
"totalCount": 3
}
}
```

If a request fails

<error code>:

- 401 unauthorized
- 404 cannot fetch entry, Id does not exist



## Attachments: P4 Code Review comment attachments

### Get attachments details

#### Summary

Get attachment details

GET /api/v10/attachments?ids[]={id}

#### Description

Get details of attachments the user has permission to view.

#### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Specify the attachments you want details returned for by specifying attachment ids on the request.	Array of ids (integers)	query	Yes

#### Example usage

Get details for attachments 1, 2, and 500

```
curl -u "username:ticket" "https://myswarm-url/api/v10/attachments?ids[]=1&ids[]=2&ids[]=500"
```

P4 Code Review responds with details for attachment ids 1, 2, and 500:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "attachments": [
 {
 "id": 1,
 "name": "motorbike.png",
 "type": "image/png",
```

```

 "size": 19844,
 "depotFile": "///swarm/attachments/0000000001-motorbike.png",
 "references": {
 "comment": [
 502
]
 }
 },
 {
 "id": 2,
 "name": "house.png",
 "type": "image/png",
 "size": 16109,
 "depotFile": "///swarm/attachments/0000000002-house.png",
 "references": {
 "comment": [
 503
]
 }
 },
 {
 "id": 500,
 "name": "car.png",
 "type": "image/png",
 "size": 26123,
 "depotFile": "///swarm/attachments/0000000500-car.png",
 "references": {
 "comment": [
 890
]
 }
 }
]
}
}

```

If a request fails

<error code>:

**404** Attachment does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }]
}

```

```
 },
 "data" : null
}
```

## Get attachment thumbnail

### Summary

Get the thumbnail of an attachment.

GET /api/v10/attachments/{id}/content/{filename}/thumb

### Description

Get the thumbnail of an attachment the user has permission to view.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Attachment id	integer	query	Yes
filename	This is the attachment filename and extension encoded to <b>URL safe Base64</b> . Various tools are available online to encode the filename to <b>URL safe Base64</b> , for example: <a href="#">BASE64 Decode and Encode</a> .	string	path	Yes

### Example usage

Get the thumbnail for attachment ID 118 ( house.png which is aG91c2UucG5n when encoded in **URL safe Base64**):

```
curl -u "username:ticket" "https://myswarm-url/api/v10/attachments/118/content/aG91c2UucG5n/thumb"
```

P4 Code Review responds with the thumbnail for the attachment.

#### If a request fails

<error code>:

**404** Attachment does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Open an attachment

#### Summary

Open an attachment.

GET /api/v10/attachments/{id}/content/{filename}

#### Description

Open the attachment the user has permission view.

Parameter	Description	Type	Parameter Type	Required
id	Attachment id	integer	query	Yes

Parameter	Description	Type	Parameter Type	Required
filename	This is the attachment filename and extension encoded to <b>URL safe Base64</b> . Various tools are available online to encode the filename to <b>URL safe Base64</b> , for example: <a href="#">BASE64 Decode and Encode</a> .	string	path	Yes

#### Example usage

Open attachment ID 118 ( house.png which is aG91c2UucG5n when encoded in **URL safe Base64**):

```
curl -u "username:ticket" "https://myswarm-url/api/v10/attachments/118/content/aG91c2UucG5n"
```

P4 Code Review opens the attachment.

#### If a request fails

<error code>:

**404** Attachment does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Download an attachment

#### Summary

Download an attachment.

GET /api/v10/attachments/{id}/content/{filename}/download

#### Description

Download the attachment the user has permission to view.

#### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Attachment id	integer	query	Yes
filename	This is the attachment filename and extension encoded to <b>URL safe Base64</b> . Various tools are available online to encode the filename to <b>URL safe Base64</b> , for example: <a href="#">BASE64 Decode and Encode</a> .	string	path	Yes

---

#### Example usage

Download attachment ID 118 ( house.png which is aG91c2UucG5n when encoded in **URL safe Base64**):

```
curl -u "username:ticket" "https://myswarm-url/api/v10/attachments/118/content/aG91c2UucG5n/download"
```

P4 Code Review responds by downloading the attachment.

#### If a request fails

<error code>:

**404** Attachment does not exist

HTTP/1.1 <response error code>

{

```
"error": <error code>,
"messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
}],
"data" : null
}
```

## Create an attachment

### Summary

Create an attachment.

POST /api/v10/attachments

### Description

Create a standalone attachment.

Once the attachment is created, P4 Code Review dynamically assigns a unique id to the attachment. To add an attachment to a comment, pass the unique attachment id as a parameter using the ["Add a comment with an attachment to a review" on page 1067](#) API endpoint.

### Parameters

Parameter	Description	Type	Parameter Type	Required
file	This is the complete path with the attachment file name.	string	path	Yes

### Example usage

Create an attachment for the house.png file:

```
curl -X POST -H "Content-Type: multipart/form-data" -u "username:ticket" "https://myswarm-url/api/v10/attachments/" -F "file=@/home/perforce/client1/house.jpg"
```

P4 Code Review responds with the details of the attachment:

HTTP/1.1 200 OK

```
{
```

```
"error": null,
"messages": [],
"data": {
 "attachments": [
 {
 "id": 12,
 "name": "house.png",
 "type": "image/png",
 "size": 106689,
 "blur": "L1SPX_I,Mw~W~qofa{WV56yDcDkq",
 "depotFile": "//depot/Jam/MAIN/src/attachments/0000000118-house.png",
 "references": []
 }
]
}
```

#### If a request fails

<error code>:

**404** Attachment does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Changes: P4 Code Review changes

### Get a list of changes

#### Summary

Get a list of changes.

GET /api/v10/changes

#### Description

Get a list of changes the user is able to view.



**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
pending	<p>Determines if all changelists, only pending changelists, or only submitted changelists are returned by the request.</p> <ul style="list-style-type: none"> <li>▪ Not set: All changelists (submitted and pending) are returned.</li> <li>▪ true: Only pending changelists are returned.</li> <li>▪ false: Only submitted changelists are returned.</li> </ul>	boolean	query	No
user	The user parameter enables you to return just the changelists created by a specific user.	string	query	No

Parameter	Description	Type	Parameter Type	Required
rootPath	<p>The rootPath parameter enables you to return all the changelists in a specific root path. The rootPath parameter is encoded to <b>URL safe Base64</b>.</p> <p>Various tools are available online to encode the id to <b>URL safe Base64</b>, for example: <a href="https://www.base64encode.org">https://www.base64encode.org</a></p>	string	query	No

### Example usage

#### Get a list of changes

```
curl -u "username:ticket" "https://myswarm-url/api/v10/changes"
```

P4 Code Review responds with the changes starting with the newest change:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "changes": [
 {
 "id": 12853,
 "date": "2022/04/14 08:59:59",
 "client": "swarm-test",
 "user": "bruno",
 "status": "submitted",
 "type": "public",
 "importedBy": null,
 "identity": null,
 "description": "Fix display bug. ",
 "jobStatus": null,
 "jobs": [],
 "stream": null,
 "files": [
```

```

 "///depot/main/myfile.txt#6"
]
},
{
 ...
 <other change ids formatted as above>
 ...
}
]
}
}

```

#### Get a list of the pending changes

```
curl -u "username:ticket" "https://myswarm-url/api/v10/changes?pending=true"
```

P4 Code Review responds with a list of pending changes starting with the newest pending change:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "changes": [
 {
 "id": 12382,
 "date": "2022/04/10 12:59:59",
 "client": "swarm-test",
 "user": "paula.boyle",
 "status": "pending",
 "type": "public",
 "importedBy": null,
 "identity": null,
 "description": "Update license details. ",
 "jobStatus": null,
 "jobs": [],
 "stream": null,
 "files": [
 "///depot/main/license.txt#2"
]
 },
 {
 ...
 <other change ids formatted as above>
 ...
 }
]
 }
}

```

```
}
}
```

Get a list of changes created by Bruno

```
curl -u "username:ticket" "https://myswarm-url/api/v10/changes?user=bruno"
```

P4 Code Review responds with the changes created by Bruno starting with the newest change:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "changes": [
 {
 "id": 12853,
 "date": "2022/04/14 08:59:59",
 "client": "swarm-test",
 "user": "bruno",
 "status": "submitted",
 "type": "public",
 "importedBy": null,
 "identity": null,
 "description": "Fix display bug. ",
 "jobStatus": null,
 "jobs": [],
 "stream": null,
 "files": [
 "//depot/main/myfile.txt#6"
]
 },
 {
 ...
 <other change ids formatted as above>
 ...
 }
]
 }
}
```

Get a list of changes in //depot/Jam/MAIN/...

//depot/Jam/MAIN/... encoded to **URL safe Base64** is Ly9kZXBvdC9KYW0vTUFJTj8uLi4=.

```
curl -u "username:ticket" "https://myswarm-
url/api/v10/changes?rootPath=Ly9kZXBvdC9KYW0vTUFJTj8uLi4="
```

P4 Code Review responds with changelists in //depot/Jam/MAIN/... starting with the newest change:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "changes": [
 {
 "id": 12853,
 "date": "2022/04/14 08:59:59",
 "client": "swarm-test",
 "user": "paula.boyle",
 "status": "pending",
 "type": "public",
 "importedBy": null,
 "identity": null,
 "description": "Update the XML and Word files. ",
 "jobStatus": null,
 "jobs": [],
 "stream": null,
 "files": [
 "//depot/Jam/MAIN/src/anotherfile.xml#3",
 "//depot/Jam/MAIN/src/myfile.xml#15",
 "//depot/Jam/MAIN/src/yetanotherfile.docx#4"
]
 },
 {
 ...
 <other change ids formatted as above>
 ...
 }
]
 }
}
```

If a request fails

<error code>:

**403** Insufficient permissions to access the changes

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",
 "text": "<error message>"
 }],
}
```

```
"data" : null
}
```

## Get a list of files on a change

### Summary

Get a list of files on a change.

GET /api/v10/changes/{id}/files

### Description

Get a list of files on a change. You must be authenticated to view the files on the change.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Change ID	integer	path	Yes

### Example usage

Get a list of files on change 12106:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/changes/12106/files"
```

P4 Code Review responds with a list of files on change 12106:

```
HTTP/1.1 200 OK
```

```
{
 "error": null,
 "messages": [],
 "data" : {
```

```

"root": "//depot/triggers",
"files": [
 {
 "depotFile": "//depot/triggers/swarm-trigger.conf",
 "action": "add",
 "type": "text",
 "rev": "1",
 "fileSize": "129",
 "digest": "635CC5C20C0C2F3534D94207C7F7FB25"
 },
 {
 "depotFile": "//depot/triggers/swarm-trigger.pl",
 "action": "add",
 "type": "text+x",
 "rev": "1",
 "fileSize": "51239",
 "digest": "0C0E14975CD4E6195E95805DEC98DA03"
 }
],
"limited": false
}
}

```

**If a request fails**

<error code>:

- **403** Insufficient permissions to access the change
- **404** Change does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",
 "text": "<error message>"
 }],
 "data": null
}

```

## Get job information for jobs associated with a change

**Summary**

Get job information for jobs associated with a change

GET /api/v10/changes/{id}/jobs

## Description

Get job information for jobs associated with a change. You must be authenticated to view the jobs associated with the change.

### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

## Parameters

Parameter	Description	Type	Parameter Type	Required
id	Change ID	integer	path	Yes

## Example usage

Get job information for change 12344

```
curl -u "username:<ticket>" "https://myswarm-url/api/v10/changes/12344/jobs"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job000020",
 "link": "/jobs/job000020",
 "fixStatus": "open",
 "description": "Fix final bugs for beta release.\n",
 "description-markdown": "Fix final bugs for beta
release."
 }
]
 }
}
```



```
}
}
```

If a request fails

<error code>:

- **403** Insufficient permissions to access the change
- **404** Change does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Add a job to a change

### Summary

Add a job to a change.

PUT /api/v10/changes/{id}/jobs/{job\_id}

### Description

Add a job to a change. You must be authenticated to add a job to a change.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Change ID	integer	path	Yes
job_id	Job ID	integer	path	Yes

**Example usage****Add a job to a change**

Add job000020 to change12344

```
curl -u "username:<ticket>" PUT "https://myswarm-url/api/v10/changes/12344/jobs/job000020"
```

P4 Code Review responds with all of the jobs associated with the change:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job000020",
 "link": "/jobs/job000020",
 "fixStatus": "open",
 "description": "Fix final bugs for beta release.\n",
 "description-markdown": "Fix final bugs for beta
release."
 }
]
 }
}
```

**Add a job to a change that is already associated with another job**

Add job001001 to change12344

```
curl -u "username:<ticket>" PUT "https://myswarm-url/api/v10/changes/12344/jobs/job001001"
```

P4 Code Review responds with all of the jobs associated with the change:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job000020",
 "link": "/jobs/job000020",
 "fixStatus": "open",
 "description": "Fix final bugs for beta release.\n",
 "description-markdown": "Fix final bugs for beta
release."
 },
 {
 "job": "job001001",
 "link": "/jobs/job001001",
 "fixStatus": "fixed",
 "description": "Bug fixes by Terry\n",
 "descriptionMarkdown": "Bug fixes by Terry"
 }
]
 }
}
```

If a request fails

<error code>:

- **403** Insufficient permissions to access the change
- **404** Change does not exist
- **500** Job does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Remove a job from a change

### Summary

Remove a job from a change.

DELETE /api/v10/changes/{id}/jobs/{job\_id}

### Description

Remove a job from a change. You must be authenticated to remove a job from a change.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Change ID	integer	path	Yes
job_id	Job ID	integer	path	Yes

### Example usage

#### Remove a job from a change

Remove job000020 from change12344:

```
curl -u "username:<ticket>" PUT "https://myswarm-url/api/v10/changes/12344/jobs/job000020"
```

P4 Code Review responds with all of the jobs associated with the change:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
}
```

```

 "data" : {
 "jobs": []
 }
 }
}

```

Remove a job from a change that is associated with more than one job

Remove job000020 from change12344:

```
curl -u "username:<ticket>" PUT "https://myswarm-url/api/v10/changes/12344/jobs/job000020"
```

P4 Code Review responds with all of the jobs associated with the change:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data" : {
 "jobs": [
 {
 "job": "job001001",
 "link": "/jobs/job001001",
 "fixStatus": "fixed",
 "description": "Bug fixes by Terry\n",
 "description-markdown": "Bug fixes by Terry"
 }
]
 }
}

```

If a request fails

<error code>:

- **403** Insufficient permissions to access the change
- **404** Change does not exist
- **500** Job does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Comments: P4 Code Review comments

### Get comments for a review, change, or job

#### Summary

Get a list of comments for a review, change, or job.

GET /api/v10/comments/<topic\_subject>/<topic\_id>

#### Description

Get a list of comments for a review, change, or job. Only comments for the specified `topic_subject` and `topic_id` are returned. Valid `topic_subject` values are `reviews`, `changes`, and `jobs`.

#### For example:

- GET /api/v10/comments/reviews/1234
- GET /api/v10/comments/changes/1234
- GET /api/v10/comments/jobs/job001234

#### Parameters

Parameter	Description	Type	Parameter Type	Required	Default Value
<code>after</code>	A comment ID to seek to. Comments up to and including the specified ID are excluded from the results and do not count towards <code>max</code> . Useful for pagination. Commonly set to the <code>lastSeen</code> property from a previous query.	integer	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
max	Maximum number of comments to return. This does not guarantee that max comments are returned. It does guarantee that the number of comments returned won't exceed max.	integer	query	No	100
ignoreArchived	Prevents archived comments from being returned.	boolean	query	No	
tasksOnly	Returns only comments that have been flagged as tasks.	boolean	query	No	

Parameter	Description	Type	Parameter Type	Required	Default Value
<code>taskState</code>	Limit the returned comments to ones that match the provided task state (one or more of <code>open</code> , <code>closed</code> , <code>verified</code> , or <code>comment</code> ).	array (of strings)	query	No	
<code>fields</code>	An optional comma-separated list (or array) of fields to show for each comment. Omitting this parameter or passing an empty value shows all fields.	string	query	No	

### Example usage

#### Get all comments for a review

Get all of the comments for review 885:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/comments/reviews/885"
```

P4 Code Review responds with the comments for the specified review:

**Tip:**

By default, this is limited to the 100 most recent comments. This can be changed by adding a `max` parameter value to the request.



```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 51,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bruno"
 "time": 1619823600,
 "updated": 1619823600,
 "edited": null,
 "body": "This is a really good comment."
 "readBy": []
 },
 {
 "id": 63,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "alice"
 "time": 1620799560,
 "updated": 1620799560,
 "edited": null,
 "body": "This is my comment.",
 "readBy": []
 },
 {
 "id": 95,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bob"
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is another comment.",

```

```
 "readBy": []
 },
],
"lastSeen": 95,
"totalCount": 3
}
}
```

**Get a list of the first two comments for a review**

Get a list of the first two comments for review 885:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/comments/reviews/885&max=2"
```

P4 Code Review responds with a list of the first two comments for review 885 and a `lastSeen` value for pagination:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 51,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bruno"
 "time": 1619823600,
 "updated": 1619823600,
 "edited": null,
 "body": "This is a really good comment.",
 "readBy": []
 },
 {
 "id": 63,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "alice"
 "time": 1620799560,
 "updated": 1620799560,
 "edited": null,
 "body": "This is my comment.",

```

```

 "readBy": [],
 },
],
"lastSeen": 63,
"totalCount": 2
}
}

```

#### Paginating a comment listing

To get the next page of a comments list (based on the example above):

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/comments/reviews/885&max=2&after=63"
```

P4 Code Review responds with the second page of results, if any comments are present after the last seen comment:

```

{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bob"
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is another comment.",
 "readBy": []
 },
],
 "lastSeen": 95,
 "totalCount": 1
 }
}

```

#### If a request fails

```
<error code>:
```

- 401 you are not authorized to view the comments in the specified review
- 404 one of the following:
  - the `topic_subject` specified is not valid, it must be a reviews, changes, or jobs
  - the `topic_id` id specified does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get a comment by ID

### Summary

Get comment details by comment id.

GET /api/v10/comments/<id>

### Description

Get comment details by comment id if the user has permission to view it. If the comment is a reply, the parent comment id is also returned.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Example usage

Get comment id 1627

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/comments/1627"
```

P4 Code Review responds with the comment details:

```

{
 "error": null,
 "messages": [],
 "data": {
 "comments":
 "id": 1627,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": 46,
 "rightLine": 48,
 "content": [
 " ",
 " // Commodo magnis posuere lobortis.",
 "- tincidunt = cumLigula * 100;",
 "- data++;",
 " int suspendisse = 0;"
],
 "type": "text",
 "review": 2004,
 "version": 1,
 "change": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 48
 },
 "attachments": [
 17,
 18,
 107,
 108,
 110,
 111
],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "aaron.gleber",
 "time": 1649147498,
 "updated": 1650640678,
 "edited": 1650638878,
 "body": "This comment gives some examples of images you can use.",
 "readBy": []
 }
}

```

Get the parent comment of comment id 525

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/comments/525"
```

P4 Code Review responds with comment id 501 which is the parent of comment id 525:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 525,
 "topic": "reviews/12373",
 "context": {
 "comment": 501,
 "version": 2
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "alice",
 "time": 1622712940,
 "updated": 1622712940,
 "edited": null,
 "body": "dao save test 1",
 "readBy": [],
 "notify": "delayed"
 }
]
 }
}
```

#### If a request fails

<error code>:

- 401 you are not authorized to view the comments
- 404 the comment id specified does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Add a comment

### Summary

Add a comment to a review, change, or job.

POST /api/v10/comments/<topic\_subject>/<topic\_id>

### Description

Add a comment to a review, change, or job. The user that makes the API request is set as comment author.

Valid `topic_subject` values are `reviews`, `changes`, and `jobs`.

### For example:

- POST /api/v10/comments/reviews/1234
- POST /api/v10/comments/changes/1234
- POST /api/v10/comments/jobs/job001234

### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>body</code>	The text content of the comment.	string	form	Yes
<code>taskState</code>	Optional task state of the comment. Valid values when adding a comment are <code>comment</code> (default) and <code>open</code> . This creates a plain comment or opens a task, respectively.  If the <code>taskState</code> parameter is not in the request, the task state is set to <code>comment</code> .	string	form	No

Parameter	Description	Type	Parameter Type	Required
notify	<p>Comment notifications can be delayed, see <a href="#">"Comment notification delay" on page 336</a>. The notify parameter enables you to set when the notification for the comment is sent.</p> <p>Valid options are:</p> <ul style="list-style-type: none"><li>■ <b>delayed</b> (default): notifications for this comment are delayed by the time set in <a href="#">"Comment notification delay" on page 580</a></li><li>■ <b>immediate</b>: the notification for this comment is sent immediately</li><li>■ <b>silent</b>: no notification is sent for this comment</li></ul>	string	query	No

---



Parameter	Description	Type	Parameter Type	Required
context	<p>Used to add the comment to a file in the review.</p> <p>Valid fields are:</p> <ul style="list-style-type: none"><li>▪ file mandatory unless attribute or comment are set: File to comment on. Valid only for changes and reviews topics. For example, //depot/main/README.txt.</li><li>▪ leftline optional, but if specified, you must also specify the rightline and content parameters. Integer: Left-side diff line number to attach the inline comment to. Valid only for changes and reviews topics.</li><li>▪ rightline optional, but if specified, you must also specify the leftline and content parameters. Integer: Right-side diff line number to attach the inline comment to. Valid only for changes and reviews topics.</li></ul>	array	form	No

Parameter	Description	Type	Parameter Type	Required
	<ul style="list-style-type: none"><li>■ content optional, but if specified, you must also specify the leftline and rightline parameters. Array of strings: Provide the content of the codeline the comment is on and the four preceding codelines. This is used to specify a short excerpt of context in case the lines being commented on change during the review. When set to null, P4 Code Review makes an effort to build the content on its own. Because this involves file operations, it might be slow.</li><li>■ version optional, integer: With a reviews topic, this field specifies which version to attach the comment to.</li><li>■ attribute optional: Set to description to comment on the review description.</li><li>■ comment optional, integer: Set to the comment id this comment is replying to.</li></ul>			

## Example usage

### Add a comment to a review

Add a comment to review 885, make the comment an open task, and send the notification immediately:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/comments/reviews/885?notify=immediate"
```

The "mybodyfilename.txt" file contains the authenticated user's comment and makes the comment an open task:

```
{
 "body": "This is another comment.",
 "taskState": "open"
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 95,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "open",
 "likes": [],
 "user": "bob",
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is another comment.",
 "readBy": [],
 "notify": "immediate"
 },
],
 "userDelayedNotificationCount": 1
 }
}
```

### Add a comment to a file in a review

Add a comment to the end of the //depot/main/README.txt file on review 123:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/comments/reviews/123"
```

The "mybodyfilename.txt" file contains the authenticated user's comment for the file:

```
{
 "body": "This is a comment on a file.",
 "context": {
 "file": "///depot/main/README.txt"
 }
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 105,
 "topic": "reviews/123",
 "context": {
 "file": "///depot/main/README.txt",
 "leftLine": null,
 "rightLine": null,
 "content": [],
 "version": 1,
 "change": 122,
 "review": 123,
 "md5": "6af0e07c2d96afacaf2da234242c3cbd",
 "name": "README.txt"
 },
 "line": null
 },
 {
 "attachments": [],
 "flags": [],
 "taskState": "open",
 "likes": [],
 "user": "bob"
 },
 {
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is a comment on a file.",
 "readBy": [],
 "notify": "delayed"
 }
],
 "userDelayedNotificationCount": 1
 }
}
```

**Add a comment to a line in a file and include context for the comment****Tip:**

To populate the content field, you need to know the content of the line you are commenting on and the content of the four lines before the line you are commenting on

Add a comment to a line in a file on review 110 with the following information:

- Full filepath and filename of the file to comment on
- Left and right line numbers of the diff to comment on
- Text of the codeline you are commenting on and the four codelines before that to provide context for the comment
- Review version to add the comment to

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/comments/reviews/110"
```

The "mybodyfilename.txt" file contains the details for the new comment:

```
{
 "body": "This is a comment on a line in a file with context.",
 "context": {
 "file": "//depot/main/README.txt"
 "rightLine": 54,
 "leftLine": null,
 "content": {
 "FORTRAN default = \"\" ;\n",
 "LIBDIR default = /boot/develop/libraries ;\n",
 "-LINK default = $(CC) ;\n",
 "+LINK default = mwld ;\n",
 "+LINKFLAGS default = \"\" ;\n"
 },
 "version": 3
 }
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 143,
 "topic": "reviews/110",
 "context": {
 "file": "//depot/main/README.txt"
 "leftLine": null,

```

```
 "rightLine": 54,
 "content": [
 "FORTRAN default = \"\" ;\n",
 "LIBDIR default = /boot/develop/libraries ;\n",
 "-LINK default = $(CC) ;\n",
 "+LINK default = mwld ;\n",
 "+LINKFLAGS default = \"\" ;\n"
],
 "version": 3,
 "change": 109,
 "review": 110,
 "md5": "6af0e07c2d96afacaf2da234242c3cbd",
 "name": "README.txt",
 "line": 54
 }
 "attachments": [],
 "flags": [],
 "taskState": "open",
 "likes": [],
 "user": "bob"
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is a comment on a line in a file with context.",
 "readBy": [],
 "notify": "delayed"
},
],
"userDelayedNotificationCount": 1
}
```

#### Add a comment to a review description

Add a comment to the description of review 885 and delay the notification:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/comments/reviews/885?notify=delayed"
```

The "mybodyfilename.txt" file contains the authenticated user's comment and adds it to the review description:

```
{
 "body": "This is a comment on the description.",
 "context": {
 "attribute": "description"
 },
}
```

P4 Code Review responds the comment entity:

```

{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 99,
 "topic": "reviews/885",
 "context": {
 "attribute": "description"
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bob"
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is a comment on the description.",
 "readBy": [],
 "notify": "delayed"
 },
],
 "userDelayedNotificationCount": 1
 }
}

```

#### Reply to a comment

You have two options when replying to a comment. Although the URLs for the options are different, they both do the same job:

- ["Reply to a comment using POST /api/v10/comments/<topic\\_subject>/<topic\\_id>" below](#)
- ["Reply to a comment using POST /api/v10/comments/<id>" on the next page](#)

#### Reply to a comment using POST /api/v10/comments/<topic\_subject>/<topic\_id>

Add a reply to comment id 99 in review 885:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/comments/reviews/885"
```

The "mybodyfilename.txt" file contains the authenticated user's comment text and the comment the reply is for:

```

{
 "body": "This is a reply to a comment.",
 "context": {
 "comment": 99
 }
}

```

```
}
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 109,
 "topic": "reviews/885",
 "context": {
 "comment": 99,
 "version": 2
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bruno"
 },
 {
 "time": 1622712940,
 "updated": 1622712940,
 "edited": null,
 "body": "This is a reply to a comment.",
 "readBy": [],
 "notify": "delayed"
 }
],
 "userDelayedNotificationCount": 1
 }
}
```

#### Reply to a comment using POST /api/v10/comments/<id>

Add a reply to comment id 99 in review 885:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/comments/99"
```

The "mybodyfilename.txt" file contains the authenticated user's comment text and the comment the reply is for:

```
{
 "body": "This is a reply to a comment.",
 "topic": "reviews/885"
}
```

P4 Code Review responds the comment entity:



```

{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 120,
 "topic": "reviews/885",
 "context": {
 "comment": 99,
 "version": 2
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bruno"
 "time": 1622712940,
 "updated": 1622712940,
 "edited": null,
 "body": "This is another reply to a comment.",
 "readBy": [],
 "notify": "delayed"
 }
]
 "userDelayedNotificationCount": 1
 }
}

```

#### If a request fails

<error code>:

- 400 one of the following:
  - the `idspecified` must be an integer greater than 0
  - invalid `context` field specified, valid fields are `file`, `leftline`, `rightline`, `content`, `version`, `attribute`, and `comment`
  - invalid `notify` value specified, value must be `delayed`, `immediate`, or `silent`
  - invalid `taskState` value specified, value must be `open` or `comment`
- 403: you are not authorized to create this comment
- 404 the review, change, job, review version, file line number, or file specified does not exist
- 409 invalid `topic_subject` specified, value must be `reviews`, `changes`, or `jobs`
- 500 internal error, request failed

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Edit a comment

### Summary

Edit a comment.

POST /api/v10/comments/{comment\_id}/edit

### Description

**Important:**

Only the author of the original comment can edit it. Users with *admin* or *super* permissions cannot edit a comment by another user.

Edit a comment. Only the parameters specified in the API request are updated for the comment, the rest of the comment parameters are left unchanged.

When a comment is edited:

- The comment time stamp is marked as (*edited*) to show that the comment has been changed.
- All of the likes for the comment are removed.
- If users have marked the comment as read, the comment read status is reset to unread for those users.
- P4 Code Review sends a notification to everyone involved in the review, including the review author and the reviewers, but not the editor of the comment.

### Parameters

Parameter	Description	Type	Parameter Type	Required
body	The text content of the comment.	string	form	No

Parameter	Description	Type	Parameter Type	Required
taskState	<p>The task state of the comment. Valid values are comment, open, addressed, and verified.</p> <p>Certain transitions (such as moving from open to verified) are not possible without an intermediate step (addressed, in this case).</p>	string	form	No

Parameter	Description	Type	Parameter Type	Required
<code>notify</code>	<p>Comment notifications can be delayed, see <a href="#">"Comment notification delay" on page 336</a>. The <code>notify</code> parameter enables you to set when the notification for the comment is sent.</p> <p>Valid options are:</p> <ul style="list-style-type: none"><li>■ <code>delayed</code> (default): notifications for this comment are delayed by the time set in <a href="#">"Comment notification delay" on page 580</a></li><li>■ <code>immediate</code>: the notification for this comment is sent immediately</li><li>■ <code>silent</code>: no notification is sent for this comment</li></ul>	string	query	No

---

**Edit a comment**

Edit the text of comment id 95, set its task state to addressed, and send the comment notification immediately:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://my-swarm-host/api/v10/comments/95/edit?notify=immediate"
```

The "mybodyfilename.txt" file contains the comment author's edited text and the new task state value:

```
{
 "body": "This is an edit to this comment.",
 "taskState": "addressed"
}
```

P4 Code Review responds with:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 95,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "addressed",
 "likes": [],
 "user": "bob"
 "time": 1620813784,
 "updated": 1620813784,
 "edited": true,
 "body": "This is an edit to this comment.",
 "readBy": [],
 "notify": "immediate"
 },
],
 "userDelayedNotificationCount": 1
 }
}
```

#### If a request fails

<error code>:

- 400 one of the following:
  - the comment-idspecified must be an integer greater than 0
  - invalid notify value specified, value must be delayed, immediate, or silent
  - invalid taskState value specified, value must be open, comment, addressed, or verified
- 403: you are not authorized to edit this comment, you must be the comment author
- 404 the comment\_id specified does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Mark a comment as read

### Summary

Mark a comment as read for the authenticated user.

POST /api/v10/comments/{id}/read

### Description

Mark a comment as read for the authenticated user.

### Example usage

Mark comment id 1626 as read for Bruno:

```
curl -X POST -u "bruno:ticket" "https://my-swarm-host/api/v10/comments/1626/read"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 1626,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consetetur. Condimentum sed hymenaeos sit, pellentesque",

```

```

 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
],
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36
},
"attachments": [
 14,
 15,
 16
],
"flags": [
 "closed"
],
"taskState": "comment",
"likes": [],
"user": "aaron.gleber",
"time": 1649147456,
"updated": 1650981576,
"edited": 1649147475,
"body": "We have lots of repeated comments.",
"readBy": [
 "bruno"
],
"notify": "delayed"
}
]
}
}
}

```

**If a request fails****<error code>:**

- 400 the comment-idspecified must be an integer greater than 0
- 404 the comment\_id specified does not exist

HTTP/1.1 &lt;response error code&gt;

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],

```

```
"data" : null
}
```

## Mark a comment as unread

### Summary

Mark a comment as unread for the authenticated user.

POST /api/v10/comments/{id}/unread

### Description

Mark a comment as unread for the authenticated user.

### Example usage

Mark comment id 1626 as unread for Bruno:

```
curl -X POST -u "bruno:ticket" "https://my-swarm-host/api/v10/comments/1626/unread"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 1626,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
],
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36
 }
 }
]
 }
}
```



```

 },
 "attachments": [
 14,
 15,
 16
],
 "flags": [
 "closed"
],
 "taskState": "comment",
 "likes": [],
 "user": "aaron.gleber",
 "time": 1649147456,
 "updated": 1650981576,
 "edited": 1649147475,
 "body": "We have lots of repeated comments.",
 "readBy": [],
 "notify": "delayed"
 }
]
}
}

```

#### If a request fails

<error code>:

- 400 the comment-idspecified must be an integer greater than 0
- 404 the comment\_id specified does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Mark all comments read

### Summary

Mark all comments on a review as read for the authenticated user.

POST /api/v10/reviews/{id}/comments/read

## Description

Mark all comments on a review as read for the authenticated user.

## Example usage

Mark comments on review 2004 as read for Vernon Renno:

```
curl -X POST -u "vernon.renno:ticket" "https://my-swarm-host/api/v10/reviews/2004/comments/read"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 1626,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
],
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36
 },
 "attachments": [
 14,
 15,
 16
],
 "flags": [
 "closed"
],
 "taskState": "comment",
 "likes": [],
 "user": "aaron.gleber",
 }
]
 }
}
```

```

 "time": 1649147456,
 "updated": 1650981576,
 "edited": 1649147475,
 "body": "We have lots of repeated comments.",
 "readBy": [
 "vernon.renno"
],
 "notify": "delayed"
 },
 {
 ...
 <other comment ids formatted as above>
 ...
 }
]
}
}

```

**If a request fails****<error code>:**

- 400 the review id specified must be an integer greater than 0
- 404 the review id specified does not exist

HTTP/1.1 &lt;response error code&gt;

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Mark all comments unread

**Summary**

Mark all comments on a review as unread for the authenticated user.

POST /api/v10/reviews/{id}/comments/unread

**Description**

Mark all comments on a review as unread for the authenticated user.

**Example usage**

Mark comments on review 2004 as unread for Vernon Renno:

```
curl -X POST -u "vernon.renno:ticket" "https://my-swarm-host/api/v10/reviews/2004/comments/unread"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 1626,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
],
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36
 },
 "attachments": [
 14,
 15,
 16
],
 "flags": [
 "closed"
],
 "taskState": "comment",
 "likes": [],
 "user": "aaron.gleber",
 "time": 1649147456,
 "updated": 1650981576,
 "edited": 1649147475,
 }
]
 }
}
```

```

 "body": "We have lots of repeated comments.",
 "readBy": [],
 "notify": "delayed"
 },
 {
 ...
 <other comment ids formatted as above>
 ...
 }
]
}
}

```

**If a request fails**

<error code>:

- 400 the review id specified must be an integer greater than 0
- 404 the review id specified does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Archive a comment

**Summary**

Archive a comment.

POST /api/v10/comments/{id}/archive

**Description**

Archive a comment. The Archive option is only available for top level comments, if the comment has replies they are archived with the parent comment.

**Example usage**

Archive comment id 1626:

```
curl -X POST -u "bruno:ticket" "https://my-swarm-host/api/v10/comments/1626/archive"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 1626,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
],
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36
 },
 "attachments": [
 14,
 15,
 16
],
 "flags": [
 "closed"
],
 "taskState": "comment",
 "likes": [],
 "user": "aaron.gleber",
 "time": 1649147456,
 "updated": 1650988005,
 "edited": 1649147475,
 "body": "We have lots of repeated comments.",
 "readBy": [
 "vernon.renno"
],
 "notify": "delayed"
 }
]
 }
}
```

```

 },
 {
 "id": 1628,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
],
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36,
 "comment": 1626
 },
 "attachments": [
 19
],
 "flags": [
 "closed"
],
 "taskState": "comment",
 "likes": [],
 "user": "aaron.gleber",
 "time": 1649147599,
 "updated": 1650988005,
 "edited": 1649147611,
 "body": "This is a reply comment,",
 "readBy": []
 }
]
}
}
}

```

**If a request fails****<error code>:**

- 400 the comment-idspecified must be an integer greater than 0
- 404 the comment\_id specified does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Restore an archived comment

### Summary

Restore an archived comment.

POST /api/v10/comments/{id}/unarchive

### Description

Restore an archived comment. If the archived comment has replies, the replies are also restored.

### Example usage

Restore archived comment id 1626:

```
curl -X POST -u "bruno:ticket" "https://my-swarm-host/api/v10/comments/1626/unarchive"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 {
 "id": 1626,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 " + * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
]
 }
 }
 }
}
```



```

 "+ * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
],
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36
},
"attachments": [
 14,
 15,
 16
],
"flags": [],
"taskState": "comment",
"likes": [],
"user": "aaron.gleber",
"time": 1649147456,
"updated": 1650989315,
"edited": 1649147475,
"body": "We have lots of repeated comments.",
"readBy": [],
"notify": "delayed"
},
{
 "id": 1628,
 "topic": "reviews/2004",
 "context": {
 "file": "//projects/acme/main/src/xml.c",
 "leftLine": null,
 "rightLine": 36,
 "version": 1,
 "content": [
 "/*",
 " * Fringilla dui feugiat natoque. Mauris fames cursus, adipiscing justo",
 " * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 " + * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque",
 " + * arcu consectetur. Condimentum sed hymenaeos sit, pellentesque"
]
 },
 "change": null,
 "review": null,
 "md5": "e66e0d0d88b2f39fb721132adda7ecf5",
 "name": "xml.c",
 "line": 36,
 "comment": 1626
},
"attachments": [
 19

```

```
],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "aaron.gleber",
 "time": 1649147599,
 "updated": 1650989315,
 "edited": 1649147611,
 "body": "This is a reply comment,",
 "readBy": []
 }
]
}
}
```

#### If a request fails

<error code>:

- 400 the comment-idspecified must be an integer greater than 0
- 404 the comment\_id specified does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Add a reaction

### Summary

Add a reaction.

POST /api/v10/comments/{id}/addReaction

### Description

Add a thumbs up to a comment.

## Parameters

Parameter	Description	Type	Parameter Type	Required
reactions	Used to add a reaction icon to a comment. Valid fields are: <ul style="list-style-type: none"><li>thumbsup mandatory, added to a comment.</li></ul>	array	form	Yes

## Example usage

Add a thumbs up to the comment id 28:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://my-swarm-host/api/v10/comments/28/addReaction"
```

The "mybodyfilename.txt" file contains the username and reaction icon:

```
{
 "reactions": {
 "thumbsup": "bruno"
 }
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 28,
 "topic": "reviews/12625",
 "context": {
 "file": "//jam/main/src/make.c",
 "version": 1,
 "leftLine": 5,
 "rightLine": 5,

```

```
"content": [
 " * Copyright 1993, 1995 Christopher Seiwald.",
 " *",
 "- * This file is part of Jam - see jam.c for Copyright information. This is change 1 and this
is change 2 and change 3 change 4",
 "+ * This file is part of Jam - see jam.c for Copyright information. This is change 1 and
this is change 2 and change 3 change 4 and change 4",
 " */"
],
"change": null,
"review": null,
"md5": "8e79caff1ae9cdbd21b1581b554331ba",
"name": "make.c",
"line": 5
},
"attachments": [],
"flags": [],
"taskState": "comment",
"likes": [],
"user": "bruno",
"time": 1690801379,
"updated": 1690801556,
"edited": null,
"body": "THis is test reaction comment",
"readBy": [],
"notify": "delayed",
"reactions": {
 "thumbsup": [
 "bruno"
]
}
}
}
]
```

**If a request fails**

<error code>:

- 400 the `comment-idspecified` must be an integer greater than 0
- 404 the `comment_id` specified does not exist

## Remove a reaction

**Summary**

Remove a reaction.

DELETE /api/v10/comments/{id}/removeReaction

**Description**

Removes the thumbs up from a comment.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
reactions	Used to remove a reaction icon from a comment. Valid fields are: <ul style="list-style-type: none"><li>thumbsup mandatory, removed from a comment.</li></ul>	array	form	Yes

**Example usage**

Remove the thumbs up from the comment id 28:

```
curl -X DELETE -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://my-swarm-host/api/v10/comments/28/removeReaction"
```

The "mybodyfilename.txt" file contains the username and the reaction icon to be removed from the comment:

```
{
 "reactions": {
 "thumbsup": "bruno"
 }
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 28,
```

```
"topic": "reviews/12625",
"context": {
 "file": "//jam/main/src/make.c",
 "version": 1,
 "leftLine": 5,
 "rightLine": 5,
 "content": [
 " * Copyright 1993, 1995 Christopher Seiwald.",
 " *"
],
 "- * This file is part of Jam - see jam.c for Copyright information. This is change 1 and this
is change 2 and change 3 change 4",
 "+ * This file is part of Jam - see jam.c for Copyright information. This is change 1 and
this is change 2 and change 3 change 4 and change 4",
 " */"
],
"change": null,
"review": null,
"md5": "8e79caff1ae9cdbc21b1581b554331ba",
"name": "make.c",
"line": 5
},
"attachments": [],
"flags": [],
"taskState": "comment",
"likes": [],
"user": "bruno",
"time": 1690801379,
"updated": 1690801556,
"edited": null,
"body": "THis is test reaction comment",
"readBy": [],
"reactions": [],
"notify": "delayed"
}
]
}
}
```

**If a request fails**

<error code>:

- 400 the comment-idspecified must be an integer greater than 0
- 404 the comment\_id specified does not exist

## Add a comment with an attachment to a review

### Summary

Add a comment with an attachment to a review.

POST /api/v10/comments/reviews/{id}

### Description

Add a comment with an attachment to a review. This is a two-step process:

1. Create a stand-alone attachment.

Once the attachment is created, P4 Code Review assigns a unique id to the attachment. The unique attachment id is returned in the response. See ["Create an attachment" on page 1011](#).

2. Pass the unique attachment id as a parameter using the POST /api/v10/comments/reviews/{id} API endpoint.

You have now successfully added a comment with an attachment to a review.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID.	integer	path	Yes
body	The text content of the comment.	string	form	Yes
taskState	Optional task state of the comment. Valid values when adding a comment are comment (default) and open. This creates a plain comment or opens a task, respectively.  If the taskState parameter is not in the request, the task state is set to comment.	string	form	No

Parameter	Description	Type	Parameter Type	Required
notify	<p>Comment notifications can be delayed, see <a href="#">"Comment notification delay" on page 336</a>. The notify parameter enables you to set when the notification for the comment is sent.</p> <p>Valid options are:</p> <ul style="list-style-type: none"><li>■ <b>delayed</b> (default): notifications for this comment are delayed by the time set in <a href="#">"Comment notification delay" on page 580</a></li><li>■ <b>immediate</b>: the notification for this comment is sent immediately</li><li>■ <b>silent</b>: no notification is sent for this comment</li></ul>	string	query	No



Parameter	Description	Type	Parameter Type	Required
context	<p>Used to add the comment to a file in the review.</p> <p>Valid fields are:</p> <ul style="list-style-type: none"> <li>file mandatory unless attribute or comment are set: File to comment on. Valid only for <b>changes</b> and <b>reviews</b> topics. For example, <code>//depot/main/README.txt</code>.</li> <li>type: the filetype of the file, <b>text</b>, <b>binary</b>, <b>symlink</b>, <b>unicode</b>, <b>utf8</b>, <b>utf16</b>, <b>apple</b>, or <b>resource</b>. For more information on filetypes, see the <a href="#">Base filetypes</a> section of the <a href="#">P4 CLI Reference</a>.</li> <li>version optional, integer: With a <b>reviews</b> topic, this field specifies which version to attach the comment to.</li> <li>leftline optional, but if specified, you must also specify the rightline and content parameters. Integer: Left-side diff line number to attach the inline comment to. Valid only for <b>changes</b> and <b>reviews</b> topics.</li> </ul>	array	form	No

Parameter	Description	Type	Parameter Type	Required
	<ul style="list-style-type: none"> <li>rightline optional, but if specified, you must also specify the leftline and content parameters. Integer: Right-side diff line number to attach the inline comment to. Valid only for changes and reviews topics.</li> <li>content optional, but if specified, you must also specify the leftline and rightline parameters. Array of strings: Provide the content of the codeline the comment is on and the four preceding codelines. This is used to specify a short excerpt of context in case the lines being commented on change during the review. When set to null, P4 Code Review makes an effort to build the content on its own. Because this involves file operations, it might be slow.</li> </ul>			
attachments	Attachment id generated using the <a href="#">"Create an attachment" on page 1011</a> API endpoint.	integer	form	Yes

**Example usage**

Create a comment with attachment id 12, make the comment an open task, and delay sending the comment notification:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/comments/reviews/885"
```

The "mybodyfilename.txt" file contains the authenticated user's comment, the attachment id 12, and makes the comment an open task:

```
{
 "body": "This is another comment.",
 "taskState": "open"
 "context": {
 "file": "//depot/Jam/MAIN/src/jamgram.c",
 "type": "text",
 "version": 0,
 "leftLine": 13,
 "rightLine": 13,
 "content": [
 "#define yyerror (yyerrflag=0)",
 "#define YYRECOVERING (yyerrflag!=0)",
 "-/* cfront 1.2 defines \"c_plusplus\" instead of \"__cplusplus\" */",
 "+/* cfront 1.2 defines \"c_plusplus\" instead of \"__cplusplus\" */ change 1",
 "#ifdef c_plusplus"
]
 },
 "attachments": [12]
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 123,
 "topic": "reviews/12923",
 "context": {
 "file": "//depot/Jam/MAIN/src/jamgram.c",
 "version": 0,
 "leftLine": 13,
 "rightLine": 13,
 "content": [
 "#define yyerror (yyerrflag=0)",
 "#define YYRECOVERING (yyerrflag!=0)",
 "-/* cfront 1.2 defines \"c_plusplus\" instead of \"__cplusplus\" */",

```

```
"+/* cfront 1.2 defines \"c_plusplus\" instead of \"__cplusplus\" */ change 1",
"#ifdef c_plusplus"
]
}
"attachments": [
 12
],
"flags": [],
"taskState": "open",
"likes": [],
"user": "bruno",
"time": 1699003426,
"updated": 1699003426,
"edited": null,
"body": "This is another comment.",
"readBy": [],
"notify": "delayed"
}
],
"userDelayedNotificationCount": 2
}
}
```

#### If a request fails

<error code>:

- 400 one of the following:
  - the `idspecified` must be an integer greater than 0
  - invalid `context` field specified, valid fields are `file`, `leftline`, `rightline`, `content`, `version`, `attribute`, and `comment`
  - invalid `notify` value specified, value must be `delayed`, `immediate`, or `silent`
  - invalid `taskstate` value specified, value must be `open` or `comment`
- 403: you are not authorized to create this comment
- 404 the attachment does not exist
- 500 internal error, request failed

## Files: P4 Code Review files

### Get file data

#### Summary

Get file data for the specified file

GET /api/v10/files/{id}

### Description

Get file data for the specified file.

**Important:** You must be authenticated to view the file information.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	<p>File ID, this is the full filepath including the filename encoded to <b>URL safe Base64</b>.</p> <p>Various tools are available online to encode the id to <b>URL safe Base64</b>, for example: <a href="#">BASE64 Decode and Encode</a>.</p>	string	path	Yes
fileRevision	<p>Specify the revision of the file using one of the following options:</p> <ul style="list-style-type: none"><li>■ Specific file revision number: fileRevision=#nnn</li><li>■ File revision in a pending or submitted changelist: fileRevision=@=nnn</li><li>■ File revision in a submitted changelist only: fileRevision=@nnn</li></ul> <p>Where nnn is the revision or changelist number you want.</p>	string	query	No

## Example usage

### Get file data for a specific revision of a file

Get file data for `//depot/main/myfile.txt` revision 3. The full filepath and filename must be encoded in **URL safe Base64**.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/files/Ly9kZXBvdC9tYWluL215ZmlsZS50eHQ?fileRevision=#3"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "filename": "//depot/main/myfile.txt"
 "contentLink": "http://myswarm-url/view/depot/main/myfile.txt",
 "fileRevision": "#3",
 "contentType": "text/plain",
 "changelid": "12871"
 }
}
```

### If a request fails

`<error code>`:

- **400** Invalid file revision specified. Must be one of the following: `@` or `@=` followed by the changelist number, or `#` followed by the file revision number.
- **403** Insufficient permissions to access the file
- **404** File does not exist

HTTP/1.1 `<response error code>`

```
{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",
 "text": "<error message>"
 }],
 "data": null
}
```

## Get a file diff

### Summary

Get a file diff for the specified file

GET /api/v10/files/{id}/diff

### Description

Get a file diff for the specified file

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	<p>File ID, this is the full filepath including the filename for the diff encoded to <b>URL safe Base64</b>.</p> <p>Various tools are available online to encode the id to <b>URL safe Base64</b>, for example: <a href="#">BASE64 Decode and Encode</a>.</p>	string	path	Yes
from	<p>Specify the file revision to compare against in the form %23{revision-number}. This is the revision that would be displayed on the left side of a P4 Code Review diff.</p> <p>If the from parameter is not specified, the head revision is used.</p>	string	query	No

Parameter	Description	Type	Parameter Type	Required
to	Specify the file revision being compared in the form %23{revision-number}. This is the revision that would be displayed on the right side of a P4 Code Review diff.	string	query	Yes
type	<p>Specify the file type for the diff:</p> <ul style="list-style-type: none"><li>■ file: All file types except stream specs. This is the default if the parameter is unset.</li><li>■ stream: Stream spec files.</li></ul>	string	query	No
lines	Specify the number of lines of context around each diff returned in the response.	integer	query	No



Parameter	Description	Type	Parameter Type	Required
ignoreWs	<p>Specify the whitespace options for the diff:</p> <ul style="list-style-type: none"><li>■ Whitespace characters such as space, tab, and newline characters are returned in the response. This is the default if the parameter is unset.</li><li>■ 0: Whitespace characters such as space, tab, and newline characters are not returned in the response.</li><li>■ 1: Whitespace changes are not highlighted. Used for file types where whitespace changes are not important.</li></ul>	integer	query	No

Parameter	Description	Type	Parameter Type	Required
	<ul style="list-style-type: none"><li>2: Whitespace changes are highlighted. Used for file types where whitespace is important.</li></ul>			

---

Parameter	Description	Type	Parameter Type	Required
<code>max_size</code>	<p>P4 Code Review limits the size of files displayed in both reviews and standard file views. Files larger than 100 KB are paginated. How the files are paginated depends on the type of review being displayed.</p> <ul style="list-style-type: none"><li>■ When viewing added or deleted files, the entire file is treated as a single diff, with large files paginated according to the 100 KB limit. The <code>max_size</code> does not affect pagination.</li></ul>	integer	query	No

Parameter	Description	Type	Parameter Type	Required
	<ul style="list-style-type: none"><li>When viewing edited files, <code>max_size</code> is still relevant but works differently alongside pagination. This is because diffs can be small even though they are in a large file.</li></ul> <p>Specify the maximum sized file that can be returned as a diff for an edit:</p> <ul style="list-style-type: none"><li>-1: No limit. Allows files of any size to be returned.</li><li>0: Use the value defined by the <code>max_size</code> configuration setting, see <a href="#">"max_size" on page 613</a>. By default, this value is 1MB unless <code>max_size</code> is not explicitly set.</li></ul>			

---

Parameter	Description	Type	Parameter Type	Required
	<ul style="list-style-type: none"> <li>nnnn: where nnnn is a positive integer specifying the maximum file size in bytes.</li> </ul> <p>The lowest max_size value defaults to 50 bytes. If you set the max_size value lower than 50 bytes then P4 Code Review will ignore it.</p> <p>For information on known limitations with pagination, see <a href="#">"max_size" on page 613 in Files configuration</a>.</p> <p>When a file is in the edit state, depending on the max_size value, a fetchContent flag is set in the response. The fetchContent flag determines if the size of a file is larger than the max_size value and accordingly enables or disables the show more context buttons in the file diff panel. For more information about the show more context buttons, see <a href="#">"File diff panel" on page 436</a>.</p>			

Parameter	Description	Type	Parameter Type	Required
	Files larger than the max_size value sets the fetchContent flag to false. When the fetchContent flag is set to false, all the show more context buttons in the file diff panel are disabled. When the fetchContent flag is set to true, all the show more context buttons in the file diff panel are enabled.			

---

Parameter	Description	Type	Parameter Type	Required
max_diffs	<p>Specify the maximum number of diff sections (chunks) to return in the response:</p> <ul style="list-style-type: none"><li>▪ -1: Returns an unlimited number of diff sections (chunks) in the response.</li><li>▪ nnnn: where nnnn is the maximum number of diff sections (chunks) returned in the response.</li><li>▪ 0: Return the number of diffs specified by the max_diffs configuration setting in the response, see "<a href="#">max_diffs</a>" on <a href="#">page 595</a>. This is the default if the parameter is unset.</li></ul>	integer	query	No

Parameter	Description	Type	Parameter Type	Required
<code>offset</code>	<p>Specify the diff section (chunk) to start displaying from. The number of diff sections displayed from the <code>offset</code> parameter value is governed by the <code>max_diffs</code> parameter value specified.</p> <p>For example, if you just want to return the third diff section, set <code>offset=2</code> and <code>max_diffs=1</code>.</p>	integer	query	No
<code>fromFile</code>	<p>If the file has been moved or renamed, and edited, use the <code>fromFile</code> parameter to specify the original filepath including the filename encoded to <b>URL safe Base64</b>.</p> <p>Various tools are available online to encode the <code>fromFile</code> to <b>URL safe Base64</b>, for example: <a href="#">BASE64 Decode and Encode</a>.</p>	string	query	No

### Example usage

#### Get a file diff

Get the file diff between revisions 233 and 234 of `//projects/acme/main/src/aliquitaq/quosquo/do/53.txt` and don't return whitespace characters. The full filepath and filename must be encoded in **URL safe Base64**.



```
curl -u "username:ticket" "https://myswarm-url/api/v10
/files/Ly9wcm9qZWNOcy9hY21lL21haW4vc3JlL2FsaXFpdGFxL3F1b3NxdW8vZG8vNTMudHh0/diff
?from=%233&to=%234&ignoreWs=0&type=file"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "isCut": false,
 "isSame": false,
 "header": "--- a///projects/acme/main/src/aliqitaq/quosquo/do/53.txt#3\n+++
b///projects/acme/main/src/aliqitaq/quosquo/do/53.txt#4\n",
 "summary": {
 "adds": 1,
 "deletes": 0,
 "updates": 0
 },
 "diffs": [
 "@@ -30,10 +30,12 @@\n このリリースは、バージョン1.0.1g の OpenSSL ライブラリをリンクする
こと\n によって、Heartbleed CVE-2014-0160 脆弱性に対処しています。 \n \n -----
-----\n \n+Hello\n+\n 以前のリリースとの互換性 \n -----
-----\n \n 1. PERFORCE ブローカは、2007.2以降にリリースされたPERFORCE クライアント及び
\n サーバと互換性があります。 \n"
],
 "paging": {
 "diffs": 1,
 "offset": null
 }
 }
}
```

Get a file diff when the file size is greater than max\_size value

Get the file diff between revisions 232 and 40 of //jam/main/src/compile3.c and don't return whitespace characters. The full filepath and filename must be encoded in **URL safe Base64**. In the response code, the fetchContent flag is set to false as the file size is greater than the max\_size value.

```
curl -u "username:ticket" "https://myswarm-url/api/v10
/files/Ly9qYW0vbWFpbj9zcmMvY29tcGlzZTMuYw/diff?from=%232&to=%40%3D13444&ignoreWs=
0&type=file"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
```

```

"messages": [],
"data": {
 "isCut": 1000,
 "isSame": false,
 "header": "--- a///jam/main/src/compile3.c#2\n+++
b///jam/main/src/compile3.c@=13444\n",
 "summary": {
 "adds": 1,
 "deletes": 3,
 "updates": 3
 },
 "fetchContent": false,
 "diffs": [
 "@@ -23357,11 +23357,10 @@\n *tcompile_set() - compile the \"set variable\"
statement\n *tcompile_setcomp() - support for `rule` - save parse tree\n *tcompile_setexec
() - support for `actions` - save execution string\n *tcompile_settings() - compile the \"on =\"
(set variable on exec) statement\n *tcompile_switch() - compile 'switch' rule\n- * * External
routines:\n *\n *tcompile_foreach() - compile the \"for x in y\" statement\n *tcompile_if() -
compile 'if' rule\n *tcompile_include() - support for 'include' - call include() on file\n
*tcompile_local() - declare (and set) local variables\n",
 "@@ -24981,24 +24980,10 @@\n *tcompile_set() - compile the \"set variable\"
statement\n *tcompile_setcomp() - support for `rule` - save parse tree\n *tcompile_setexec
() - support for `actions` - save execution string\n *tcompile_settings() - compile the \"on =\"
(set variable on exec) statement\n *tcompile_switch() - compile 'switch' rule\n- * * External
routines:\n- *\n- *tcompile_foreach() - compile the \"for x \"
],
 "paging": {
 "diffs": 2,
 "offset": null
 }
}
}

```

Get a file diff when the file size is not greater than max\_size value

Get the file diff between revisions 232 and 40 of //jam/main/src/compile3.c and don't return whitespace characters. The full filepath and filename must be encoded in **URL safe Base64**. In the response code, the fetchContent flag is set to true as the file size is not greater than the max\_size value.

```

curl -u "username:ticket" "https://myswarm-url/api/v10
/files/Ly9qYW0vbWFPbi9zcmMvY29tcGlzZTMuYw/diff?from=%232&to=%40%3D13444&ignoreWs=
0&type=file"

```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],

```

```

"data": {
 "isCut": false,
 "isSame": false,
 "header": "--- a///jam/main/src/compile3.c#2\n+++
b///jam/main/src/compile3.c@=13444\n",
 "summary": {
 "adds": 1,
 "deletes": 3,
 "updates": 3
 },
 "fetchContent": true,
 "diffs": [
 "@@ -23357,11 +23357,10 @@\n *tcompile_set() - compile the \"set variable\"
statement\n *tcompile_setcomp() - support for `rule` - save parse tree\n *tcompile_setexec
() - support for `actions` - save execution string\n *tcompile_settings() - compile the \"on =\"
(set variable on exec) statement\n *tcompile_switch() - compile 'switch' rule\n- ** External
routines:\n\n *tcompile_foreach() - compile the \"for x in y\" statement\n *tcompile_if() -
compile 'if' rule\n *tcompile_include() - support for 'include' - call include() on file\n
*tcompile_local() - declare (and set) local variables\n",
 "@@ -24981,24 +24980,10 @@\n *tcompile_set() - compile the \"set variable\"
statement\n *tcompile_setcomp() - support for `rule` - save parse tree\n *tcompile_setexec
() - support for `actions` - save execution string\n *tcompile_settings() - compile the \"on =\"
(set variable on exec) statement\n *tcompile_switch() - compile 'switch' rule\n- ** External
routines:\n- *tcompile_foreach() - compile the \"for x in y\" statement\n- *tcompile_if() -
compile 'if' rule\n- *tcompile_include() - support for 'include' - call include() on file\n-
*tcompile_local() - declare (and set) local variables\n- *tcompile_null() - do nothing -- a stub
for parsing\n- *tcompile_rule() - compile a single user defined rule\n- *tcompile_rules() -
compile a chain of rules\n- *tcompile_set() - compile the \"set variable\" statement\n-
*tcompile_setcomp() - support for `rule` - save parse tree\n- *tcompile_setexec() - support for
`actions` - save execution string\n- *tcompile_settings() - compile the \"on =\" (set variable on
exec) statement\n- *tcompile_switch() - compile 'switch' rule\n */\n\n * Copyright 1993, 1995
Christopher Seiwald.\n ** External routines:\n\n",
 "@@ -36086,10 +36071,18 @@\n *tcompile_set() - compile the \"set variable\"
statement\n *tcompile_setcomp() - support for `rule` - save parse tree\n *tcompile_setexec
() - support for `actions` - save execution string\n *tcompile_settings() - compile the \"on =\"
(set variable on exec) statement\n *tcompile_switch() - compile 'switch' rule\n+ **\n+ **
Adding new block\n+ ** Adding new block\n+ ** Adding new block\n+ ** Adding new
block\n+ ** Adding new block\n+ ** Adding new block\n+ ** Adding new block\n+ **
External routines:\n\n *tcompile_foreach() - compile the \"for x in y\" statement\n
*tcompile_if() - compile 'if' rule\n *tcompile_include() - support for 'include' - call include() on
file\n",
 "@@ -57336,23 +57329,10 @@\n *tcompile_setexec() - support for `actions` - save
execution string\n *tcompile_settings() - compile the \"on =\" (set variable on exec)
statement\n *tcompile_switch() - compile 'switch' rule\n ** External routines:\n\n *tcompile_foreach() - compile the \"for x in y\" statement\n- *tcompile_if() - compile 'if' rule\n-
*tcompile_include() - support for 'include' - call include() on file\n- *tcompile_local() - declare
(and set) local variables\n- *tcompile_null() - do nothing -- a stub for parsing\n- *tcompile_rule
() - compile a single user defined rule\n- *tcompile_rules() - compile a chain of rules\n-

```

```

*\tcompile_set() - compile the \"set variable\" statement\n- *\tcompile_setcomp() - support for
`rule` - save parse tree \n- *\tcompile_setexec() - support for `actions` - save execution string
\n- *\tcompile_settings() - compile the \"on =\" (set variable on exec) statement\n- *\tcompile_
switch() - compile 'switch' rule\n- ** External routines:\n- *\n *\tcompile_foreach() - compile the
\"for x in y\" statement\n *\tcompile_if() - compile 'if' rule\n *\tcompile_include() - support for
'include' - call include() on file\n *\tcompile_local() - declare (and set) local variables\n",
 "@@ -99842,11 +99822,11 @@\n *\tcompile_setcomp() - support for `rule` - save parse
tree \n *\tcompile_setexec() - support for `actions` - save execution string \n *\tcompile_
settings() - compile the \"on =\" (set variable on exec) statement\n *\tcompile_switch() -
compile 'switch' rule\n- ** External routines:\n- *\n+ ****\n *\tcompile_foreach() - compile the
\"for x in y\" statement\n *\tcompile_if() - compile 'if' rule\n *\tcompile_include() - support for
'include' - call include() on file\n *\tcompile_local() - declare (and set) local variables\n
*\tcompile_null() - do nothing -- a stub for parsing\n",
 "@@ -99870,11 +99850,34 @@\n *\tcompile_setcomp() - support for `rule` - save parse
tree \n *\tcompile_setexec() - support for `actions` - save execution string \n *\tcompile_
settings() - compile the \"on =\" (set variable on exec) statement\n *\tcompile_switch() -
compile 'switch' rule\n- ** External routines:\n- *\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+
****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+ ****\n+
****\n+ ****\n+ ****\n+ ****\n *\tcompile_foreach() - compile the \"for x in y\" statement\n
*\tcompile_if() - compile 'if' rule\n *\tcompile_include() - support for 'include' - call include() on
file\n *\tcompile_local() - declare (and set) local variables\n *\tcompile_null() - do nothing -- a
stub for parsing\n",
 "@@ -99968,11 +99971,11 @@\n *\tcompile_setcomp() - support for `rule` - save parse
tree \n *\tcompile_setexec() - support for `actions` - save execution string \n *\tcompile_
settings() - compile the \"on =\" (set variable on exec) statement\n *\tcompile_switch() -
compile 'switch' rule\n- ** External routines:\n- *\n+ ****\n *\tcompile_foreach() - compile the
\"for x in y\" statement\n *\tcompile_if() - compile 'if' rule\n *\tcompile_include() - support for
'include' - call include() on file\n *\tcompile_local() - declare (and set) local variables\n
*\tcompile_null() - do nothing -- a stub for parsing\n"
],
 "paging": {
 "diffs": 7,
 "offset": null
 }
}
}

```

#### Get a file diffs where each individual file diff size is more than 100kb

This endpoint retrieves a paginated portion of a file diff when the total diff exceeds 100kb (this limit is a hard coded pagination limit).

When a file diff exceeds this size, the endpoint response includes a section (or "chunk") of the diff along with metadata and pagination information to request subsequent chunks.

```

curl --location --request GET 'http://my-swarmurl/api/v10
/files/Ly9qYW0vbWFPbi9zcmMvZmIsZW50LmM/diff?from=%231&to=%40%3D12343&ignoreWs=0&
type=file' \
--header 'Authorization: Basic YnJ1bm86UGFzc3cwcmQ='

```

**Tip:** These examples are truncated due to the size of the text strings.

If the file has been added or deleted, the response looks like this:

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "isCut": false,
 "isSame": false,
 "isContentChunk": 1000,
 "contentChunkStart": 0,
 "contentChunkEnd": 1000,
 "contentChunkStartLine": 1,
 "contentChunkEndLine": 33,
 "wholeDiffSize": 27582,
 "currentDiffChunkSize": 1000,
 "header": "--- a/dev/null\n+++ b///depot/Jam/MAIN/src/compile_big.c@=13126\n",
 "summary": {
 "adds": 33,
 "deletes": 0,
 "updates": 0
 },
 "diffs": [
 "@@ -1,0 +1,33 @@\n+/*\n+ * Copyright 1993, 1995 Christopher Seiwald.\n+ *\n+ * This file is part of Jam - see jam.c for Copyright information.\n+ */\n+# include\n+ \"jam.h\"\n+# include \"lists.h\"\n+# include \"parse.h\"\n+# include \"compile.h\"\n+# include \"variable.h\"\n+# include \"rules.h\"\n+# include \"newstr.h\"\n+# include\n+ \"make.h\"\n+# include \"search.h\"\n+\n+/*\n+ * compile.c - compile parsed jam statements\n+ *\n+ * External routines:\n+ *\n+ * \tcompile_foreach() - compile the 'for x in y' statement\n+ *\n+ * \tcompile_if() - compile 'if' rule\n+ *\n+ * \tcompile_include() - support for 'include' - call include() on\n+ file\n+ *\n+ * \tcompile_local() - declare (and set) local variables\n+ *\n+ * \tcompile_null() - do nothing -- a stub for parsing\n+ *\n+ * \tcompile_rule() - compile a single user defined rule\n+ *\n+ * \tcompile_rules() - compile a chain of rules\n+ *\n+ * \tcompile_set() - compile the 'set variable' statement\n+ *\n+ * \tcompile_setcomp() - support for `rule` - save parse tree\n+ *\n+ * \tcompile_setexec() - support for `actions` - save execution string\n+ *\n+ * \tcompile_settings() - compile the 'on =' (set variabl",
],
 "paging": {
 "diffs": 1,
 "offset": null
 }
 }
}
```

If the file has been edited, the response looks like this:



### If a request fails

- **400** Invalid file revision specified.
- **404** File does not exist or you do not have permission to view it

```
{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",
 "text": "<error message>"
 }],
}
```

```
"data" : null
}
```

## File edit

This API endpoint only supports text files. File edit is a Technology Preview feature.

Features offered in Technology Preview are experimental and not guaranteed to always work as expected. If you have feedback and functionality suggestions, email [techpreview@perforce.com](mailto:techpreview@perforce.com).

### Summary

Edit the file content and submit or shelve the file

PUT /api/v10/files/{id}

### Description

Edit the file content and submit or shelve the file.

### Requirements:

- You must have permissions to edit the file.
- If the file version changes while you are editing it or if you are not editing the Head revision of the file, committing your file changes will overwrite the current head revision of the file.

If you are unsure, shelving your change is a safer option because it will not overwrite the head revision of the file and you can do a manual resolve when you commit the shelf.

- File edit is enabled by default but can be disabled by your P4 Code Review administrator, see ["allow\\_edits" on page 615](#).



## Parameters

Parameter	Description	Type	Parameter Type	Required
id	<p>File ID, this is the full filepath including the filename encoded to <b>URL safe Base64</b>.</p> <p>Various tools are available online to encode the id to <b>URL safe Base64</b>, for example: <a href="#">BASE64 Decode and Encode</a></p>	string	path	Yes
content	<p>Specify the entire file content to be shelved or submitted. Lines in the content must be separated with backslash and n characters \n.</p>	string	body	Yes

Parameter	Description	Type	Parameter Type	Required
description	<p>Specify a changelist description.</p> <p><b>Optional:</b> you can create a review for your change or add it to an existing review by including a review keyword in the <b>Change description</b>. For instruction on creating a review and adding a changelist to a review using review keywords, see <a href="#">"Create a review"</a> on page 669 and <a href="#">"Add a changelist to a review"</a> on page 670.</p>	string	body	No
action	<p>Specify the file action as either <b>shelve</b> or <b>submit</b>.</p>	string	body	Yes

### Example usage

#### Shelve file with edited content

Shelve `//depot/main/myfile.txt` file with edited content. The full filepath and filename must be encoded in **URL safe Base64**.

```
curl -X PUT -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://myswarm-url/api/v10/files/Ly9kZXBvdC9tYWluL215ZmlsZS50eHQ"
```

The "mybodyfilename.txt" file contains the file content, description and the shelve action:

```
{
 "content": "My original line\nMy nice new line\nAnother original line"
 "description": "Added a new line."
 "action": "shelve"
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data" : {
 "filename": "//depot/main/myfile.txt"
 "contentLink": "http://myswarm-url/view/depot/main/myfile.txt",
 "fileRevision": "@=12896",
 "contentType": "text/plain",
 "changelid": "12896"
 }
}
```

#### If a request fails

<error code>:

- **400** Invalid file revision specified. Must be one of the following: @ or @= followed by the changelist number, or # followed by the file revision number.
- **403** Insufficient permissions to submit or shelve the file
- **404** File does not exist
- **500** Submit command failed because a newer revision of the file exists, you must resolve and submit your changes or revert your changes.

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

# Groups: P4 Code Review groups

## Get a list of groups

### Summary

Get a list of groups.

GET /api/v10/groups

### Description

Get a list of groups.

### Parameters

Parameter	Description	Type	Parameter Type	Required
[ids]	An optional array of group ids to return in the response. Omitting this parameter or passing an empty value shows all groups.	string	query	No
[fields]	An optional comma-separated list (or array) of fields to show for each group. Omitting this parameter or passing an empty value shows all fields.	string	query	No

Parameter	Description	Type	Parameter Type	Required
[expand]	<p>An optional parameter to specify whether details of any subgroups in a group are returned in the response. Omitting the <code>expand</code> parameter or passing an empty value will not return subgroup details.</p> <ul style="list-style-type: none"><li>■ <code>true</code>: Returns details of any subgroups for each group returned. This is done recursively, so if a subgroup has a subgroup, its details are also returned.</li><li>■ <code>false</code>: Details of any subgroups in the groups are not returned.</li></ul>	boolean	query	No

### Example usage

#### Get a list of all groups

```
curl -u "username:ticket" "https://myswarm-url/api/v10/groups"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "groups": [
 {
 "id": "Mercury",
 "description": "Group for project Mercury that includes Development, QA, and Docs",
 "maxResults": null,
 "maxScanRows": null,
 "maxLockTime": null,
 "maxOpenFiles": "unset",
 "timeout": 43200,
 "passwordTimeout": null,
 "ldapConfig": null,
 "ldapSearchQuery": null,
 "ldapUserAttribute": null,
 "subgroups": [
 "mercury-dev",
 "mercury-docs",
 "mercury-qa"
],
 "owners": [
 "allison.clayborne",
 "jack.boone"
],
 "users": [
 "allison.clayborne",
 "jack.boone",
 "claire.brevia",
 "alex.randolph"
],
 "name": "Mercury",
 "emailFlags": [],
 "group_notification_settings": null,
 "emailAddress": null,
 "useMailingList": "mercury.group@nowhere.xxjzzj.com"
 },
 {
 ...
 }
]
 }
}
```

```
}
}
```

#### Limiting returned fields

To limit the returned fields when fetching groups:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/groups?fields=id,owners,users"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "groups": [
 {
 "id": "Mercury",
 },
 "owners": [
 "allison.clayborne",
 "jack.boone"
],
 "users": [
 "allison.clayborne",
 "jack.boone",
 "claire.brevia",
 "alex.randolph"
],
 },
 {
 ...
 <Other group ids formatted as above>
 ...
 }
]
}
```

#### Limiting groups returned and expanding subgroup details

Return the id, name, and subgroups fields for the Mercury group and return subgroup details:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/groups?fields=id,name,subgroups&ids=Mercury&expand=true"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "groups": [
 {
 "id": "Mercury",
 "name": "Mercury",
 "subgroups": [
 "mercury-dev",
 "mercury-docs",
 "mercury-qa"
]
 },
 {
 "id": "mercury-dev",
 "name": "mercury-dev",
 "subgroups": []
 },
 {
 "id": "mercury-docs",
 "name": "mercury-docs",
 "subgroups": []
 },
 {
 "id": "mercury-qa",
 "name": "mercury-qa",
 "subgroups": []
 }
]
 }
}
```

## Jobs: P4 Code Review Perforce jobs

### Get a list of Perforce jobs

#### Summary

Get a list of Perforce jobs.

GET /api/v10/jobs

#### Description

Get a list of Perforce jobs.



**Parameters**

Parameter	Description	Type	Parameter Type	Required
filter	Filter Perforce jobs by using a combination of a job field and value. The job fields and values available depend on your Perforce job configuration.	string	query	No
max	Maximum number of Perforce jobs to return. This does not guarantee that max jobs are returned. It does guarantee that the number of jobs returned won't exceed max. If not specified, the 50 most recent jobs are returned.	integer	query	No

**Example usage****Get all Perforce jobs**

```
curl -u "username:ticket" "https://myswarm-url/api/v10/jobs"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
```

```

{
 "job": "job001002",
 "link": "/jobs/job001002",
 "fixStatus": "open",
 "description": "The description is blank when not specified for schedules in release 2.0.
Please resolve.\n",
 "descriptionMarkdown": "The description is blank when not
specified for schedules in release 2.0.
\n
 Please resolve."
},
{
 ...
 <Other job ids formatted as above>
 ...
}
]
}
}

```

Get Perforce job id job000050

```
curl -u "username:ticket" "https://myswarm-url/api/v10/jobs?filter=id=job000050"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job000050",
 "link": "/jobs/job000050",
 "fixStatus": "fixed",
 "description": "Update API call for status. Use the new repsonse style.\n",
 "descriptionMarkdown": "Update API call for
status.
\n Use the new repsonse
style."
 }
]
 }
}

```

Get the 10 most recent Perforce jobs with a status of "fixed"

```
curl -u "username:ticket" "https://myswarm-url/api/v10/jobs?filter=status=fixed&max=10"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job000999",
 "link": "/jobs/job000999",
 "fixStatus": "fixed",
 "description": "Copyright text is incomplete. Please resolve.\n",
 "descriptionMarkdown": "Copyright text is
incomplete.
\n Please resolve."
 },
 {
 ...
 <Other job ids formatted as above>
 ...
 }
]
 }
}
```

#### If a request fails

<error code>:

**500** Invalid field name specified, the error text contains the invalid field name.

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Login: P4 Code Review login

### Get P4 Code Review user session details

#### Summary

Get P4 Code Review session details of a user.

GET /api/v10/session

### Description

Get the P4 Code Review session details of a user.

### Example usage

Get the session details for bruno:

```
curl -u "bruno:<ticket>" https://my-swarm-host/api/v10/session
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [{
 "code": "user-login-successful",
 "text": "User logged in."
 }],
 "data": {
 "user": {
 "User": "bruno",
 "Type": "standard",
 "Email": "bruno@swarm.local",
 "FullName": "bruno",
 "isAdmin": true,
 "isSuper": false
 }
 }
}
```

### If a request fails

<error code>:

- 401 Incorrect or missing credentials
- 403 Unauthorized

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
}
```

```
"data" : null
}
```

## Create a new P4 Code Review user session

### Summary

Create a new P4 Code Review user session.

POST /api/v10/session

### Description

Create a new P4 Code Review user session.

When creating a new P4 Code Review user session with a valid ["sso" on page 181](#) user, you must call ["Get SSO authentication URL" on page 1113](#) API endpoint to get the navigation URL. This navigation URL must be opened in a web browser to login the user. Once the user is logged in, a login successful message for this user is displayed in the ["Create a new P4 Code Review user session "](#) above API endpoint.

## Parameters

Parameter	Description	Type	Parameter Type	Required
method	<p>Login service type identifier. The default value is basic.</p> <ul style="list-style-type: none"><li>■ basic: uses basic authentication to login to P4 Code Review</li><li>■ sso: uses "sso" on page 181 configurable as follows:<ul style="list-style-type: none"><li>■ enabled all users must use P4 AS to log in to P4 Code Review.</li></ul></li></ul> <p>The login page displays an input box to enter the email or username. Once the input box is filled, the <b>Log in with SSO</b> button is activated. See "Log in with SSO" on page 384</p>	string	body	No

Parameter	Description	Type	Parameter Type	Required
	<ul style="list-style-type: none"><li>■ optional P4 AS is available for users to log in to P4 Code Review but is not enforced.  The login page displays an input box to enter the email or username. Once the input box is filled, the <b>Log in with SSO</b> button is enabled. To log in with SSO, see <a href="#">"Log in with SSO" on page 384</a>.</li></ul>			

---



Parameter	Description	Type	Parameter Type	Required
	<p>The user can also manually log in to P4 Code Review using the <b>Log in with credentials</b> button. To log in manually, see "<a href="#">Log in with a password</a>" on <a href="#">page 383</a>.</p> <p>If you switch to <b>Log in with credentials</b> and then decide to use single sign-on, click <b>Log in with SSO</b>.</p>			

Parameter	Description	Type	Parameter Type	Required
	<div><div>■ disabled</div><p>P4 AS is not available to P4 Code Review. This is the default value.</p><p>The login page displays two input boxes to enter the email or username, and to enter the password. Once the input boxes are filled, the <b>Log in with credentials</b> button is activated. See "<a href="#">Log in with a password</a>" on <a href="#">page 383</a>.</p></div>			
username	Username	string	body	Yes

Parameter	Description	Type	Parameter Type	Required
password	User ticket (recommended) or password  <div> <b>Important:</b>            If your P4 Server is set to security level 3 or higher, you must use a ticket value for the password parameter.         </div>	string	body	Yes
remember	<ul style="list-style-type: none"> <li>■ true: User stays logged in between browser restarts.</li> <li>■ false: User does not stay logged in between browser restarts. This is the default if the remember parameter is not in the body or if it does not have a value.</li> </ul>	Boolean	body	No

### Example usage

Create a new P4 Code Review session for bruno using the basic method

```
curl -X POST -H "Content-Type: application/json" -d "@mybodyfilename.txt" "https://my-swarm-host/api/v10/session"
```

The "mybodyfilename.txt" file contains the login details for bruno:

```
{
 "method": "basic",
 "username": "bruno",
 "password": "A1AFB97F0F218DF7B122F229C7DECA46",
 "remember": false
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [
 {
 "code": "user-login-successful",
 "text": "User logged in."
 }
],
 "data": {
 "user": {
 "User": "bruno",
 "Type": "standard",
 "Email": "bruno@swarm.local",
 "FullName": "bruno",
 "isAdmin": true,
 "isSuper": false
 }
 }
}
```

Create a new P4 Code Review session for bruno using the sso method:

```
curl -X POST -H "Content-Type: application/json" -d "@mybodyfilename.txt" "https://my-swarm-host/api/v10/session"
```

The "mybodyfilename.txt" file contains the login details for bruno:

```
{
 "method": "sso",
 "username": "bruno",
 "remember": false
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
```

```

"error": null,
"messages": [
 {
 "code": "user-login-successful",
 "text": "User logged in."
 }
],
"data": {
 "user": {
 "User": "bruno",
 "Type": "standard",
 "Email": "bruno@swarm.local",
 "FullName": "bruno",
 "isAdmin": true,
 "isSuper": false
 }
}
}

```

#### If a request fails

<error code>:

- 401 Incorrect or missing credentials
- 403 Unauthorized

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Get SSO authentication URL

### Summary

Get P4 Code Review SSO authentication URL.

GET /api/v10/get-sso-auth-url

### Description

Get P4 Code Review SSO authentication URL.

### Example usage

Get P4 Code Review SSO authentication URL for bruno.

```
curl -u "bruno:<ticket>" https://my-swarm-host/api/v10/get-sso-auth-url
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "data": {
 "url":
 "https://10.153.120.29:3000/oidc/login/01HR6QMK0MPC5J506C2WG5ZZ0R?instanceId=none",
 "isValid": true,
 "errors": null
 },
 "messages": null,
 "error": null
}
```

### If a request fails

<error code>:

- 401 Incorrect or missing credentials
- 403 Unauthorized

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Login to P4 Code Review with SAML

### Important:

P4 Code Review now supports P4 AS (HAS) as a Single Sign-On (SSO) provider. This helps to simplify configuration and create a more robust SSO solution.

### Summary

Login to P4 Code Review with SAML

POST /api/v10/saml/login

### Description

Login to P4 Code Review with SAML.

### Parameters

Parameter	Description	Type	Parameter Type	Required
method	Login service type identifier is set to use <a href="#">saml</a> configuration to login to P4 Code Review.	string	body	No
redirect	<p>Options are:</p> <ul style="list-style-type: none"><li>■ true or not specified: P4 Code Review redirects the user to the HTTP_REFERER url or to the specified custom <a href="#">"logout_url"</a> on <a href="#">page 605</a> if it has been set.</li><li>■ false: P4 Code Review does not redirect the user.</li></ul>	string	query	No

## Example usage

### Login in to P4 Code Review with SAML

Login in to P4 Code Review with SAML.

```
curl -X POST -u "super:<ticket>" "https://myswarm-url/api/v10/saml/login"
```

On successful login P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "isValid": "true",
 "url": "<url to redirect to>"
 }
}
```

### Login in to P4 Code Review with SAML and redirect=false

Login in to P4 Code Review with SAML and redirect=false

```
curl -X POST -u "super:<ticket>" "https://myswarm-url/api/v10/saml/login?redirect=false"
```

On successful login P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "isValid": "true"
 }
}
```

### If a request fails

<error code>:

400 Error occurred with the SAML login

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",

```



```
"text" : "<error message>"
 },
 "data" : null
}
```

## Logout of a P4 Code Review user session

### Summary

Logout of a P4 Code Review user session.

DELETE /api/v10/session

### Description

Logout of a P4 Code Review user session.

### Example usage

Logout bruno of a P4 Code Review user session.

```
curl -X DELETE -u "bruno:<ticket>" https://my-swarm-host/api/v10/session
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [
 {
 "code": "user-logged-out",
 "text": "Successful Logout."
 }
],
 "data": {
 "url": "/"
 }
}
```

### If a request fails

<error code>:

- 401 Incorrect or missing credentials
- 403 Unauthorized

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Menus: P4 Code Review main menu items

### Get main menu details

#### Summary

Get details of the P4 Code Review main menu items.

GET /api/v10/menus

#### Description

Get details of the P4 Code Review main menu items including [custom menu items](#).

#### Note:

Any users can get the P4 Code Review main menu details, the user does not need to be authenticated.

Parameter	Description	Type	Parameter Type	Required
project	Specify a project in the request to return the menu items for the project.	string	query	No

#### Example usage

##### Fetch P4 Code Review menu items

Get details of the P4 Code Review main menu items:

```
curl "https://myswarm-url/api/v10/menus"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "menu": [
 {
 "id": "dashboard",
 "enabled": true,
 "target": "/",
 "cssClass": "component",
 "title": "dashboard",
 "priority": 100,
 "roles": null
 },
 {
 "id": "activity",
 "enabled": true,
 "target": "/activity/",
 "cssClass": "activity",
 "title": "activity",
 "priority": 120,
 "roles": null
 },
 {
 "id": "reviews",
 "enabled": true,
 "target": "/reviews/",
 "cssClass": "component",
 "title": "reviews",
 "priority": 130,
 "roles": null
 },
 {
 "id": "projects",
 "enabled": true,
 "target": "/projects/",
 "cssClass": "projects",
 "title": "projects",
 "priority": 140,
 "roles": null
 },
 {
 "id": "files",
 "enabled": true,
 "target": "/files/",
 "cssClass": "files",
```

```
"title": "files",
"priority": 150,
"roles": null
},
{
 "id": "changes",
 "enabled": true,
 "target": "/changes/",
 "cssClass": "changes",
 "title": "changes",
 "priority": 160,
 "roles": null
},
{
 "id": "jobs",
 "enabled": true,
 "target": "/jobs/",
 "cssClass": "jobs",
 "title": "jobs",
 "priority": 170,
 "roles": null
},
{
 "id": "groups",
 "enabled": true,
 "target": "/groups/",
 "cssClass": "groups",
 "title": "groups",
 "priority": 180,
 "roles": null
},
{
 "id": "workflows",
 "enabled": true,
 "target": "/workflows/",
 "cssClass": "workflows",
 "title": "workflows",
 "priority": 190,
 "roles": null
},
{
 "id": "tests",
 "enabled": true,
 "target": "/testdefinitions/",
 "cssClass": "tests",
 "title": "tests",
 "priority": 200,
 "roles": null
}
```

```

 },
 {
 "id": "jenkins_blue",
 "enabled": true,
 "target": "http://my-jenkins.instance.blue.com",
 "cssClass": "custom_menu",
 "title": "Jenkins Build",
 "priority": 210,
 "roles": "authenticated"
 }
],
 "contextMetadata": []
}
}

```

#### Fetch menu items for a project (authenticated and project owner)

Get menu items for project Jam when the authenticated user owns the project:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/menus?project=jam"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "menu": [
 {
 "id": "activity",
 "enabled": true,
 "target": "/projects/jam/activity/",
 "cssClass": "activity",
 "title": "activity",
 "priority": 120,
 "roles": null
 },
 {
 "id": "reviews",
 "enabled": true,
 "target": "/projects/jam/reviews/",
 "cssClass": "component",
 "title": "reviews",
 "priority": 130,
 "roles": null
 },
 {
 "id": "files",

```

```
 "enabled": true,
 "target": "/projects/jam/files/",
 "cssClass": "files",
 "title": "files",
 "priority": 150,
 "roles": null
 },
 {
 "id": "changes",
 "enabled": true,
 "target": "/projects/jam/changes/",
 "cssClass": "changes",
 "title": "changes",
 "priority": 160,
 "roles": null
 },
 {
 "id": "settings",
 "enabled": true,
 "target": "/projects/jam/settings/",
 "cssClass": "settings",
 "title": "settings",
 "priority": 200,
 "roles": null
 }
],
"contextMetadata": {
 "contextName": "jam"
}
}
```

**Fetch menu items for a project (unauthenticated or not project owner)**

Get menu items for project Jam without authentication or when the authenticated user is not a project owner:

```
curl "https://myswarm-url/api/v10/menus?project=jam"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "menu": [
 {
 "id": "activity",
```

```

 "enabled": true,
 "target": "/projects/jam/activity/",
 "cssClass": "activity",
 "title": "activity",
 "priority": 120,
 "roles": null
 },
 {
 "id": "reviews",
 "enabled": true,
 "target": "/projects/jam/reviews/",
 "cssClass": "component",
 "title": "reviews",
 "priority": 130,
 "roles": null
 },
 {
 "id": "files",
 "enabled": true,
 "target": "/projects/jam/files/",
 "cssClass": "files",
 "title": "files",
 "priority": 150,
 "roles": null
 },
 {
 "id": "changes",
 "enabled": true,
 "target": "/projects/jam/changes/",
 "cssClass": "changes",
 "title": "changes",
 "priority": 160,
 "roles": null
 }
],
"contextMetadata": {
 "contextName": "Jam"
}
}
}

```

**If a request fails****<error code>:**

- 404 Project specified does not exist

HTTP/1.1 &lt;response error code&gt;

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Projects : P4 Code Review Projects

### Get List of Projects

#### Summary

Get List of Projects and metadata

GET /api/v10/projects

#### Description

Returns a list of projects in P4 Code Review that are visible to the current user.

#### Note:

The branch paths field is always returned empty for all project branches returned with the project list endpoint. This is because calculating the paths for all of the project branches is an expensive operation when there are a lot of projects and paths. To check the branch paths for a specific project, use the ["Get Project Information" on page 1128](#) endpoint.

#### Tip:

- tests and deploy fields:
  - **If a project has an owner:** only project owners and users with *super* user permissions can view the tests and deploy fields when fetching projects.
  - **If a project does not have an owner:** only project members and users with *super* user permissions can view the tests and deploy fields when fetching projects.
- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.



## Parameters

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list or array of top level fields to return for the projects. Omitting this parameter or passing an empty value shows all fields.	string	query	No

## Example usage

### Fetch a list of projects

To fetch all projects:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/projects"
```

Pagination is not currently supported by this endpoint. P4 Code Review responds with a list of projects:

```
{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "id": "jam",
 "branches": [
 {
 "id": "main",
 "name": "Main",
 "workflow": null,
 "paths": [],
 "defaults": {
 "reviewers": []
 },
 "minimumUpVotes": null,
 "retainDefaultReviewers": false,
 "moderators": [],
 "moderators-groups": []
 }
]
 }
]
 }
}
```

```
 },
 ...
 ...
],
 "defaults": {
 "reviewers": []
 },
 "deleted": false,
 "deploy": {
 "enabled": false,
 "url": null
 },
 "description": "This is the project Jam description",
 "emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
 },
 "jobview": null,
 "members": [
 "Bruno",
 "raj"
],
 "minimumUpVotes": null,
 "name": "Jam",
 "owners": [],
 "private": true,
 "retainDefaultReviewers": false,
 "subgroups": [],
 "tests": {
 "enabled": false,
 "url": null,
 "postBody": null,
 "postFormat": "url"
 },
 "workflow": null,
},
{
 ...
 <other projects formatted as above>
 ...
},
],
}
}
```

**Fetch specified fields for all projects**

To fetch the id, description, and members fields for all projects, use one of the following methods:

- **Using a comma-separated list of top level fields:**

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/projects?fields=id,description,members"
```

- **Using an array of top level fields:**

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/projects?fields[]=id&fields[]=description&fields[]=members"
```

Pagination is not currently supported by this endpoint. P4 Code Review responds with a list of all projects:

```
{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "id": "blue-book",
 "description": "This is a private project for use only by users with sufficient clearance.",
 "members": [
 "alex.randolph",
 "allison.clayborne"
],
 },
 {
 "id": "jam",
 "description": "This the project Jam description.",
 "members": [
 "Bruno",
 "raj"
],
 },
 {
 "id": "jplugin",
 "description": "A Java plugin for continuous integration.",
 "members": [
 "allison.clayborne",
 "jack.boone",
 "steve.russell"
],
 },
],
 },
}
```

**If a request fails**

<error code>:

400 metadata request invalid, boolean required

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get Project Information

### Summary

Get Project Information

GET /api/v10/projects/{id}

### Description

Retrieve all information about a project or specify which fields you want to return for the project.

#### Tip:

- tests and deploy fields:
  - **If a project has an owner:** only project owners and users with *super* user permissions can view the tests and deploy fields when fetching projects.
  - **If a project does not have an owner:** only project members and users with *super* user permissions can view the tests and deploy fields when fetching projects.
- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

## Parameters

Parameter	Description	Type	Parameter Type	Required
id	Project ID	string	path	Yes
fields	An optional comma-separated list or array of top level fields to return for the project. Omitting this parameter or passing an empty value shows all fields.	string	query	No

## Example usage

### Fetch all fields for a project

To fetch all fields for project jam:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/projects/jam"
```

P4 Code Review responds with a project entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "id": "jam",
 "branches": [
 {
 "id": "main",
 "name": "Main",
 "workflow": null,
 "paths": [
 "//depot/Jam/MAIN/...",
 "//jam/main/..."
],
 "defaults": {
```

```
 "reviewers": [],
 },
 "minimumUpVotes": null,
 "retainDefaultReviewers": false,
 "moderators": [],
 "moderators-groups": []
},
...
...
],
"defaults": {
 "reviewers": []
},
"deleted": false,
"deploy": {
 "enabled": false,
 "url": ""
},
"description": "This the project Jam description",
"emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
},
"jobview": "",
"members": [
 "Bruno",
 "raj"
],
"minimumUpVotes": null,
"name": "Jam",
"owners": [],
"private": false,
"retainDefaultReviewers": false,
"subgroups": [],
"tests": {
 "enabled": false,
 "url": "",
 "postBody": "",
 "postFormat": "url"
},
"workflow": null
},
],
}
}
```

**Fetch specified fields for a project**

To fetch the id, description, and members fields for project Jam, use one of the following methods:

- **Using a comma-separated list of top level fields:**

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/projects/jam?fields=id,description,members"
```

- **Using an array of top level fields:**

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/projects/jam?fields[]=id&fields[]=description&fields[]=members"
```

P4 Code Review responds with a project entity containing the requested fields:

```
{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "id": "jam",
 "description": "This the project Jam description",
 "members": [
 "Bruno",
 "raj"
],
 },
],
 },
}
```

#### If a request fails

<error code>:

404 Project does not exist or you do not have permission to view it

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get a list of Perforce jobs on a project

### Summary

Get a list of Perforce jobs on a project.

GET /api/v10/projects/{project\_id}/jobs

### Description

Get a list of Perforce jobs on a project.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>filter</code>	<p>Filter Perforce jobs by using a combination of a job field and value. The job fields and values available depend on your Perforce job configuration.</p> <p>If you have a job filter selected for the project you can also limit jobs returned by field name and value, see Job filter in <a href="#">Project settings</a>.</p>	string	query	No

---



Parameter	Description	Type	Parameter Type	Required
max	Maximum number of Perforce jobs to return. This does not guarantee that max jobs are returned. It does guarantee that the number of jobs returned won't exceed max. If not specified, the 50 most recent jobs are returned.	integer	query	No

### Example usage

Get all Perforce jobs on the Acme project

```
curl -u "username:ticket" "https://myswarm-url/api/v10/projects/acme/jobs"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job001002",
 "link": "/jobs/job001002",
 "fixStatus": "open",
 "subsystem": "backend",
 "description": "The description is blank when not specified for schedules in release 2.0.
Please resolve.\n",
 "descriptionMarkdown": "The description is blank when not
specified for schedules in release 2.0.
\n
 Please resolve."
 },
 {
```

```
...
 <Other job ids formatted as above>
 ...
}
]
}
}
```

**Get the 10 most recent Perforce jobs in project Acme**

```
curl -u "username:ticket" "https://myswarm-url/api/v10/projects/acme/jobs?max=10"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job000999",
 "link": "/jobs/job000999",
 "fixStatus": "fixed",
 "subsystem": "docs",
 "description": "Copyright text is incomplete. Please resolve.\n",
 "descriptionMarkdown": "Copyright text is
incomplete.
\n Please resolve."
 },
 {
 ...
 <Other job ids formatted as above>
 ...
 }
]
 }
}
```

**Get Perforce jobs in the range of job000050 to job000059 that are part of project Acme**

```
curl -u "username:ticket" "https://myswarm-url/api/v10/projects/acme/jobs?filter=job00005*"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
```

```

"messages": [],
"data": {
 "jobs": [
 {
 "job": "job000050",
 "link": "/jobs/job000050",
 "fixStatus": "fixed",
 "subsystem": "frontend",
 "description": "Update API call for status. Use the new repsonse style.\n",
 "descriptionMarkdown": "Update API call for
status.
\n Use the new repsonse
style."
 }
]
}
}

```

Get all Perforce jobs with a Job filter matching subsystem=frontend for project Acme

```
curl -u "username:ticket" "https://myswarm-
url/api/v10/projects/acme/jobs?filter=subsystem=frontend"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "jobs": [
 {
 "job": "job000050",
 "link": "/jobs/job000050",
 "fixStatus": "fixed",
 "subsystem": "frontend",
 "description": "Update API call for status. Use the new repsonse style.\n",
 "descriptionMarkdown": "Update API call for
status.
\n Use the new repsonse
style."
 }
]
 }
}

```

If a request fails

<error code>:

- 404 Project does not exist or you do not have permission to view it
- 500 Invalid field name specified, the error text contains the invalid field name.

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Create a new project

### Summary

Create a new P4 Code Review project.

POST /api/v10/projects

### Description

Create a new P4 Code Review project.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Project ID	string	path	Yes

### Example usage

Create project Saturn:

```
curl -X POST -u "username:ticket" -H "Content-Type: application/json" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/projects"
```

The "mybodyfilename.txt" file contains details for project Saturn:

```
{
 "id": "saturn",
 "name": "Saturn",
 "description": "Project Saturn is our moonshot project.",
}
```

```

"jobView": "project=swarm",
"members": ["bruno", "alice", "bob"],
"defaults": {
 "reviewers": ["bob", "alice"]
},
"tests": {
 "enabled": false
},
"branches": [
 {
 "id": "main",
 "name": "main",
 "paths": ["//depot/saturn/..."],
 "defaults": {
 "reviewers": ["bob"]
 }
 }
]
}

```

P4 Code Review responds with:

```

{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "id": "saturn",
 "name": "Saturn",
 "defaults": {
 "reviewers": {
 "bob",
 "alice"
 }
 },
 "description": "Project Saturn is our moonshot project.",
 "members": [
 "bruno"
],
 "subgroups": [],
 "owners": [],
 "branches": [
 {
 "id": "main",
 "name": "main",
 "paths": [
 "//depot/saturn/..."
]
 }
]
 }
]
 }
}

```

```
"defaults": {
 "reviewers": {
 "Aruna_Gupta": []
 }
},
"moderators": [],
"moderators-groups": [],
"workflow": null,
"retainDefaultReviewers": false,
"minimumUpVotes": null
}
],
"jobview": null,
"emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
},
"tests": {
 "enabled": false,
 "url": null,
 "postBody": null,
 "postFormat": "url"
},
"deploy": {
 "enabled": false,
 "url": null
},
"deleted": false,
"private": false,
"workflow": null,
"retainDefaultReviewers": false,
"minimumUpVotes": null
}
]
}
}
```

**If a request fails**

<error code>:

400 Invalid data specified for the project or invalid references such as references to tests and workflows.

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
```

```
"code" : "<code string>",
"text" : "<error message>"
}},
"data" : null
}
```

## Update a project

### Summary

Update a P4 Code Review project.

PATCH /api/v10/projects/{{PROJECTID}}

### Description

Update details for a P4 Code Review project.

#### Tip:

If sharing is switched off for a workflow, any projects or branches that were associated with the workflow while it was shared will remain associated with it.

If you edit a project or branch associated with that unshared workflow, you will still see the name of the workflow in the **Workflow** field, even if you don't own that workflow. If you remove that workflow from the project/branch you will not be able to see the workflow in the **Workflow** dropdown list unless you own it.

#### Note:

By default, any member of a project can edit the project's configuration. Administrators can configure P4 Code Review to prevent changes to the project's name and branch definition(s). See ["Projects" on page 659](#) for details.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Project ID	string	path	Yes

### Example usage

Update project Saturn:

```
curl -X PATCH -u "username:ticket" "https://my-swarm-host/api/v10/projects/saturn"
```

```
{"workflow": "1"}
```

P4 Code Review responds with the details of the updated project.

```
{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "name": "Saturn",
 "defaults": {
 "reviewers": ["bob", "alice"]
 },
 "description": "",
 "members": [
 "bruno", "alice", "bob"
],
 "subgroups": [],
 "owners": [
 "bob"
],
 "branches": [
 {
 "id": "main",
 "name": "main",
 "paths": ["//depot/saturn/..."],
 "defaults": {
 "reviewers": ["bob"]
 }
 }
],
 "jobview": "",
 "emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
 },
 "tests": {
 "enabled": false,
 "url": "",
 "postBody": "",
 "postFormat": "url"
 },
 "deploy": {
 "enabled": false,
 "url": ""
 },
 "deleted": true,
 "private": false,
 "workflow": 1,
 "retainDefaultReviewers": false,
 }
]
 }
}
```



```

 "minimumUpVotes": null,
 "id": "saturn"
 }
]
}
}

```

**If a request fails**

<error code>:

404 Project does not exist or you do not have permission to view it

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Delete a project

**Summary**

Delete a P4 Code Review project.

DEL /api/v10/projects/{{PROJECTID}}

**Description**

Delete a P4 Code Review project.

**Note:**

- Users with *super* or *admin* privileges in P4 Server can delete projects.
  - If the P4 Code Review user has insufficient permissions, cleanup on the deleted P4 projects will fail.
- Project owners can delete projects that they own.
- If a project has no owners, any member of the project can delete the project.

When you delete a project, P4 Code Review will try to clean it up by doing the following:

- The deleted project is removed from the P4 Code Review project list on the dashboard, and from project searches.

- The deleted project is removed from the profile page of the project owners, members, moderators, and followers.
- The project name is removed from the **Project** column on the ["Reviews list" on page 406](#) page.
- Reviews that belong to the deleted project are not changed. The open and closed reviews remain accessible, their [review states](#), [comments](#), and [tasks](#) can be modified as normal.
- Reviews that belong to the deleted project, the project branch name link in the [review heading](#) is replaced with a link to the common depot location that contains the files included in the review.

**Note:**

The deleted project name cannot be reused for a new project. This behavior is not case sensitive, this means that if you delete a project called **Project B** you cannot create a new project called **project b**.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Project ID	string	path	Yes

**Example usage**

Delete project Saturn:

```
curl -X DEL -u "username:ticket" "https://my-swarm-host/api/v10/projects/saturn"
```

```
{
 "id": "saturn",
 "name": "Saturn",
 "description": "Project Saturn is our moonshot project.",
 "jobView": "project=swarm",
 "members": ["bruno", "alice", "bob"],
 "defaults": {
 "reviewers": ["bob", "alice"]
 },
 "tests": {
 "enabled": false
 },
 "workflow": 12,
 "branches": [
 {
 "id": "main",
 "name": "main",

```

```

 "paths": ["//depot/saturn/..."],
 "defaults": {
 "reviewers": ["bob"]
 }
 }
]
}

```

P4 Code Review responds with the details of the deleted project.

```

{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "name": "Saturn",
 "defaults": {
 "reviewers": ["bob", "alice"]
 },
 "description": "",
 "members": [
 "bruno", "alice", "bob"
],
 "subgroups": [],
 "owners": [
 "bob"
],
 "branches": [
 {
 "id": "main",
 "name": "main",
 "paths": ["//depot/saturn/..."],
 "defaults": {
 "reviewers": ["bob"]
 }
 }
],
 "jobview": "",
 "emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
 },
 "tests": {
 "enabled": false,
 "url": "",
 "postBody": "",
 "postFormat": "url"
 }
 }
]
 }
}

```

```
"deploy": {
 "enabled": false,
 "url": ""
},
"deleted": true,
"private": false,
"workflow": null,
"retainDefaultReviewers": false,
"minimumUpVotes": null,
"id": "saturn"
}
]
}
}
```

**If a request fails**

<error code>:

- 401 This operation is limited to project or group owners.
- 404 Project does not exist or you do not have permission to view it

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Undelete a Project

**Summary**

Undelete a P4 Code Review project.

POST /api/v10/projects/{PROJECTID}/undelete

**Description**

Undelete a P4 Code Review project.

**Note:**

- Users with *super* or *admin* privileges in P4 Server can undelete projects.
- Project owners can undelete projects that they own.
- If a project has no owners, any member of the project can undelete the project.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
<code>id</code>	Project ID	string	path	Yes

**Example usage**

Undelete project Saturn that has two branches:

```
curl -X POST -u "username:ticket" "https://my-swarm-host/api/v10/projects/saturn/undelete"
```

P4 Code Review responds with the details of the undeleted project.

```
{
 "error": null,
 "messages": [],
 "data": {
 "projects": [
 {
 "name": "Saturn",
 "defaults": {
 "reviewers": []
 },
 "description": null,
 "members": [
 "mei"
],
 "subgroups": [],
 "owners": [],
 "branches": [
 {
 "name": "branch1",
 "paths": [
 "//depot/paths/..."
],
 "id": "branch1",
 "moderators": [],
 "moderators-groups": []
 }
]
 }
]
 }
}
```

```
 "defaults": {
 "reviewers": []
 },
 "workflow": null,
 "retainDefaultReviewers": false,
 "minimumUpVotes": null
 },
 {
 "name": "branch2",
 "paths": [
 "//gwt/..."
],
 "id": "branch2",
 "moderators": [],
 "moderators-groups": [],
 "defaults": {
 "reviewers": []
 },
 "workflow": null,
 "retainDefaultReviewers": false,
 "minimumUpVotes": null
 }
],
"jobview": null,
"emailFlags": {
 "change_email_project_users": "1",
 "review_email_project_members": "1"
},
"deleted": false,
"private": false,
"workflow": null,
"retainDefaultReviewers": false,
"minimumUpVotes": null,
"id": "saturn"
}
]
}
}
```

**If a request fails**

<error code>:

- 401 This operation is limited to project or group owners.
- 404 Project does not exist or you do not have permission to view it.

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Reviews: P4 Code Reviews

### Get a list of reviews

#### Summary

Get a list of reviews.

GET /api/v10/reviews

#### Description

List the reviews the user has permission to view.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

## Parameters

Parameter	Description	Type	Parameter Type	Required
max	Specify the maximum number of results to return as an integer greater than 0. If the max parameter is not specified, the 50 most recent reviews are returned.	integer	query	No
project[]	To limit the results to a list of reviews in a specific project or projects, specify the projects you want reviews returned for.	string or an array (of strings)	query	No
after	<p>A review ID to seek to. Reviews up to and including the specified id are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.</p> <p>Reviews are returned in the order they were created in, starting with the most recently created review.</p> <div><b>Note:</b> You cannot use the after and afterUpdated parameters in the same request, they are mutually exclusive.</div>	string	query	No



Parameter	Description	Type	Parameter Type	Required
afterUpdated	<p>Return reviews updated on the day before the date/time specified in the afterUpdated parameter. The parameter value must be set in seconds since epoch. The seconds since epoch value for the day reviews are returned for is included at the end of the response.</p> <p>Used by the P4 Code Review team to step backwards one day at a time, loading reviews in an iterative process by using the value returned in a response as the afterUpdated parameter value for the next request.</p> <p>To return the most recently updated reviews on top, use the resultOrder parameter in conjunction with afterUpdated parameter. For example, resultOrder=updated</p> <p><b>For example:</b></p> <ul style="list-style-type: none"> <li>▪ <b>Requested:</b> afterUpdated=1606233362 (Tuesday November 24th, 2020, 15:56:02)</li> <li>▪ <b>Response fetched for:</b> any reviews updated on Monday November 23rd, 2020. 1606175999 (Monday, 23 November 2020)</li> </ul>	string	query	No

Parameter	Description	Type	Parameter Type	Required
	23:59:59) is added to the end of the response			
	<b>Note:</b> You cannot use the after and afterUpdated parameters in the same request, they are mutually exclusive.			
resultOrder	Return most recently updated reviews on top. Valid value is updated. For example, resultOrder=updated  When using resultOrder=updated, reviews will be retrieved for an entire day at a time. If there are hundreds or thousands of reviews updated that day, then all of them will be retrieved, regardless of the maximum number of reviews allowed to be displayed on the review page.	string	query	No

Parameter	Description	Type	Parameter Type	Required
state[]	<p>To limit the results to a list of reviews in a specific state or states, specify the states you want reviews returned for.</p> <p><b>Valid review states are:</b></p> <ul style="list-style-type: none"><li>▪ needsRevision review needs revision</li><li>▪ needsReview review needs review</li><li>▪ approved review is approved</li><li>▪ approved:isPending review is approved but not committed</li><li>▪ approved:commit review is approved and committed</li><li>▪ approved:notPending review is approved and committed</li><li>▪ rejected review is rejected</li><li>▪ archived review is archived</li></ul>	string or an array (of strings)	query	No

Parameter	Description	Type	Parameter Type	Required
keywords	To limit the results to a list of reviews that contain a specified keyword in a specified review field, use the keywords parameter in conjunction with the keywordsFields parameter. Multiple keywordFields can be specified.	string	query	No but required if keywordsFields is used
keywordsFields[]	<p>To limit the results to a list of reviews that contain a specified keyword in a specified review field, use the keywordsFields parameter in conjunction with the keywords parameter. Multiple keywordFields can be specified.</p> <p><b>Valid keywordFields indexed on a review are:</b></p> <ul style="list-style-type: none"> <li>▪ changes</li> <li>▪ author</li> <li>▪ participants</li> <li>▪ hasReviewer</li> <li>▪ description</li> <li>▪ updated</li> <li>▪ projects</li> <li>▪ state</li> <li>▪ testStatus</li> <li>▪ pending</li> <li>▪ groups</li> <li>▪ id</li> </ul>	string or an array (of strings)	query	No but required if keywords is used

Parameter	Description	Type	Parameter Type	Required
<code>fields[]</code>	To limit the fields returned for the reviews, add the <code>fields</code> parameter to the request. Multiple fields can be specified.	string or an array (of strings)	query	No

### Example usage

#### Get a list of reviews

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews"
```

P4 Code Review responds with the review entities:

#### Tip:

By default, this is limited to the 50 most recent reviews. This can be changed by adding a `max` parameter value to the request.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12344,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 }
 }
]
}
```

```
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
 ...
 <other review ids formatted as above>
 ...
],
"totalCount": 1,
"lastSeen": 12345
}
}
```

**Get a list of the 25 most recent reviews**

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?max=25"
```

P4 Code Review responds with the 25 most recent review entities:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12344,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
```

```

 "bruno": [],
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
},
...
<other review ids formatted as above>
...
],
"totalCount": 1,
"lastSeen": 12345
}
}

```

Get a list of reviews for the myproject project

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?project=myproject"
```

P4 Code Review responds with the review entities for the myproject project:

**Tip:**

By default, this is limited to the 50 most recent reviews. This can be changed by adding a max parameter value to the request.

HTTP/1.1 200 OK

```
{
 "error": null,
```

```
"messages": [],
"data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12344,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
],
...
<other review ids formatted as above>
...
],
"totalCount": 1,
"lastSeen": 12345
}
}
```

Get a list of reviews for the myproject and gemini projects

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?project[]=myproject&project[]=gemini"
```



P4 Code Review responds with the review entities for the myproject and gemini projects:

**Tip:**

By default, this is limited to the 50 most recent reviews. This can be changed by adding a max parameter value to the request.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12344,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
],
 "id": 12356,
```

```
"type": "default",
"changes": [
 12350,
],
"commits": [],
"author": "jsmith",
"approvals": null,
"participants": ["jsmith"],
"participantsData": {
 "jsmith": []
},
"hasReviewer": 0,
"description": "Another one of my reviews",
"created": 1594265070,
"updated": 1594266228,
"projects": {
 "gemini": ["main"]
},
"state": "needsReview",
"stateLabel": "Needs Review",
"testStatus": "pass",
"testDetails": [],
"deployStatus": null,
"deployDetails": [],
"pending": true,
"commitStatus": [],
"groups": [],
"complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
},
},
...
<other review ids formatted as above>
...
],
"totalCount": 2,
"lastSeen": 12356
}
}
```

Get a list of reviews that have jam in the projects or description fields

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?keywords=jam&keywordsFields
[]=projects&keywordsFields[]=description"
```

P4 Code Review responds with the review entities with "jam" in either the projects or description fields:

**Tip:**

By default, this is limited to the 50 most recent reviews. This can be changed by adding a max parameter value to the request.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12340,
 "type": "default",
 "changes": [
 12339,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "jam": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
},
{
 "id": 12360,
 "type": "default",
```

```
"changes": [
 12359,
],
"commits": [],
"author": "jsmith",
"approvals": null,
"participants": ["jsmith"],
"participantsData": {
 "jsmith": []
},
"hasReviewer": 0,
"description": "Another one of my reviews and this one has jam in the description",
"created": 1594265070,
"updated": 1594266228,
"projects": {
 "gemini": ["main"]
},
"state": "needsReview",
"stateLabel": "Needs Review",
"testStatus": "pass",
"testDetails": [],
"deployStatus": null,
"deployDetails": [],
"pending": true,
"commitStatus": [],
"groups": [],
"complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
},
},
...
<other review ids formatted as above>
...
],
"totalCount": 2,
"lastSeen": 12360
}
}
```

#### Get a review associated with a change

Get details of a review associated with a specific change 12345. The curl example is using the `&fields=id` parameter to reduce the size of the output.

```
curl https://mywarm-url/api/v11/reviews?keywords=12345&keywordsFields[]=changes&fields=id
```

Swarm responds with the review associated with the change 12345 which is 12346.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [
 {
 "id": 12346
 }
],
 "totalCount": 1,
 "lastSeen": 12344
 }
}
```

#### Limit fields returned in a list of reviews

Get a list of reviews and limit the fields returned to id, author, and pending:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?fields[]=id&fields[]=pending&fields[]=author"
```

P4 Code Review responds with the review entities:

#### Tip:

By default, this is limited to the 50 most recent reviews. This can be changed by adding a max parameter value to the request.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "author": "bruno",
 "pending": true,
 },
 ...
 <other review ids formatted as above>
 ...
],
 "totalCount": 7,
 "lastSeen": 12352
}
```

Get a list of reviews created after review 12344

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?after=12344"
```

P4 Code Review responds with the review entities:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12345,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
],
 ...
 <other review ids formatted as above>
```

```

...
],
"totalCount": 7,
"lastSeen": 12352
}
}

```

#### Get a list of reviews updated during the day before the set date

Request reviews updated on the day before 1606233362 (Tuesday November 24th, 2020, 15:56:02)

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?afterUpdated=1606233362"
```

P4 Code Review return all reviews that were updated on the day before the date/time specified in your request. In this example, P4 Code Review responds with any reviews that were updated on Monday November 23rd 2020. The seconds since epoch value P4 Code Review used is added to the end of the response:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12345,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1606119044,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 }
]
}

```

```
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
 ...
 <other review ids formatted as above>
 ...
],
"totalCount": 25,
"lastSeen": "0",
"afterUpdated": "1606089600"
}
}
```

**Get a list of reviews that need review**

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?state=needsReview"
```

P4 Code Review responds with the review entities in the needsReview state:

**Tip:**

By default, this is limited to the 50 most recent reviews. This can be changed by adding a max parameter value to the request.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12344,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 }
 }
]
}
```



```

 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
 },
 ...
 <other review ids formatted as above>
 ...
],
"totalCount": 1,
"lastSeen": 12345
}
}

```

#### Get a list of reviews that are open

Reviews are considered open when they are in any one of three states:

- needsReview
- needsRevision
- approved:isPending

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews?state[]=approved:isPending&state[]=needsReview&state[]=needsRevision"
```

P4 Code Review responds with the review entities for open reviews:

**Tip:**

By default, this is limited to the 50 most recent reviews. This can be changed by adding a max parameter value to the request.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12344,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
],
 "id": 12356,
 "type": "default",
```

```

 "changes": [
 12350,
],
 "commits": [],
 "author": "jsmith",
 "approvals": null,
 "participants": ["jsmith"],
 "participantsData": {
 "jsmith": []
 },
 "hasReviewer": 0,
 "description": "Another one of my reviews",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "gemini": ["main"]
 },
 "state": "needsRevision",
 "stateLabel": "Needs Revision",
 "testStatus": "pass",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 },
 },
 ...
 <other review ids formatted as above>
 ...
],
"totalCount": 2,
"lastSeen": 12356
}
}

```

Get a list of reviews updated during the day before the set date and get the most recently updated review on top

Request reviews updated on the day before 1606233362 (Tuesday November 24th, 2020, 15:56:02) and order with the most recently updated review on top.

```
curl -u "username:ticket" "https://myswarm-
url/api/v10/reviews?afterUpdated=1606233362&resultOrder=updated"
```

P4 Code Review return all reviews that were updated on the day before the date/time specified in your request and orders the most recently updated review on top. In this example, P4 Code Review responds with any reviews that were updated on Monday November 23rd 2020. The seconds since epoch value P4 Code Review used is added to the end of the response:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [
 {
 "id": 12130,
 "type": "default",
 "changes": [
 12129,
 12131
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": [
 "bruno"
],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Lorem Ipsum is simply dummy text of the printing and typesetting industry.",
 "created": 1713180845,
 "updated": 1713180849,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": null,
 "previousTestStatus": "",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [
 "Administrators"
],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 0,

```

```

 "lines_deleted": 0
 }
},
{
 "id": 12127,
 "type": "default",
 "changes": [
 12126,
 12128
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": [
 "bruno"
],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "New Review",
 "created": 1713180715,
 "updated": 1713180718,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": null,
 "previousTestStatus": "",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [
 "Administrators"
],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 }
},
{
 "id": 12119,
 "type": "default",
 "changes": [
 12118,

```

```
 12120,
 12121
],
 "commits": [
 12121
],
 "author": "bruno",
 "approvals": {
 "mei": [
 1
]
 },
 "participants": [
 "bruno",
 "mei"
],
 "participantsData": {
 "bruno": [],
 "mei": {
 "vote": {
 "value": 1,
 "version": 1,
 "isStale": false
 }
 }
 },
 "hasReviewer": 1,
 "description": "Testttt",
 "created": 1713167013,
 "updated": 1713167061,
 "projects": [],
 "state": "approved",
 "stateLabel": "Approved",
 "testStatus": null,
 "previousTestStatus": "",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": false,
 "commitStatus": [],
 "groups": [
 "Administrators"
],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 }
}
```

```
}
},
{
 "id": 12115,
 "type": "default",
 "changes": [
 12114,
 12116,
 12117
],
 "commits": [
 12117
],
 "author": "bruno",
 "approvals": {
 "mei": [
 1
]
 },
 "participants": [
 "bruno",
 "mei"
],
 "participantsData": {
 "bruno": [],
 "mei": {
 "vote": {
 "value": 1,
 "version": 1,
 "isStale": false
 }
 }
 },
 "hasReviewer": 1,
 "description": "ANother new review",
 "created": 1713166904,
 "updated": 1713166936,
 "projects": [],
 "state": "approved",
 "stateLabel": "Approved",
 "testStatus": null,
 "previousTestStatus": "",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": false,
 "commitStatus": [],
 "groups": [
```

```
 "Administrators"
],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 }
},
{
 "id": 12111,
 "type": "default",
 "changes": [
 12110,
 12112,
 12113
],
 "commits": [
 12113
],
 "author": "bruno",
 "approvals": {
 "mei": [
 1
]
 },
 "participants": [
 "bruno",
 "mei"
],
 "participantsData": {
 "bruno": [],
 "mei": {
 "vote": {
 "value": 1,
 "version": 1,
 "isStale": false
 }
 }
 },
 "hasReviewer": 1,
 "description": "New Test Review",
 "created": 1713166672,
 "updated": 1713166740,
 "projects": [],
 "state": "approved",
 "stateLabel": "Approved",
 "testStatus": null,
```



```

 "previousTestStatus": "",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": false,
 "commitStatus": [],
 "groups": [
 "Administrators"
],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 "lines_deleted": 0
 }
 },
 "totalCount": 5,
 "lastSeen": null,
 "afterUpdated": 1713080340
}
}

```

#### If a request fails

<error code>:

- 400 one of the following:
  - the input specified must be an integer greater than 0
  - invalid keywordsFields field specified
- 401 you are not authorized to view reviews

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Get review information

### Summary

Get review information and metadata.

GET /api/v10/reviews/{id}

### Description

Retrieve all of the information and metadata for a review.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes
transitions	To limit the results to a list of the state transitions that are allowed for the review, add the transitions parameter to the request.	query	path	No
fields	To limit the fields returned for the review, add the fields parameter to the request. Multiple fields can be specified.	string or an array (of strings)	query	No

## Example usage

Fetch information for review number 12345

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews/12345"
```

P4 Code Review responds with a review entity:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [1,3],
 "commits": [],
 "author": "normal-user",
 "approvals": null,
 "participants": ["normal-user"],
 "participantsData": {
 "normal-user": []
 },
 },
 "hasReviewer": 0,
 "description": "Added a file\n",
 "created": 1576595749,
 "updated": 1576595754,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "transitions": {
 "needsRevision": "Needs Revision",
 "approved": "Approve",
 "approved:commit": "Approve and Commit",
 "rejected": "Reject",
 "archived": "Archive"
 },
 "testStatus": null,
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "versions": [
 {
 "difference": 1,
 "stream": "//jam/main",
```

```
 "streamSpecDifference": 0,
 "change": 2,
 "user": "normal-user",
 "time": 1576595752,
 "pending": true,
 "addChangeMode": "replace",
 "archiveChange": 3,
 "testRuns": {}
 },
 {
 ...
 <other "versions" formatted as above>
 ...
 }
],
"description-markdown": "Added a file"
}]
}
}
```

**Fetch a list of the state transitions that are allowed for review 12345**

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews/12345/transitions"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "transitions": {
 "needsRevision": "Needs Revision",
 "approved": "Approve",
 "approved:commit": "Approve and Commit",
 "rejected": "Reject",
 "archived": "Archive"
 }
 }
}
```

**Limit fields returned for review number 12345**

Limit the fields return for review 12345 to the review author and the committed status of the review.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews/12345?fields[]=commitStatus&fields[]=author"
```

P4 Code Review responds with a review entity:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [
 {
 "author": "normal-user",
 "commitStatus": []
 }
]
 }
}
```

**If a request fails**

<error code>:

404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get read by status for files in a review

### Summary

Get the read by status for each file in a review for the authenticated user.

GET /api/v10/reviews/{id}/files/readby

### Description

Get the read by status for each file in a review for the authenticated user. This is the status the user sets when they select **Mark file as read/Mark file as unread** for a file.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Example usage**

Get a list of the read by status of each file in review 12345 for the authenticated user.

```
curl -u "bruno:ticket" "https://myswarm-url/api/v10/reviews/12345/files/readby"
```

With `bruno` is the authenticated user, P4 Code Review responds with the following:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "filesInfo": [
 {
 "review": 12345,
 "depotFile": "///depot/jam/main/src/Build.com",
 "readBy": {
 "bruno": {
 "version": 1,
 "digest": "82D5161601D12DB46F184D3F0778A16D"
 }
 }
 }
]
 }
}
```

**If a request fails**

<error code>:

404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
```

```

"code" : "<code string>",
"text" : "<error message>"
}},
"data" : null
}

```

## Get transitions and items that are blocking approval for a review

### Summary

Get a list of transitions and items that are blocking approval for a review.

GET /api/v10/reviews/{{REVIEWID}}/transitions

### Description

Get a list of transitions and details about the items that are blocking approval for a review.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Pending moderator approvals required for a review

Provides a list of branch moderators that are blocking approval for a review. Depending on the configuration of `moderator_approval` configurable in the `SWARM_ROOT/data/config.php` file, the branch moderator approval is required.

Changing the state of any review associated with a moderated branch is restricted as follows:

Only moderators can approve or reject reviews on the branch.

To add moderators to a branch:

1. Click the **Manage moderators** button.
2. Choose the **Users** or **Groupstab**.
  - a. If you selected **Users**, start typing a the name of a user. Suggested users on the P4 Server will appear automatically.
  - b. If you selected **groups**, then all of the members of that group will have the same moderator privileges for that project branch. Use this option if you wanted to add several users at once.
3. Click Add.

After you finish setting up the branch and save the project, reviews on this branch can only be approved or rejected by moderators.

Moderators can also transition a review to any other state. Below is a summary of branch moderator roles and review state rules.

Role	Allowed actions	Restrictions
Moderators	<p>Can move a review to any state.</p> <p>Can approve or reject reviews.</p> <p>Can prevent automatic approvals.</p>	None (unless <code>disable_self_approve</code> is enabled for self-approval). For more information, see <a href="#">disable_self_approve</a> .
Authors (not moderators)	<p>Can change state to:</p> <ul style="list-style-type: none"><li>■ Needs review</li><li>■ Needs revision</li><li>■ Archived</li></ul> <p>Can attach committed changelists.</p>	Cannot approve or reject reviews.
Authors (also moderators)	<p>Can change state to:</p> <ul style="list-style-type: none"><li>■ Approve</li><li>■ Rejected</li><li>■ Needs review</li><li>■ Needs revision</li><li>■ Archived</li></ul> <p>Can attach committed changelists.</p>	Cannot approve their own reviews if <code>disable_self_approve</code> is enabled. For more information, see <a href="#">disable_self_approve</a> .





Role	Allowed actions	Restrictions
Project members	Can change state to: <ul style="list-style-type: none"> <li>Needs review</li> <li>Needs revision</li> </ul> Can attach committed changelists.	Cannot approve, reject, or archive reviews.
Other users	None	Cannot change review states.
All roles	Can start a review.	Cannot change state if it's not in their allowed states (for example, can't change from Rejected).

#### Additional notes

- Moderators can prevent automatic approvals. For more information about automatically approving reviews using workflow rules see ["Workflow rules" on page 510](#).
- By default, if a review spans multiple branches with different moderators, only one moderator from any branch needs to approve it. You can change this (via a global setting) to require one moderator per branch. If a moderator belongs to multiple branches, one approval counts for each. For instructions on how to configure moderator behavior, see ["Moderator behavior when a review spans multiple branches" on page 679](#).

To delete a moderator, click the delete icon against their name.

Branch moderators(1), only moderators can approve or reject reviews
Manage moderators

Name	Type	
 Allison Clayborne	User	

Rows per page: 25
1-1 of 1

**Minimum Up votes required for a review**

Provides the number of **Minimum Up votes** required for each project that you have access to. This includes the maximum number of **Minimum Up votes** required for all branches in a project blocking approval for a review.

For [private project](#), the maximum number of **Minimum Up votes** for all private projects and their branches that is blocking approval for a review is returned.

Use **Minimum up votes** to set the minimum number of up votes required for reviews in this branch.

A review cannot be approved until all of the [Required reviewers](#) have voted up the review and the **Minimum up votes** specified has been satisfied.

When this setting is defined at the project level, minimum number of up votes is automatically applied to all branches. However, if you set **Minimum up votes** to **1** or higher on a specific branch, that branch will use its own setting and ignore the project-level setting.

To use the project-level value, set **Minimum up votes** to **Inherit from project**. This is set by default for new branches.

To override the project setting and use a custom value for a certain branch, set **Minimum up votes** to **1** or higher on that branch.

**Notes on minimum up votes**

- If a review includes changes across multiple projects or branches, the minimum number of up votes on each project or branch must be met before the review can be approved.
- Required reviewers are included when counting up votes.
- When the workflow rule **Count votes up from** is set to **Members** for a project or branch, only up votes from members belong to that project or branch will count toward the minimum up votes requirement. For more information on the **Count votes up from** rule, see "[Workflow rules](#)" on page 510.

If workflows are disabled, all votes are counted, not just votes from project members.

**Restriction:** If the **Minimum up votes** requirement is higher than the total number of reviewers on a review, it will be impossible to approve it, even if every reviewer votes to approve. Take this into consideration when setting **Minimum up votes**.

**Tests blocking approval for a review**

Provides a list of workflow tests that have **Failed** or have an **In progress** status that is blocking approval for a review. For more information about a failed test, see [Tests](#).

**Example usage**

Example for review 12345 shows a list of transitions and pending moderator approvals for two projects

```
GET /api/v10/reviews/12345/transitions
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 error: null,
 messages: [],
 data: {
 transitions: {
 "needsRevision": "Needs Revision",
 "rejected": "Reject",
 "archived": "Archive"
 },
 blocked: {
 approved: {
 moderators: {
 projects: {
 {
 id: "project1",
 name: "Project1",
 branches: {
 {
 id: "main",
 name: "Main Line",
 Users: {
 {
 id: "earl",
 name: "Earl Ashby",
 vote: 0,
 },
 },
 },
 },
 },
 },
 },
 },
 },
 {
 {
 id: "project2",
 name: "Project2",
 branches: {
 {
 id: "main",
 name: "Main Line",
 Users: {
 {
 id: "quinn",
 name: "Quinn Cass",
 vote: 0,
 },
 },
 },
 },
 },
 },
 },
}
```

```
 },
 },
},
},
},
}
```

Example for review 12345 shows a list of transitions and two Minimum Up votes that are blocking approval

GET /api/v10/reviews/12345/transitions

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 error: null,
 messages: [],
 data: {
 transitions: {
 "needsRevision": "Needs Revision",
 "rejected": "Reject",
 "archived": "Archive"
 },
 blocked: {
 approved: {
 votes: {
 projects: {
 project1: {
 id: "project1",
 name: "Project1",
 minimumUpVotes: 2
 }
 }
 }
 }
 }
 }
}
```

Example for review 12345 shows a list of transitions and required Minimum Up votes that are blocking approval for a public project and a private project

GET /api/v10/reviews/12345/transitions

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 error: null,
 messages: [],
 data: {
 transitions: {
 "needsReview": "Needs Review",
 "rejected": "Reject",
 "archived": "Archive"
 },
 blocked: {
 approved: {
 votes: {
 projects: {
 jam: {
 id: "jam",
 name: "Jam",
 minimumUpVotes: 2
 }
 },
 privateProjects: 3
 }
 }
 }
 }
}
```

Example for review 12345 shows a list of transitions and three tests that are blocking approval

GET /api/v10/reviews/12345/transitions

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 error: null,
 messages: [],
 data: {
 transitions: {
 "needsReview": "Needs Review",
 "rejected": "Reject",
 "archived": "Archive"
 },
 blocked: {
 approved: {
 tests: ["1", "2", "3"],
 },
 }
 }
}
```

```
 },
 },
}
```

**If a request fails**

<error code>:

- 401 User must be authenticated
- 403 You do not have permission to view the review
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get list of files that changed in a review

**Summary**

Get a list of files that changed between specified versions of a review.

GET /api/v10/reviews/{id}/files?from={x}&to={y}

**Description**

Get a list of files that changed between specified versions of a review.

**The GET request returns the following:**

- **root**: the common root depot path of the files that changed between the specified versions of the review.
- **limited**: P4 Code Review limits the number of files presented for changelists based on the [max\\_changelist\\_files](#) setting for the P4 Server connection. By default, a maximum of 1000 files are returned:
  - **true**: the number of files in the review is  $\geq$  `max_changelist_files`. Some of the files changed in the specified review versions might not have been returned by this request.
  - **false**: the number of files in the review is  $<$  `max_changelist_files`. All of the files changed in the specified review versions have been returned by this request.

- **For each file:**

- `depotFile`: the filename and path of the file.
- `action`: how the file changed, **Add**, **Edit**, or **Delete**.
- `type`: the filetype of the file, **text**, **binary**, **symlink**, **unicode**, **utf8**, **utf16**, **apple**, or **resource**.

For more information on filetypes, see the [Base filetypes](#) section of the [P4 CLI Reference](#).

- `rev`: the file revision number.
- `fileSize`: the file size in bytes.
- `digest`: the file digest.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

#### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>id</code>	Review ID	integer	path	Yes

Parameter	Description	Type	Parameter Type	Required
from	<p>Specify a value for from:</p> <ul style="list-style-type: none"><li>0 specifies the base version of the review.</li><li>-1 specifies the current #head version of the review files in the P4 Server.</li><li>If omitted, from defaults to 0 which is the base version of the review.</li><li>Must not be greater than the latest version of the review.</li></ul>	integer	query	No

---



Parameter	Description	Type	Parameter Type	Required
to	<p>Specify a value for to:</p> <ul style="list-style-type: none"> <li>■ If omitted, to defaults to the latest version of the review.</li> <li>■ Cannot be 0 or -1</li> <li>■ Must not be greater than the latest version of the review.</li> </ul>	integer	query	No

### Example usage

Get a list of files that changed between version 1 and 3 of review 12345

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews/12345/files?from=1&to=3"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "root": "//jam/main/src",
 "limited": false,
 "files": [{
 "depotFile": "//jam/main/src/execonvms.c",
 "action": "edit",
 "type": "text",
 "rev": "3",
 "fileSize": "3343",
 "digest": "82D5161601D12DB46F184D3F0778A16D"
```

```
 },
 {
 "depotFile": "//jam/main/src/jam.h",
 "action": "add",
 "type": "text",
 "rev": "2",
 "fileSize": "7364",
 "digest": "2E94B379F201AD02CF7E8EE33FB7DA99"
 }
]
}
```

**If a request fails**

<error code>:

- 400 invalid review version number requested
- 403 Insufficient permissions to access the review
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get activity for a review

**Summary**

Get a list of activities for the specified review id.

GET /api/v10/reviews/{id}/activity

**Description**

Get a list of activities for the specified review id, filtered by permissions.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
after	An activity ID to seek to. Activity entries up to and including the specified ID are excluded from the results and do not count towards max. Useful for pagination. Commonly set to the lastSeen property from a previous query.	integer	query	No

Parameter	Description	Type	Parameter Type	Required
<code>max</code>	<p>Maximum number of activity entries to return. This does not guarantee that max entries are returned. It does guarantee that the number of entries returned won't exceed max. Server-side filtering may exclude some activity entries for permissions reasons.</p> <p>If the <code>max</code> parameter is not included in the request, only the most recent 100 activities are returned.</p>	integer	query	No
<code>fields</code>	<p>An optional comma-separated list (or array) of fields to show. Omitting this parameter or passing an empty value shows all fields.</p>	string	query	No

### Example usage

Get a list of activities for review 1236

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/reviews/1236/activity"
```

P4 Code Review responds with:

**Tip:**

By default, this is limited to the 100 most recent activities. This can be changed by adding a max parameter value to the request.

```
{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1646,
 "type": "review",
 "link": [
 "review",
 {
 "review": "1236",
 "version": 4
 }
],
 "user": null
 "action": "Automated tests reported",
 "target": "review 1236",
 "preposition": "failed tests for",
 "description": "Make a review.",
 "details": [],
 "topic": "reviews/1236",
 "depotFile": null,
 "time": 1620228301,
 "behalfOf": null,
 "projects": [],
 "followers": [
 "bruno"
 "bob"
],
 "streams": {
 "review-1236",
 "user-",
 "personal-",
 "personal-macy.winter",
 "personal-super"
 },
 "change": 1235
 },
 {
 {
 Other ids formatted as above
 }
],
 "totalCount": 100,
```

```
"lastSeen": 1646
}
```

#### Activity pagination

To get the second page of activities for the review limited to a max of 1 and showing specified fields only (based on the previous example):

```
curl -u "username:ticket"
"https://myswarm.url/api/v10
/reviews/1236/activity?max=1&after=1646&fields=id,action,date,description,type"
```

P4 Code Review responds with:

```
{
 "error": null,
 "messages": null,
 "data": {
 "activity": [
 {
 "id": 1645,
 "type": "review",
 "action": "Automated tests reported",
 "description": "Make a review.",
 "time": 1626945417
 }
],
 "totalCount": 1,
 "lastSeen": 1645
 }
}
```

#### If a request fails

<error code>:

- 400 One of the following:
  - The max parameter value must be an integer greater than 0
  - The after parameter must be an integer greater than 0
- 403 Insufficient permissions to access the review
- 404 Review does not exist
- 500 Internal error, request failed

HTTP/1.1 <response error code>

```
{
```

```

"error": <error code>,
"messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
}],
"data" : null
}

```

## Get a list of comments on a review

### Summary

Get a list of comments on a review.

GET /api/v10/reviews/{id}/comments

### Description

Get a list of comments on a review.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

### Example usage

#### Get a list of comments on a review

```
curl -u "username:ticket" "https://myswarm-url/api/v10/reviews/368/comments"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "84": {
 "id": 84,
 "topic": "reviews/368",
 "context": {
 "attribute": "description",
 "review": 368,
 "version": false
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "steve.russell",
 "time": 1594121651,
 "updated": 1594121651,
 "edited": null,
 "body": "@paula.boyle - can you take a look?",
 "readBy": []
 },
 "85": {
 "id": 85,
 "topic": "reviews/368",
 "context": {
 "file": "//projects/mercury/dev/src/api/client.cc",
 "leftLine": null,
 "rightLine": 7,
 "content": [
 " */",
 " ",
 " #include \"client.h\"",
 " ",
 " #define OK \"OK\""
],
 "type": "text",
 "review": 368,
 "version": 1
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
```



```

 "likes": [],
 "user": "paula.boyle",
 "time": 1594121935,
 "updated": 1594121935,
 "edited": null,
 "body": "Is this better? const char *OK=\"OK\";",
 "readBy": []
 },
 ...
 <other comments formatted as above>
 ...
}
}
]
}
}
}

```

#### If a request fails

<error code>:

- 403 Insufficient permissions to access the review
- 404 Review does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Get the number of delayed notifications on a review

### Summary

Gets the number of [delayed comment notifications](#) for the authenticated user for a review.

GET /api/v10/reviews/{id}/notify

### Description

Gets the number of [delayed comment notifications](#) for the authenticated user for a review.

**Example usage**

Get the number of delayed comment notifications for Bruno on review 12345:

```
curl -u "bruno:ticket" "https://myswarm-url/api/v10/reviews/12345/notify"
```

P4 Code Review responds with the number of delayed comment notifications for Bruno on Review 12345:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "userDelayedNotificationCount": 3
 }
}
```

**If a request fails**

<error code>:

404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get Dashboard reviews for the authenticated user

**Summary**

Gets a list of reviews for the authenticated user's dashboard

GET /api/v10/reviews/dashboard

**Description**

Gets a list of reviews for the authenticated user's dashboard. For information about when reviews are displayed in the user dashboard, see ["My Dashboard" on page 288](#).

In addition to the normal review information, the following information is returned for the dashboard filter:

- `roles` lists the roles the authenticated user has for the review.
- `projectsForUser` lists *projectids* the authenticated user is an owner, member, or moderator of.

### Example usage

Get the dashboard reviews for Bruno

```
curl -u "bruno:ticket" "https://myswarm-url/api/v10/reviews/dashboard"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [
 12344,
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": ["bruno"],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "Review for my code change",
 "created": 1594265070,
 "updated": 1594266228,
 "projects": {
 "myproject": ["main"]
 },
 "state": "needsRevision",
 "stateLabel": "Needs Revision",
 "testStatus": "fail",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 1,
 },
]
 }
}
```

```
 "lines_deleted": 0
 },
 "roles": [
 "author",
 "moderator"
],
},
...
<other review ids formatted as above>
...
],
"totalCount": 1,
"lastSeen": 12345,
"projectsForUser": [
 "jam"
 "mercury",
]
}
}
```

**If a request fails**

<error code>:

401 User must be authenticated

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get a .zip archive of files in a review

**Summary**

Download a .zip archive of the files in the most recent version of a review.

GET /api/v10/reviews/{id}/archive

**Description**

Get a .zip archive of the files in the most recent version of a review. The zip command-line tool must be installed on the P4 Code Review server to use this endpoint.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Example usage**

Download a .zip archive of the files in review 12345.

GET /api/v10/reviews/12345/archive

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "archive": {
 "phase": "initializing",
 "error": null,
 "success": false,
 "progress": 0
 }
 }
}
```

If you repeat the request during the compression phase P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "archive": {
 "phase": "compressing",
 "error": null,
 "success": false,
 "progress": 50
 }
 }
}
```

When .zip file has been created, P4 Code Review downloads it.

**If a request fails**

<error code>:

- 401 User must be authenticated
- 404 Review does not exist
- .Zip command line tool not installed

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Refresh projects associated with a review

**Summary**

Refresh the list of projects review files are associated with.

POST /api/v10/reviews/{id}/refreshProjects

**Description**

Use the `refreshProjects` API endpoint to check if the review files are associated with any projects created or updated after the review was last updated. If any projects are found that are not already associated with the review, the review is linked to them.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Example usage**

Refresh projects associated with review 12345

```
curl -X POST -u "username:ticket" "https://myswarm-url/api/v10/reviews/12345/refreshProjects"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [
 {
 "id": 12345,
 "type": "default",
 "changes": [
 12344
],
 "commits": [
 12344
],
 "author": "bruno",
 "approvals": null,
 "participants": [
 "bruno"
],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "streamy",
 "created": 1608644775,
 "updated": 1612371881,
 "projects": [
 "bluebook",
 "jam",
 "mercury"
],
 "state": "needsRevision",
 "stateLabel": "Needs Revision",
 "testStatus": null,
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": false,
 "commitStatus": [],
 "groups": [
 "swarm-project-my-precious",
 "swarm-project-jam",
 "Administrators"
],
 "complexity": {
 "files_modified": 1,
```

```
 "lines_added": 0,
 "lines_edited": 0,
 "lines_deleted": 0
 }
]
}
}
```

**If a request fails**

<error code>:

- 401 User must be authenticated
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Set vote for the authenticated user

**Summary**

Set the vote for the authenticated user to be up, down, or cleared

POST /api/v10/reviews/{id}/vote

**Description**

Set the vote for the authenticated user to be up, down, or cleared.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.



**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes
vote	Specify the vote value you want to set on the review. Valid vote values are up, down, and clear. If the requested vote value already exists on the latest version of the review, the vote is not changed.	string	body	Yes

Parameter	Description	Type	Parameter Type	Required
version	<p>Votes can only be set on the latest version of a review. The version parameter is used to check that you are voting on the latest version of the review.</p> <p>If the specified version is not the latest version of the review, the vote is not applied to the review and a 400 error is returned.</p> <p>If the version parameter is not included, the vote is applied to the latest version of the review.</p>	integer	body	No

---

### Example usage

#### Vote up version 1 of review 12345

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://myswarm-url/api/v10/reviews/12345/vote"
```

The "mybodyfilename.txt" file contains the authenticated user's vote and the version of the review they are voting on:

```
{
 "vote": "up"
 "version": "1"
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "vote": {
 "value": 1,
 "version": 1,
 "isStale": false
 }
 }
}
```

#### If a request fails

<error code>:

- 400 Version specified is not the latest version of the review
- 401 User must be authenticated
- 403 Insufficient permissions to access the review
- 404 Review does not exist
- 409 Invalid vote specified, the response error message lists valid votes.

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",
 "text": "<error message>"
 }],
 "data": null
}
```

## Change review state

### Summary

Change the state of a review.

POST /api/v10/reviews/{id}/transitions

**Description**

Change the state of a review. The state that a review can be changed to depends on a number of factors including its current state. You can check that the state change you want to make is allowed for the review by making a ["Get transitions and items that are blocking approval for a review" on page 1179](#) request first. For more information on review states and restrictions on changing them, see ["States" on page 464](#)

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
<u>id</u>	Review ID	integer	path	Yes

---

Parameter	Description	Type	Parameter Type	Required
transition	<p>Set the state transition to make to the review with the transition parameter.</p> <p><b>Review state transitions:</b></p> <ul style="list-style-type: none"><li>▪ needsRevision</li><li>▪ needsReview</li><li>▪ approved only available if the voting requirements for the review are satisfied</li><li>▪ committed only available for "Pre-commit model" on <a href="#">page 402</a> reviews that have been approved</li><li>▪ approved:commit only available for unapproved "Pre-commit model" on <a href="#">page 402</a> reviews when the voting requirements for the review are satisfied</li><li>▪ rejected</li><li>▪ archived</li></ul>	string	body	Yes

Parameter	Description	Type	Parameter Type	Required
jobs	Optional, add Perforce jobs to the review when the state is changed with the jobs parameter. Multiple jobs can be added but they must be comma separated. For information on Perforce jobs, see <a href="#">"Jobs" on page 315</a> .	array	body	No
fixedStatus	<p>Optional, only used if the jobs parameter is used. Specify the Perforce job status when the state is changed with the fixedStatus parameter. Options depend on the configuration of your jobspec fields, see <a href="#">"Jobs" on page 315</a>.</p> <p>The approved:commit state change updates the Perforce job description, all other state changes add the text as a comment on the job.</p>	String	body	No

---

Parameter	Description	Type	Parameter Type	Required
<code>cleanup</code>	<p>Only used when the review is committed, options are:</p> <ul style="list-style-type: none"><li>■ <code>true</code> P4 Code Review will attempt to automatically clean up any changelists left behind after the review has been committed, including removing any shelved files. This option can be removed by an administrator, see "<a href="#">Review cleanup</a>" on <a href="#">page 666</a> for details.</li><li>■ <code>false</code> P4 Code Review will not try to automatically cleanup any changelists left behind after the review is committed. This is the default if the <code>cleanup</code> parameter is not present.</li></ul>	Boolean	body	No

### Example usage

Change the state of review 12345 to **Approved and Committed**, add Perforce jobs "job000001" and "job000015", set job status to closed, and get P4 Code Review to attempt to cleanup any changelist that are left behind.

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://myswarm-url/api/v10/reviews/12345/transitions"
```

The "mybodyfilename.txt" file contains the state you want to change the review to and any other parameters you want to add:

```
{
 "transition": "approved:commit",
 "jobs": ["job000001", "job000015"],
 "fixStatus": "closed",
 "cleanup": true
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "state": "approved:commit",
 "stateLabel": "Approved and Committed"
 }
}
```

### If a request fails

<error code>:

- 400 Specified review state does not exist
- 401 Insufficient permissions to change review state
- 403 Insufficient permissions to access the review
- 404 Review does not exist
- 409 Specified state is not a valid transition for this review. API response message lists all valid transition states for the review

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",

```



```
"text" : "<error message>"
 },
 "data" : null
}
```

## Change review author

### Summary

Change review author.

PUT /api/v10/reviews/{id}/author

### Description

Change the review author.

#### Important:

By default you cannot change the author of a review, this option must be enabled by your P4 Code Review administrator. See ["Allow author change" on page 675](#) for details.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

### Example usage

#### Change the author of review 12345

```
curl -X PUT -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm-url/api/v10/reviews/12345/author"
```

The "mybodyfilename.txt" file contains the new author username:

```
{
 "author": "asmith"
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "author": "asmith"
 }
}
```

#### If a request fails

<error code>:

- 400 Author change not allowed, set allow\_author\_change to true
- 401 Insufficient permissions to change author
- 403 Insufficient permissions to access the review
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Replace review description

### Summary

Replace review description.

PUT /api/v10/reviews/{id}/description

### Description

Replace the review description with a new description.

**Optional: also replace pending changelist description**

Only available if you are editing the description of a pre-commit review and you are the original author of the changelist that created the review.

To also apply your review description changes to the original changelist description, specify `'updateOriginalChangelist' : true`.

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

**Example usage**

Replace the description for review 12345

```
curl -X PUT -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm-url/api/v10/reviews/12345/description"
```

The "mybodyfilename.txt" file contains the new description:

```
{
 "description": "This is my replacement description text and it is better than the original
description."
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
```

```
"description": "This is my replacement description text and it is better than the original
description.",
"description-markdown": "This is my replacement description text
and it is better than the original description."
}
}
```

**If a request fails**

<error code>:

- 401 Insufficient permissions to change description
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Replace review participants

**Summary**

Replace review participants.

PUT /api/v10/reviews/{id}/participants

**Description**

Replace the review participants with a new set of participants.

**Note:**

- The text file used for your request must contain a complete list of all of the participants you want on the review (users and groups). The existing review participants are completely replaced by the participants in the text file. For example, if you already have some group participants on the review and you do not specify any group roles in the text file, all of the participant groups will be removed from the review.

- Default reviewers can be removed from a review but retained default reviewers cannot be removed from a review, see ["Default reviewers" on page 502](#) and ["Retain default reviewers" on page 503](#).

If your request tries to remove a retained default reviewer, your request will succeed but with a message in the response indicating that the retained default reviewer has not been removed.

- You cannot reduce the voting requirement of a retained default reviewer but you can increase it, see ["Retain default reviewers" on page 503](#).

If your request tries to reduce the voting requirement of a retained default reviewer, your request will succeed but with a message in the response indicating that the retained default reviewer has not been changed.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

#### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>id</code>	Review ID	integer	path	Yes

#### Example usage

##### Replace participants for review 12345

```
curl -X PUT -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm-url/api/v10/reviews/12345/participants"
```

The "mybodyfilename.txt" file contains the new review participants and their voting requirements:

```
{
 "participants": {
 "users": {
 "asmith": {
 "required": "yes"
 },
 },
 },
}
```

```
"jbloggs": {
 "required": "no"
},
"molsen": {
 "required": "no"
}
},
"groups": {
 "docs": {
 "required": "none"
 },
 "qa": {
 "required": "all"
 },
 "dev": {
 "required": "one"
 },
 "testers": {
 "required": "one"
 }
}
}
```

P4 Code Review responds with a list of all of the participants and their voting requirements:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": {
 "combinedReviewers": [
 "'molsen' must be a reviewer with a minimum requirement of 'required'",
 "'testers' must be a reviewer with a minimum requirement of 'require all'"
]
 },
 "data": {
 "participants": {
 "users": {
 "asmith": {
 "required": "yes"
 },
 "jbloggs": {
 "required": "no"
 },
 "molsen": {
 "required": "yes",
 "minimumRequired": "yes"
 }
 }
 }
 }
}
```

```

 },
 "groups": {
 "dev": {
 "required": "one"
 },
 "docs": {
 "required": "none"
 },
 "qa": {
 "required": "all"
 },
 "testers": {
 "required": "all",
 "minimumRequired": "all"
 }
 }
 }
}
}

```

#### If a request fails

<error code>:

- 400 Invalid participant data, for example: unknown participant id or voting requirement
- 401 Insufficient permissions to replace participants
- 403 Insufficient permissions to edit reviewers
- 404 Review does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Append review participants

### Summary

Append review participants.

POST /api/v10/reviews/{id}/participants

## Description

Append more participants to the existing review participants.

### Note:

Only the users and groups specified in the request are appended to the review. For example, if you don't want to append any group participants to the review there is no need to include the groups role in the text file.

### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

## Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

## Example usage

### Append participants to review 12345

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://myswarm-url/api/v10/reviews/12345/participants"
```

The "mybodyfilename.txt" file contains the new review participants and their voting requirements. In this example, bjones is an optional participant:

```
{
 "participants": {
 "users": {
 "bjones": []
 }
 }
}
```

P4 Code Review responds with a list of all of the participants and their voting requirements:



HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "participants": {
 "users": {
 "asmith": {
 "required": "yes"
 },
 "jbloggs": {
 "required": "no"
 },
 "molsen": {
 "required": "yes",
 "minimumRequired": "yes"
 },
 "bjones": []
 },
 "groups": {
 "dev": {
 "required": "one"
 },
 "docs": {
 "required": "none"
 },
 "qa": {
 "required": "all"
 },
 "testers": {
 "required": "all",
 "minimumRequired": "all"
 }
 }
 }
 }
}
```

#### If a request fails

<error code>:

- 400 Invalid participant data, for example: unknown participant id or voting requirement
- 401 Insufficient permissions to append participants
- 403 Insufficient permissions to edit reviewers
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Add a comment to a review

### Summary

Add a comment to a review.

POST /api/v10/reviews/{id}/comments

### Description

Add a comment to a review. The user that makes the API request is set as comment author.

### Parameters

Parameter	Description	Type	Parameter Type	Required
body	The text content of the comment.	string	form	Yes
taskState	Optional task state of the comment. Valid values when adding a comment are comment (default) and open. This creates a plain comment or opens a task, respectively.  If the taskState parameter is not in the request, the task state is set to comment.	string	form	No

Parameter	Description	Type	Parameter Type	Required
notify	<p>Comment notifications can be delayed, see <a href="#">"Comment notification delay" on page 336</a>. The notify parameter enables you to set when the notification for the comment is sent.</p> <p>Valid options are:</p> <ul style="list-style-type: none"><li>■ <b>delayed</b> (default): notifications for this comment are delayed by the time set in <a href="#">"Comment notification delay" on page 580</a></li><li>■ <b>immediate</b>: the notification for this comment is sent immediately</li><li>■ <b>silent</b>: no notification is sent for this comment</li></ul>	string	query	No

Parameter	Description	Type	Parameter Type	Required
context	<p>Used to add the comment to a file in the review.</p> <p>Valid fields are:</p> <ul style="list-style-type: none"><li>▪ file mandatory unless attribute or comment are set: File to comment on. Valid only for changes and reviews topics. For example, //depot/main/README.txt.</li><li>▪ leftline optional, but if specified, you must also specify the rightline and content parameters. Integer: Left-side diff line number to attach the inline comment to. Valid only for changes and reviews topics.</li><li>▪ rightline optional, but if specified, you must also specify the leftline and content parameters. Integer: Right-side diff line number to attach the inline comment to. Valid only for changes and reviews topics.</li></ul>	array	form	No

---

Parameter	Description	Type	Parameter Type	Required
	<ul style="list-style-type: none"> <li>content optional, but if specified, you must also specify the leftline and rightline parameters. Array of strings: Provide the content of the codeline the comment is on and the four preceding codelines. This is used to specify a short excerpt of context in case the lines being commented on change during the review. When set to null, P4 Code Review makes an effort to build the content on its own. Because this involves file operations, it might be slow.</li> <li>version optional, integer: With a reviews topic, this field specifies which version to attach the comment to.</li> <li>attribute optional: Set to description to comment on the review description.</li> <li>comment optional, integer: Set to the comment id this comment is replying to.</li> </ul>			

## Example usage

### Add a comment to a review

Add a comment to review 885, make the comment an open task, and send the comment notification immediately:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/885/comments?notify=immediate"
```

The "mybodyfilename.txt" file contains the authenticated user's comment and makes the comment an open task:

```
{
 "body": "This is another comment.",
 "taskState": "open"
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 95,
 "topic": "reviews/885",
 "context": [],
 "attachments": [],
 "flags": [],
 "taskState": "open",
 "likes": [],
 "user": "bob",
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is another comment.",
 "readBy": [],
 "notify": "immediate"
 },
],
 "userDelayedNotificationCount": 1
 }
}
```

### Add a comment to a file in a review

Add a comment to the end of the //depot/main/README.txt file on review 123:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/123/comments"
```

The "mybodyfilename.txt" file contains the authenticated user's comment for the file:

```
{
 "body": "This is a comment on a file.",
 "context": {
 "file": "//depot/main/README.txt"
 }
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 105,
 "topic": "reviews/123",
 "context": {
 "file": "//depot/main/README.txt",
 "leftLine": null,
 "rightLine": null,
 "content": [],
 "version": 1,
 "change": 122,
 "review": 123,
 "md5": "6af0e07c2d96afacaf2da234242c3cbd",
 "name": "README.txt",
 "line": null
 },
 "attachments": [],
 "flags": [],
 "taskState": "open",
 "likes": [],
 "user": "bob",
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is a comment on a file.",
 "readBy": [],
 "notify": "delayed"
 },
],
 "userDelayedNotificationCount": 1
 }
}
```

**Add a comment to a line in a file and include context for the comment****Tip:**

To populate the content field, you need to know the content of the line you are commenting on and the content of the four lines before the line you are commenting on

Add a comment to a line in a file on review 110 with the following information:

- Full filepath and filename of the file to comment on
- Left and right line numbers of the diff to comment on
- Text of the codeline you are commenting on and the four codelines before that to provide context for the comment
- Review version to add the comment to

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/110/comments"
```

The "mybodyfilename.txt" file contains the details for the new comment:

```
{
 "body": "This is a comment on a line in a file with context.",
 "context": {
 "file": "//depot/main/README.txt"
 "rightLine": 54,
 "leftLine": null,
 "content": [
 "FORTRAN default = \"\" ;\n",
 "LIBDIR default = /boot/develop/libraries ;\n",
 "-LINK default = $(CC) ;\n",
 "+LINK default = mwld ;\n",
 "+LINKFLAGS default = \"\" ;\n"
],
 "version": 3
 }
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 143,
 "topic": "reviews/110",
 "context": {
 "file": "//depot/main/README.txt"
 "leftLine": null,

```



```

 "rightLine": 54,
 "content": [
 "FORTRAN default = \"\" ;\n",
 "LIBDIR default = /boot/develop/libraries ;\n",
 "-LINK default = $(CC) ;\n",
 "+LINK default = mwld ;\n",
 "+LINKFLAGS default = \"\" ;\n"
],
 "version": 3,
 "change": 109,
 "review": 110,
 "md5": "6af0e07c2d96afacaf2da234242c3cbd",
 "name": "README.txt",
 "line": 54
 }
 "attachments": [],
 "flags": [],
 "taskState": "open",
 "likes": [],
 "user": "bob"
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is a comment on a line in a file with context.",
 "readBy": [],
 "notify": "delayed"
},
],
"userDelayedNotificationCount": 1
}
}

```

#### Add a comment to a review description

Add a comment to the description of review 885 and delay the notification:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/885/comments?notify=delayed"
```

The "mybodyfilename.txt" file contains the authenticated user's comment and adds it to the review description:

```

{
 "body": "This is a comment on the description.",
 "context": {
 "attribute": "description"
 },
}

```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 99,
 "topic": "reviews/885",
 "context": {
 "attribute": "description"
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bob"
 "time": 1620813784,
 "updated": 1620813784,
 "edited": null,
 "body": "This is a comment on the description.",
 "readBy": [],
 "notify": "delayed"
 },
],
 "userDelayedNotificationCount": 1
 }
}
```

#### Reply to a comment

You have two options when replying to a comment. Although the URLs for the options are different, they both do the same job:

- ["Reply to a comment using POST /api/v10/reviews/{id}/comments" below](#)
- ["Reply to a comment using POST /api/v10/reviews/{id}/comments/{id}" on the facing page](#)

#### Reply to a comment using POST /api/v10/reviews/{id}/comments

Add a reply to comment id 99 in review 885:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/885/comments"
```

```
{
 "body": "This is a reply to a comment.",
 "context": {
 "comment": 99
 }
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
 {
 "id": 109,
 "topic": "reviews/885",
 "context": {
 "comment": 99,
 "version": 2
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bruno"
 "time": 1622712940,
 "updated": 1622712940,
 "edited": null,
 "body": "This is a reply to a comment.",
 "readBy": [],
 "notify": "delayed"
 }
],
 "userDelayedNotificationCount": 1
 }
}
```

#### Reply to a comment using POST /api/v10/reviews/{id}/comments/{id}

Add a reply to comment id 99 in review 885:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/885/comments/99"
```

```
{
 "body": "This is another reply to a comment."
}
```

P4 Code Review responds the comment entity:

```
{
 "error": null,
 "messages": [],
 "data": {
 "comments": [
```

```
{
 "id": 120,
 "topic": "reviews/885",
 "context": {
 "comment": 99,
 "version": 2
 },
 "attachments": [],
 "flags": [],
 "taskState": "comment",
 "likes": [],
 "user": "bruno"
 "time": 1622712940,
 "updated": 1622712940,
 "edited": null,
 "body": "This is another reply to a comment.",
 "readBy": [],
 "notify": "delayed"
}
]
"userDelayedNotificationCount": 1
}
```

**If a request fails**

<error code>:

- 400 one of the following:
  - the idspecified must be an integer greater than 0
  - invalid context field specified, valid fields are file, leftline, rightline, content, version, attribute, and comment
  - invalid notify value specified, value must be delayed, immediate, or silent
  - invalid taskstate value specified, value must be open or comment
- 403: you are not authorized to create this comment
- 404 the review, change, job, review version, file line number, or file specified does not exist
- 409 invalid topic\_subject specified, value must be reviews, changes, or jobs
- 500 internal error, request failed

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
```

```

 "text" : "<error message>"
 },
 "data" : null
}

```

## Append change to a pre-commit review

### Summary

Append a changelist to the changelist in a pre-commit review.

POST /api/v10/reviews/{id}/appendchange

### Description

Append a changelist to the changelist in a pre-commit review. Not available for post-commit reviews. For information about what happens when you append a changelist to a review, see ["Append a pending changelist to a review" on page 451](#).

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes
change_id	Changelist number	integer	body	Yes

### Example usage

Append changelist 12456 to review 12345:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/12345/appendchange"
```

```
{
 "changeId": 12456
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [12344, 12346, 12456],
 "commits": [],
 "author": "normal-user",
 "approvals": null,
 "participants": ["normal-user"],
 "participantsData": {
 "normal-user": []
 },
 "hasReviewer": 0,
 "description": "Added a file\n",
 "created": 1576595749,
 "updated": 1576595754,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "transitions": {
 "needsRevision": "Needs Revision",
 "approved": "Approve",
 "approved:commit": "Approve and Commit",
 "rejected": "Reject",
 "archived": "Archive"
 },
 "testStatus": null,
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "versions": [
 {
 "difference": 1,
 "stream": "//jam/main",
 "streamSpecDifference": 0,
 }
]
 }
 }
}
```

```

 "change": 2,
 "user": "normal-user",
 "time": 1576595752,
 "pending": true,
 "addChangeMode": "append",
 "archiveChange": 3,
 "testRuns": {}
 },
 {
 "description-markdown": "Added a file"
 }
}
}

```

#### If a request fails

<error code>:

- 401 Insufficient permissions to add change
- 403 Insufficient permissions to access the review
- 404 Review or change does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Replace change on a review

### Summary

Replace the changelist on a review.

POST /api/v10/reviews/{id}/replacewithchange

### Description

Replace the changelist on a review. Can be used for pre-commit and post-commit reviews. For information about what happens when you replace a changelist on a review, see ["Replace review with a pending changelist" on page 452](#) and ["Replace review with a committed changelist" on page 453](#).

**Tip:**

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes
change_id	Changelist number	integer	body	Yes

**Example usage**

Replace changelist on review 12345 with changelist 12556:

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://my-swarm-host/api/v10/reviews/12345/replacewithchange"
```

```
{
 "changeId": 12256
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [12556],
 "commits": [],
 "author": "normal-user",
```



```

 "approvals": null,
 "participants": ["normal-user"],
 "participantsData": {
 "normal-user": []
 },
 "hasReviewer": 0,
 "description": "Added a file\n",
 "created": 1576595749,
 "updated": 1576595754,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "transitions": {
 "needsRevision": "Needs Revision",
 "approved": "Approve",
 "approved:commit": "Approve and Commit",
 "rejected": "Reject",
 "archived": "Archive"
 },
 "testStatus": null,
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "versions": [
 {
 "difference": 1,
 "stream": "//jam/main",
 "streamSpecDifference": 0,
 "change": 2,
 "user": "normal-user",
 "time": 1576595752,
 "pending": true,
 "addChangeMode": "replace",
 "archiveChange": 3,
 "testRuns": []
 },
 {
 "description-markdown": "Added a file"
 }
]
 }
}

```

If a request fails

<error code>:

- 401 Insufficient permissions to add change
- 403 Insufficient permissions to access the review
- 404 Review or change does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Send all delayed comment notifications on a review

### Summary

Send all of the [delayed comment notifications](#) for the authenticated user for a review.

POST /api/v10/reviews/{review\_id}/notify

### Description

Send all of the [delayed comment notifications](#) for the authenticated user for a review.

### Example usage

Send all of the delayed comment notifications for Bruno on review 12345:

```
curl POST -u "bruno:ticket" "https://myswarm-url/api/v10/reviews/12345/notify"
```

P4 Code Review responds with the number of delayed notifications sent:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "userDelayedNotificationCount": 3
 }
}
```

### If a request fails

<error code>:

404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Delete review participants

### Summary

Delete review participants.

DELETE /api/v10/reviews/{id}/participants

### Description

Delete review participants from a review.

#### Note:

- Only the users and groups specified in the request are deleted from the review. For example, if you don't want to delete any group participants from the review there is no need to include the groups role in the text file.
- Default reviewers can be removed from a review but retained default reviewers cannot be removed from a review, see ["Default reviewers" on page 502](#) and ["Retain default reviewers" on page 503](#).

If your request tries to remove a retained default reviewer, your request will succeed but with a message in the response indicating that the retained default reviewer has not been removed.

#### Tip:

- Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

**Example usage****Delete participants from review 12345**

```
curl -X DELETE -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://myswarm-url/api/v10/reviews/12345/participants"
```

The "mybodyfilename.txt" file contains the participants you want to delete from the review:

```
{
 "participants": {
 "users": {
 "jbloggs": [],
 "molsen": []
 },
 "groups": {
 "docs": []
 }
 }
}
```

P4 Code Review responds with a list of all of the participants and their voting requirements:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": {
 "combinedReviewers": [
 "'molsen' must be a reviewer with a minimum requirement of 'required'"
]
 },
 "data": {
 "participants": {
 "users": {
 "asmith": {
 "required": "yes"
 },
 "molsen": {
```

```

 "required": "yes",
 "minimumRequired": "yes"
 },
 "bjones": []
},
"groups": {
 "dev": {
 "required": "one"
 },
 "qa": {
 "required": "all"
 },
 "testers": {
 "required": "all",
 "minimumRequired": "all"
 }
}
}
}
}
}

```

#### If a request fails

<error code>:

- 400 Unknown participant id
- 401 Insufficient permissions to delete participants
- 403 Insufficient permissions to edit reviewers
- 404 Review does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Obliterate a review

### Summary

Obliterate a review.

DELETE /api/v10/reviews/{id}

## Description

### Important:

Obliterate must be used with care, the review and all of its associated metadata are permanently deleted. An obliterated review cannot be reinstated, not even by Perforce Support.

### Note:

- By default, you must be a user with *admin* or *super* user rights to obliterate a review.
- Optional:** P4 Code Review can be configured to allow users to obliterate reviews that they have authored. This can be configured by your P4 Code Review administrator. See ["Allow author obliterate review" on page 675](#).

Obliterate is used to permanently delete reviews that have been created by mistake. For instance, if a review is associated with the wrong changelist, or a review contains sensitive information that should not be openly available.

For information on what happens to a review when it is obliterated, see ["When you obliterate a review" on page 657](#).

### Tip:

- Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

## Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

## Example usage

Obliterate review number 12345

```
curl -X DELETE -u "username:ticket" "https://myswarm-url/api/v10/reviews/12345"
```

P4 Code Review responds with a message informing you that it has successfully Obliterated the review:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [{
 "code": "review-obliterated",
 "text": "The review with id [12345] has been obliterated."
 }],
 "data": {
 "reviews": [{
 "id": 12345,
 "type": "default",
 "changes": [1, 3],
 "commits": [],
 "author": "normal-user",
 "approvals": null,
 "participants": ["normal-user"],
 "participantsData": {
 "normal-user": []
 },
 "hasReviewer": 0,
 "description": "Hello\n",
 "created": 1576579928,
 "updated": 1576579932,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": null,
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": []
 },
 "versions": [
 {
 "difference": 1,
 "stream": "//jam/main",
 "streamSpecDifference": 0
 },
 {
 "change": 2,
 "user": "normal-user",
 "time": 1576595752,
 "pending": true,
 "addChangeMode": "replace",
 "archiveChange": 3
 },
 {
 "testRuns": []
 }
]
 }
}
```

```
...
 <other "versions" formatted as above>
 ...
}
}
}
}
}
```

**Obliterate review number 56789 (with insufficient permission)**

```
curl -X DELETE -u "username:ticket" "https://myswarm-url/api/v10/reviews/56789"
```

P4 Code Review responds with a message informing you that you do not have permission to obliterate the review:

HTTP/1.1 200 OK

```
{
 "error":403,
 "messages":[{"
 "code": "forbidden-exception",
 "text": "Failed to Obliterate the review, you need to have admin privileges or be the review
author with 'allow_author_obliterate' set to true."
 }]
 "data": null
}
```

**If a request fails**

<error code>":

- 403 Insufficient permissions to obliterate review
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```



## Enable notification for a review participant

### Summary

Enable notifications for a review participant.

DELETE /api/v10/reviews/{id}/participantNotifications

### Description

Enable notifications for a review participant. Once notifications are enabled, when you become a review participant, by joining the review or being [@mentioned](#) in a comment or in the review's description, you receive notifications for any events associated with the review.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

### Example usage

```
curl -X DELETE -u "username:ticket" "https://myswarm-url/api/v10/reviews/12345/participantNotifications"
```

P4 Code Review responds with a list of all the review participants.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "participantsData": {
 "Aruna_Gupta": [],
 "bruno": [],
 "mei": [],
 "Sunil_Patil": [],
 "swarm-group-developers": []
 }
 }
}
```

### If a request fails

<error code>":

- 401 Unauthorized
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Disable notification for a review participant

### Summary

Disable notifications for a review participant.

POST /api/v10/reviews/{id}/participantNotifications

### Description

Disable notifications for a review participant.

Once notifications are disabled, you no longer receive notifications. However, if you are [@mentioned](#) in a subsequent review comment, you do receive a notification for that comment; regular notifications remain disabled. This approach ensures that you don't miss anything that other reviewers or the review author deems important.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

### Example usage

```
curl -X POST -u "username:ticket" "https://myswarm-
url/api/v10/reviews/12345/participantNotifications"
```

P4 Code Review responds with a list of all the review participants.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "participantsData": {
 "Aruna_Gupta": [],
 "bruno": {
 "notificationsDisabled": true
 },
 "mei": [],
 "Sunil_Patil": [],
 "swarm-group-developers": []
 }
 }
}
```

**If a request fails**

<error code>":

- 401 Unauthorized
- 404 Review does not exist

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Join a review

### Summary

Join a review.

POST /api/v10/reviews/{id}/join

## Description

### Important:

The following individuals can be added as a reviewer:

- Users with *admin* or *super* privileges.
- If the review is **moderated**, the moderators.
- If the review is part of a project, but not moderated, all project members.
- If the review is not part of a project, any authenticated user.

An authenticated user can add themselves as a reviewer for a review.

## Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

## Example usage

Add vera to review 12345:

```
curl -X POST -u "vera:ticket" "https://myswarm-url/api/v10/reviews/12345/join"
```

P4 Code Review responds with a list of all the review participants.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "participants": {
 "users": {
 "Aruna_Gupta": {
 "vote": {
 "value": 1,
 "version": 2,
 "isStale": true
 }
 },
 "bruno": [],
 "earl": {
 "vote": {
 "value": -1,
```

```
 "version": 1,
 "isStale": true
 },
 "mei": [],
 "vera": []
},
"groups": {
 "staletest": []
}
}
```

If a request fails

<error code>":

- 401 Unauthorized
- 404 Review does not exist

Leave a review

Summary

Leave a review.

DEL /api/v10/reviews/{id}/leave

Description

**Tip:**  
You cannot leave a review if you are a retained default reviewer on the review or you are a member of a group that is a reviewer on the review.

An authenticated user can remove themselves as a reviewer from a review.

Parameters

Parameter	Description	Type	Parameter Type	Required
id	Review ID	integer	path	Yes

**Example usage**

Remove vera from review 12345:

```
curl -X DEL -u "vera:ticket" "https://myswarm-url/api/v10/reviews/12345/leave"
```

P4 Code Review responds with a list of all the review participants.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "participants": {
 "users": {
 "Aruna_Gupta": {
 "vote": {
 "value": 1,
 "version": 2,
 "isStale": true
 }
 },
 "bruno": [],
 "earl": {
 "vote": {
 "value": -1,
 "version": 1,
 "isStale": true
 }
 },
 "mei": []
 },
 "groups": {
 "staletest": []
 }
 }
 }
}
```

**If a request fails**

<error code>:

- 401 Unauthorized
- 404 Review does not exist

## Archive inactive reviews

### Summary

Archive inactive reviews.

POST /api/v10/reviews/archiveInactive

### Description

Archiving reviews not updated since a specific date.

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
notUpdatedSince	Not updated since date. Requires the date to be in the format YYYY-mm-dd, for example 2023-06-06	string	body	Yes
max	Optional, maximum number of reviews.	integer	body	No
description	Optional, a description that is posted as a comment for archiving.	string	body	No

**Example usage**

Archive maximum 50 reviews inactive since 2023/06/06.

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm-url/api/v10/reviews/archiveInactive"
```

The "mybodyfilename.txt" file contains the not updated since date, the maximum number of reviews, and a description:

```
{
 "notUpdatedSince": "2023-06-06",
 "max": 50,
 "description": "This is the description"
}
```

P4 Code Review responds with a list of all the inactive reviews that are archived.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "archivedReviews": [
 {
 "id": 12450,
 "type": "default",
 "changes": [
 12443,
 12451
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": [
 "bruno"
],
 "participantsData": {
 "bruno": []
 },
 "hasReviewer": 0,
 "description": "pre commit review creation via swarm UI success",
 "created": 1678794209,
 "updated": 1683615474,
 "projects": {
 "project3": [
 "branch1",
 "branch2",
 "branch3"
]
 }
 }
]
 }
}
```



```

 },
 "state": "archived",
 "stateLabel": "Archived",
 "testStatus": null,
 "previousTestStatus": "",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [
 "Administrators",
 "swarm-project-mod-project1",
 "swarm-project-mod-project2",
 "swarm-project-mod-project3",
 "swarm-project-mod-project4",
 "swarm-project-project-1",
 "swarm-project-project-2",
 "swarm-project-project-3",
 "swarm-project-project1",
 "swarm-project-project2",
 "swarm-project-project3",
 "swarm-project-project4",
 "swarm-project-project88",
 "swarm-project-project99",
 "swarm-project-test-project"
],
 "complexity": {
 "files_modified": 1,
 "lines_added": 0,
 "lines_edited": 2,
 "lines_deleted": 0
 }
 }
]
}
}

```

#### If a request fails

<error code>:

- 401 Unauthorized
- 400 Transition of the review is not allowed

## Create a Review

### Summary

Create a review.

POST /api/v10/reviews

### Description

Pass in a changelist ID to create a review. Optionally, you can also provide a description and a list of reviewers. For more information about reviewers, individual reviewers, and group reviewers, see ["Reviewers" on page 424](#).

#### Tip:

- **Private projects:** when a project is made private, only the project owners, moderators, and members, plus users with *super*-level privileges in the P4 Server, can see the project, its activity streams, and ongoing reviews. API responses honor these private project restrictions. For more information about private project restrictions, see ["Private projects" on page 366](#).
- **Permissions:** if a public review contains files that the user does not have permission to view, the review is returned but the restricted files are not displayed.

### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>change</code>	Change ID to create a review from	integer	form	Yes
<code>description</code>	Description for the new review (defaults to change description)	string	form	No
<code>reviewers</code>	A list of reviewers for the new review	an array (of strings)	form	No

Parameter	Description	Type	Parameter Type	Required
<code>requiredReviewers</code>	A list of required reviewers for the new review	an array (of strings)	form	No
<code>reviewerGroups</code>	A list of reviewer groups or required reviewer groups for the new review	an array (of strings)	form	No

When a group is a required reviewer, it can be set to operate in one of two ways:

- **Require all:** all members of the group must up-vote the review to allow the review to be approved. To do this, in the `reviewerGroups` parameter for a group, set `"required": "true"`.
- **Require one:** at least one member of the group must up-vote the review to allow the review to be approved. If any member of the group down-votes the review, the review cannot be approved.  
To do this, in the `reviewerGroups` parameter for a group, set `"required": "true"` with `"quorum": "1"`.

### Example usage

Create a review for changelist 12345.

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm-url/api/v10/reviews"
```

The "mybodyfilename.txt" file contains the changelist ID and a description:

```
{
 "change": "12345",
 "description": "This is the review description."
 "reviewers": [
 "raj",
 "mei"
],
 "requiredReviewers": [
 "vera",
 "dai"
],
}
```

```
"reviewerGroups": [
 {"name": "WebDesigners", "required": "true", "quorum": "1"},
 {"name": "Developers", "required": "true"},
 {"name": "Administrators"}
]
}
```

P4 Code Review responds with the new review entity:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "review": [
 {
 "id": 12346,
 "type": "default",
 "changes": [
 12345
],
 "commits": [],
 "author": "bruno",
 "approvals": null,
 "participants": [
 "bruno"
 "dai",
 "mei",
 "raj",
 "swarm-group-Administrators",
 "swarm-group-Developers",
 "swarm-group-WebDesigners",
 "vera"
],
 "participantsData": {
 "bruno": [],
 "dai": {
 "required": true
 },
 "mei": [],
 "raj": [],
 "swarm-group-Administrators": [],
 "swarm-group-Developers": {
 "required": true
 },
 "swarm-group-WebDesigners": {
 "required": "1"
 }
 },
 }
],
 }
}
```

```

 "vera": {
 "required": true
 }
 },
 "hasReviewer": 0,
 "description": "This is the review description.",
 "created": 1689244995,
 "updated": 1689244995,
 "projects": [],
 "state": "needsReview",
 "stateLabel": "Needs Review",
 "testStatus": null,
 "previousTestStatus": "",
 "testDetails": [],
 "deployStatus": null,
 "deployDetails": [],
 "pending": true,
 "commitStatus": [],
 "groups": [],
 "complexity": null,
 "versions": []
}
]
}
}

```

**If a request fails****<error code>:**

- 400 The input specified must be an integer greater than 0
- 401 Unauthorized
- 404 Cannot fetch change. Record does not exist.
- 403 You don't have permission to access the specified change

## Search: P4 Code Review search

### Search Redis cache

**Summary**

Search the Redis cache for users, groups, projects, files, and file content.

GET /api/v10/search?term=<term>&context=user,group,project,filePath,fileContent

**Description**

Search the Redis cache for:

- *userid*s and full names
- *groupid*s and group names
- *projectid*s and project names
- file path
- file content (only if P4 Search is configured for P4 Code Review, see ["Search" on page 684](#))

Results are returned in alphabetical order and the search is case insensitive.

#### Parameters

Parameter	Description	Type	Parameter Type	Required
<code>term</code>	Specifies the characters to search for. Searches are case insensitive.	string	path	Yes

---

Parameter	Description	Type	Parameter Type	Required
context	<p>Limit the search results, valid context values are user, group, project, filePath, and fileContent (see note below). Separate context values with a comma (,).</p> <div><b>Note:</b> If P4 Search is configured for P4 Code Review, see "Search" on page 684:</div>	string	path	Yes

Parameter	Description	Type	Parameter Type	Required
	<div><ul style="list-style-type: none"><li>■ The filePa th search is carried out by P4 Search and not P4 Code Review. This improves the performance of the search request.</li><li>■ The fileCo ntent search is available and carried out by P4 Search.</li></ul></div>			



Parameter	Description	Type	Parameter Type	Required
starts_with_only	<p>Options are:</p> <ul style="list-style-type: none"><li>■ true: searches for users/groups that start with the search term. This results in a faster search than when this parameter is set to false.</li><li>■ false: searches for the term anywhere in users/groups. This is the search used if starts_with_only is not included in the request.</li></ul>	boolean	path	No

Parameter	Description	Type	Parameter Type	Required
limit	<p>Limit the combined maximum number of search results returned for users and groups. Search results are balanced across the contexts requested and will never return more results than the specified limit.</p> <p><b>Examples:</b> based on a search that results in 8 user matches and 30 group matches before the limit is applied:</p> <ul style="list-style-type: none"><li>■ limit=7 and context=user: search result is limited to 7 users.</li><li>■ limit=6 and context=user,group: search result is limited to 3 users and 3 groups. An even limit value splits the search result limit evenly between the specified contexts.</li><li>■ limit=7 and context=user,group: search result is limited to 4 users and 3 groups. An odd limit value splits the</li></ul>	integer	path	No

Parameter	Description	Type	Parameter Type	Required
	<p>search result limit unevenly between the specified contexts with the higher number applied to the first context in the list.</p> <ul style="list-style-type: none"><li>■ limit=20 and context=user,group, search result is limited to 8 users and 12 groups.</li></ul> <p>Search results are not limited when the limit parameter is excluded from the request.</p>			

Parameter	Description	Type	Parameter Type	Required
ignoreExcludeList	<p>Options are:</p> <ul style="list-style-type: none"><li>■ true: the <code>user_exclude_list</code> and <code>group_exclude_list</code> filters are ignored and are not applied to the search results.</li><li>■ false: applies the <code>user_exclude_list</code> and <code>group_exclude_list</code> filters to the search results. This is the search used if <code>ignoreExcludeList</code> is not included in the request.</li></ul>	boolean	path	No

### Example usage

Search users, groups, and projects starting with "jo" with a limit of 4 results returned

Search users and groups starting with "jo" with a limit of 4 results returned.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/search?starts_with_only=true&term=jo&context=user,group,project&limit=4"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "user": {
 "results": [
```

```

 {
 "id": "joakly",
 "name": "Jane Oakly"
 },
 {
 "id": "jsmith",
 "name": "John Smith"
 }
]
},
"group": {
 "results": [
 {
 "id": "jogd",
 "name": "JOGD"
 }
]
},
"project": {
 "results": [
 {
 "id": "johns-project",
 "name": "Johns Project"
 }
]
},
"limit": 4
}
}

```

Search groups for "admin" anywhere in group names and *groupids*

Search for "admin" anywhere in group names and *groupids* with no limit on the number of results returned by the search.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/search?context=group&term=admin"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "group": {
 "results": [
 {
 "id": "swarm-admin",
 "name": "Swarm-admin"
 }
]
 }
 }
}

```

```
 },
 {
 "id": "jplugin-admin",
 "name": "JPlugin-Admin"
 },
 {
 "id": "administrators",
 "name": "Administrators"
 },
 {
 "id": "sysadmins",
 "name": "Sysadmins"
 }
]
}
}
```

**Search files for "myfile" anywhere in file path**

Search for "myfile" anywhere in file path with no limit on the number of results returned by the search.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/search?context=filePath&term=myfile"
```

The response returned depends on whether P4 Search is configured for P4 Code Review or not:

- If P4 Search is not configured for P4 Code Review, P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "filePath": {
 "filesCount": 6,
 "maxScore": 100,
 "results": [
 {
 "type": "file",
 "change": null,
 "depotFile": "//depot/thisIsMyFile.txt",
 "fileName": "thisIsMyFile.txt",
 "action": null,
 "fileType": null,
 "rev": null,
 "fileSize": false
 },

 <results for other files formatted as above>

],
 },
 },
}
```

- If P4 Search is configured for P4 Code Review, P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "filePath": {
 "filesCount": 6,
 "maxScore": 100,
 "results": [
 {
 "type": "revision",
 "change": 10032,
 "depotFile": "//depot/thisIsMyFile.txt",
 "fileName": "thisIsMyFile.txt",
 "action": "branch",
 "fileType": "text",
 "rev": 1,
 "fileSize": 1714
 },

 <results for other files formatted as above>

],
 }
 }
},
```

Search file content for "foobar" anywhere in the file content

**Note:**

The file content search is only available if P4 Search is configured for P4 Code Review, see ["Search" on page 684](#)

Search for "foobar" anywhere in file content with no limit on the number of results returned by the search.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/search?context=fileContent&term=foobar"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "fileContent": {
 "filesCount": 6,
```



```

 "maxScore": null,
 "results": [
 {
 "type": "content",
 "change": 12250,
 "depotFile": "//depot/jim",
 "fileName": "jim",
 "action": 'edit',
 "fileType": 'text',
 "rev": 3,
 "fileSize": 703
 },

 <results for other files formatted as above>

],
 }
}
},

```

**If a request fails**

<error code>:

422 Parameter not specified or invalid, the error message in the response contains the error details.

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Specs: Spec fields

### Get spec information

**Summary**

Get spec file fields.

GET /api/v10/specs/{spec}/ fields

**Description**

Get fields for a spec type.

Parameter	Description	Type	Parameter Type	Required
spec	<p>Spec type. Valid spec types and location in the P4 Server spec depot:</p> <ul style="list-style-type: none"> <li>branch: //spec/branch/</li> <li>change: the change spec is not stored in the P4 Server spec depot</li> <li>client: //spec/client/</li> <li>depot: //spec/depot/</li> <li>group: //spec/group/</li> <li>job: //spec/job/</li> <li>label: //spec/label/</li> <li>license: //spec/license.p4s</li> <li>protect: //spec/protect.p4s</li> <li>spec: //spec/spec/</li> <li>stream: //spec/stream/</li> <li>triggers: //spec/triggers.p4s</li> <li>typemap: //spec/typemap.p4s</li> <li>user: //spec/user/</li> </ul>	string	path	Yes

Parameter	Description	Type	Parameter Type	Required
fields	<p>Omitting the fields parameter or including the fields parameter with no fields specified, returns all of the fields in the spec including any custom fields.</p> <p>Limit the response to specific fields by including a list of comma-separated fields in the request. The fields available depend on the spec type requested. Invalid fields do not cause an error and are ignored.</p>	array of strings	path	No

#### Example usage

##### Fetch all fields for the job spec

Fetching all fields for the job spec

```
curl -u "username:ticket" "https://myswarm-url/api/v10/specs/job/fields"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "job": {
 "Job": {
 "code": "101",
 "dataType": "word",
 "displayLength": "32",
 "fieldType": "required"
 },
 "Status": {
 "code": "102",
 "dataType": "select",
```

```

 "displayLength": "10",
 "fieldType": "required",
 "options": [
 "open",
 "suspended",
 "fixed",
 "closed"
],
 "default": "open"
 },
 "Created_by": {
 "code": "103",
 "dataType": "word",
 "displayLength": "32",
 "fieldType": "once",
 "default": "$user"
 },
 "Description": {
 "code": "105",
 "dataType": "text",
 "displayLength": "0",
 "fieldType": "required",
 "default": "$blank"
 },
 "Date_modified": {
 "code": "130",
 "dataType": "date",
 "displayLength": "20",
 "fieldType": "always",
 "default": "$now"
 }
}
}
}
}

```

#### Limit results to job and status fields for the job spec

Limit the results to the job and status fields for the job spec.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/specs/job/fields?fields=status,job"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "job": {

```

```
"Job": {
 "code": "101",
 "dataType": "word",
 "displayLength": "32",
 "fieldType": "required"
},
"Status": {
 "code": "102",
 "dataType": "select",
 "displayLength": "10",
 "fieldType": "required",
 "options": [
 "open",
 "suspended",
 "fixed",
 "closed"
],
 "default": "open"
}
}
}
```

**Limit results to depot and type fields for the depot spec**

Limit the results to the depot and type fields for the depot spec.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/specs/depot/fields?fields=depot,type"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "depot": {
 "Depot": {
 "code": "251",
 "dataType": "word",
 "displayLength": "32",
 "fieldType": "key"
 },
 "Type": {
 "code": "255",
 "dataType": "word",
 "displayLength": "10",
 "fieldType": "required"
 }
 }
 }
}
```

```

 }
 }
}

```

#### Limit results to status and type fields for the change spec

Limit the results to the status and type fields for the change spec.

```
curl -u "username:ticket" "https://myswarm-url/api/v10/specs/change/fields?fields=status,type"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "change": {
 "Status": {
 "code": "205",
 "dataType": "word",
 "displayLength": "10",
 "fieldType": "always",
 "order": "5",
 "position": "R"
 },
 "Type": {
 "code": "211",
 "dataType": "select",
 "displayLength": "10",
 "fieldType": "optional",
 "order": "6",
 "position": "L",
 "options": [
 "public",
 "restricted"
]
 }
 }
 }
}

```

#### If a request fails

<error code>:

- 400 Unknown spec type requested
- 401 Insufficient permissions to fetch spec fields

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Testdefinitions: P4 Code Review Test definitions

### Get a list of test definitions

#### Summary

Get a list of test definitions and test fields

GET /api/v10/testdefinitions

#### Description

Returns a list of test definitions in P4 Code Review that are visible to the current user.

#### Tip:

Shared test definitions only display the test url, body, headers, and associated fields if you are the test definition owner. By default, this private data is visible in key data for users with list-level access. To hide key data, see ["Hiding P4 Code Review storage from regular users" on page 226](#).

#### Example usage

Get a list of P4 Code Review test definitions:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/testdefinitions"
```

P4 Code Review responds with a list of all of its test definitions:

In the example response below, you own testdefinition id1 but you do not own testdefinition id2. The test url, body, headers, and associated fields are not returned for id2 because you do not own that testdefinition.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
```



```

"data" : {
 testdefinitions: [
 {
 "id": 1,
 "name": "Main code test",
 "headers": {
 "BasicAuth": "YWxpY2U6QUJDRDEyMzQ1Njc4Cg=="
 },
 "encoding": "url",
 "body": "status={status}&change={change}&update={update}",
 "url": "http://jenkins_host:8080/job/{branches}-review-test/review/build",
 "timeout": 20,
 "owners": [
 "allison"
],
 "shared": false,
 "description": "Test for main code",
 "iterate": true
 },
 {
 "id": 2,
 "name": "Doc tests",
 "owners": [
 "bruno"
],
 "shared": true,
 "description": "Tests to check that the docs build.",
 "iterate": false
 }
]
}

```

**If the request fails**

<error code>:

501 Workflows must be enabled for testdefinitions. To enable workflows, see ["workflow" on page 743](#)

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],

```

```
"data" : null
}
```

## Get test definition information

### Summary

Get the details of a test definition

GET /api/v10/testdefinitions/{id}

### Description

Returns the specified test definition if it is visible to the current user.

**Tip:**

Shared test definitions only display the test url, body, headers, and associated fields if you are the test definition owner. By default, this private data is visible in key data for users with list-level access. To hide key data, see ["Hiding P4 Code Review storage from regular users" on page 226](#).

### Parameters

Parameter	Description	Type	Parameter Type	Required
<u>id</u>	Testdefinition ID	integer	path	Yes

### Example usage

#### Get test definition 1:

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/testdefinitions/1"
```

P4 Code Review responds with details for the test definition:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data" : {
 testdefinitions: [
 {
 "id": 1,
```

```

 "name": "Main code test",
 "headers": {
 "BasicAuth": "YWxpY2U6QUJDRDEyMzQ1Njc4Cg=="
 },
 "encoding": "url",
 "body": "status={status}&change={change}&update={update}",
 "url": "http://jenkins_host:8080/job/{branches}-review-test/review/build",
 "timeout": 20,
 "owners": [
 "allison"
],
 "shared": false,
 "description": "Test for main code",
 "iterate": true
 }
]
}
}

```

**If the request fails****<error code>:**

- 404 Test definition does not exist or you do not have permission to view it.
- 501 Workflows must be enabled for testdefinitions. To enable workflows, see ["workflow " on page 743](#)

HTTP/1.1 &lt;response error code&gt;

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Create a test definition

**Summary**

Create a test definition

POST /api/v10/testdefinitions

**Description**

Create a test definition.

## Parameters

Parameter	Description	Type	Parameter Type	Required
name	Provide a name for your test. This should be a unique name that is 1 to 32 characters in length.	string	body	Yes
description	Provide a description for the test.	string	body	No
headers	Specify name=value pairs to pass to the test suite.	array of strings	body	No
encoding	<p>Specify the encoding method for the request body parameter:</p> <ul style="list-style-type: none"><li>■ url for URL Encoded: POST parameters are parsed into name=value pairs. This is the default.</li><li>■ json for JSON Encoded: parameters are passed raw in the POST body.</li><li>■ xml for XML Encoded: parameters are passed in XML format in the POST body.</li></ul>	string	body	No (required if body parameter used)

Parameter	Description	Type	Parameter Type	Required
body	Specify parameters that your test suite requires in the request body parameter. Encoding is set with the encoding parameter.	string	body	No

---

Parameter	Description	Type	Parameter Type	Required
url	<p>Specify the test request URL that will trigger your test suite using the url parameter. 1 to 1024 characters.</p> <p>Special arguments are also available to pass details from P4 Code Review to your test suite, see <a href="#">"Pass special arguments to your test suite"</a> on page 531.</p>	string	body	Yes
<div><b>Note:</b> <b>P4 for Jenkins 1.10.11 and later:</b> P4 Code Review must send the parameters for the build to Jenkins as a POST request. To do this, specify the parameters in the body and specify url for encoding.</div>				

---

Parameter	Description	Type	Parameter Type	Required
timeout	Specify a timeout in seconds. The timeout is used when P4 Code Review connects to your test suite. If a timeout is not set, the <a href="#">http_client_options</a> timeout setting is used. By default, this is 10 seconds.	integer	body	No
owners	Specify at least one owner for the test.	array of strings	body	Yes
shared	<p>Specify if want other P4 Code Review users to be able to view or use this test definition:</p> <ul style="list-style-type: none"><li>■ true: other P4 Code Review users can use this test.</li><li>■ false or not or not specified: other P4 Code Review users will not be able to see or use this test.</li></ul>	boolean	body	No

Parameter	Description	Type	Parameter Type	Required
iterate	<p>Specify if you want P4 Code Review to create a separate test run for each branch and project the review is associated with:</p> <ul style="list-style-type: none"><li>■ <b>true</b>: creates a separate test run for each branch and project the review is associated with. The details generated by the {projects} and {branches} arguments in the URL or Body are used in the test name displayed in the <a href="#">"Tests" on page 428</a> dropdown on the review page.</li><li>■ <b>false</b> or not or not specified: only one test will be run for the review, even if multiple branches and projects are associated with the review.</li></ul>	boolean	body	No

---



## Example usage

### Create a test definition

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm.url/api/v10/testdefinitions"
```

The "mybodyfilename.txt" file contains the test definition details:

```
{
 "name": "My test def",
 "description": "My test definition description",
 "headers": {
 "BasicAuth": "user:password",
 "OtherHeader": "OtherValue"
 },
 "encoding": "json",
 "body": "{\"test\":\"{test}\"}",
 "url": "http://my-test-host/api/v10/testruns/test?test={test}",
 "timeout": 10,
 "owners": ["bruno"],
 "shared": true,
 "iterate": true
}
```

P4 Code Review responds with the testdefinition details:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "testdefinitions": [
 {
 "id": 3,
 "name": "My test def",
 "headers": {
 "BasicAuth": "user:password",
 "OtherHeader": "OtherValue"
 },
 "encoding": "json",
 "body": "{\"test\":\"{test}\"}",
 "url": "http://my-test-host/api/v10/testruns/test?test={test}",
 "timeout": 10,
 "owners": [
 "bruno"
],
 "shared": true,
 "description": "My test definition description",

```

```

 "iterate": true
 }
]
}
}

```

#### If the request fails

<error code>:

400 one of the following:

- The test definition `name` parameter is required and must be unique.
- The test definition `url` and `owners` parameters are required and cannot be empty.
- The `encoding`, `shared`, `description`, and `iterate` parameters require a value and cannot be empty.

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Testruns: P4 Code Review Test Integration

### Get testrun details of a review version

#### Summary

Get the testrun details of a review version.

GET /api/v10/reviews/{reviewId}/testruns

#### Description

Used to get the testrun details for a review version.

#### Note:

Any authenticated user can get test run details for a review version. The test run uuid is not included in the request response.

## Parameters

Parameter	Description	Type	Parameter Type	Required
<code>reviewId</code>	Review ID	integer	path	Yes
<code>version</code>	An optional parameter that enables you to specify the review version you want to return the testrun details for. Omitting this parameter or passing an empty value shows testrun details for the latest version of the review.	string	query	No

## Example of usage

Get details for all of the test runs for the latest version of review 12345

```
curl "https://myswarm.url/api/v10/reviews/12345/testruns"
```

P4 Code Review responds with the testrun details:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "testruns": [
 {
 "id": 706,
 "change": 12345,
 "version": 2,
 "test": "global1",
 "startTime": 1567895432,
 "completedTime": 1567895562,
```

```

 "status": "pass",
 "messages": [
 "Test completed successfully",
 "another message"
],
 "url": "http://my.jenkins.com/projectx/main/1224"
 },
 {
 <ids for other testruns of the review, formatted as above>
 }
]
}
}

```

**Get details for all of the test runs for version 2 of review 12345**

```
curl "https://myswarm.url/api/v10/reviews/12345/testruns?version=2"
```

P4 Code Review responds with the testrun details:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": [],
 "data": {
 "testruns": [
 {
 "id": 706,
 "change": 12345,
 "version": 2,
 "test": "global1",
 "startTime": 1567895432,
 "completedTime": 1567895562,
 "status": "pass",
 "messages": [
 "Test completed successfully",
 "another message"
],
 "url": "http://my.jenkins.com/projectx/main/1224"
 },
 {
 <ids for other testruns for version 2 of the review, formatted as above>
 }
]
 },
 "status": "pass"
}

```

**Tip:**

When you make a GET request for all of the testruns in a specific review version, P4 Code Review includes a "status" for the review version as a whole in the response. This is shown after the individual testruns. This status is calculated from the "status" values of all of the individual testruns for that review version:

- If the test "status" for any of the testruns is "running" or if any **On Demand** tests have not been run, the overall result for that version is "running".
- If no testruns are "running" and the test "status" of any of the testruns is "fail" the overall result for that version is "fail".
- If the test "status" of all of the testruns is "pass" then the overall result for that version is "pass".

**If a request fails**

HTTP/1.1 <response error code>

```
{
 "error": <high level description>,
 "messages": [{"code" : "<code string>", "msg" : "<error message>"}],
 "data" : null
}
```

## Create a testrun for a review version

**Summary**

Create a testrun for a review

POST /api/v10/reviews/{reviewid}/testruns

**Description**

Create a testrun for a review version. After you have created a testrun for a version of a review, you can update the testrun details as the test progresses using the [PATCH](#) and [PUT](#) API endpoints or the {pass}, {fail} and {update} callback urls.

**Important:**

Only *admin* users can create a test run for a review.

## Parameters

Parameter	Description	Type	Parameter Type	Required	Restriction
reviewid	Review ID.	integer	path	Yes	
version	The version of the review the testrun is being run against.	integer	body	Yes	
test	Specifies the test used for the testrun and checks that it is a valid name.	string	body	Yes	1 to 32 characters
startTime	The time the testrun started, expressed as an epoch value.	integer	body	Yes	> 0
completedTime	The time the testrun completed, expressed as an epoch value.	integer	body	No	> 0
status	The status of the testrun, options are: pass, fail, and running	string	body	No	Valid options are pass, fail, and running

Parameter	Description	Type	Parameter Type	Required	Restriction
messages	An array of one or more messages for the testrun. They should be formatted in JSON in the body of the POST request. You can pass a maximum 10 messages, if you provide more than 10 messages only the first 10 are saved. Each message can contain a maximum of 80 characters, any messages with more than 80 characters will be automatically truncated.	array	body	No	

Parameter	Description	Type	Parameter Type	Required	Restriction
url	The CI system url for the testrun, this is the link to the CI that enables you to view the current testrun results.	string	body	No	1 to 1024 characters
uuid	The uuid for the testrun. Used to enable non- <i>admin</i> users to update the testrun. If the requested uuid does not match the review an error message is returned. You must include the review version number at the end of the uuid. For example, to get version 2 of the review, append .v2 to the end of the uuid.	string	body	No	Minimum 32 max 64 characters



## Example usage

### Append a new testrun to version 2 of review 12345

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm.url/api/v10/reviews/12345/testruns"
```

The "mybodyfilename.txt" file contains the testrun details:

```
{
 "change": 12345,
 "version": 2,
 "test": "mytest",
 "startTime": 1567895432,
 "status": "running",
 "messages": [
 "Mytest running",
 "another message"
]
 "url": "http://my.jenkins.com/projectx/main/1224",
 "uuid": "FAE4501C-E4BC-73E4-A11A-FF710601BC3F.v2"
}
```

P4 Code Review responds with the testrun details:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "testruns": [
 {
 "id": 706,
 "change": 12345,
 "version": 2,
 "test": "mytest",
 "startTime": 1567895432,
 "completedTime": null,
 "status": "running",
 "messages": [
 "Mytest running",
 "another message"
],
 "url": "http://my.jenkins.com/projectx/main/1224"
 }
]
 }
}
```

**If the request fails**

HTTP/1.1 <response error code>

```
{
 "error": <high level description>,
 "messages": [{"code" : "<code string>", "msg" : "<error message>"}],
 "data" : null
}
```

## Create a testrun for a review version using a specified test

**Summary**

Create a testrun for a review using a specified test

POST /api/v10/reviews/{reviewid}/testruns/{test\_id}

**Description**

Create a testrun for a review version using a specified test. Tests are configured from the P4 Code Review **Tests** page, see ["Tests" on page 528](#). After you have created a testrun for a version of a review, you can update the testrun details as the test progresses using the [PATCH](#) and [PUT](#) API endpoints or the {pass}, {fail} and {update} callback urls.

**Important:**

Only *admin* users can create a test run for a review.

**Note:**

The test\_id parameter specified must exist in P4 Code Review, if it doesn't exist an error is generated when the API request is made.

**Parameters**

Parameter	Description	Type	Parameter Type	Required	Restriction
reviewid	Review ID.	integer	path	Yes	

Parameter	Description	Type	Parameter Type	Required	Restriction
version	The version of the review the testrun is being run against.	integer	body	Yes	
test_id	<p>Specifies the test_id used for the testrun and checks that it is a valid test id. If there is a test name in the body of the request, P4 Code Review will ignore it.</p> <p>If the test_id parameter is omitted, P4 Code Review uses the test name in the body of the request.</p>	string	path	Yes	1 to 32 characters
startTime	The time the testrun started, expressed as an epoch value.	integer	body	Yes	> 0

Parameter	Description	Type	Parameter Type	Required	Restriction
completedTime	The time the testrun completed, expressed as an epoch value.	integer	body	No	> 0
status	The status of the testrun, options are: pass, fail, and running.	string	body	No	Valid options are pass, fail, and running

Parameter	Description	Type	Parameter Type	Required	Restriction
messages	An array of one or more messages for the testrun. They should be formatted in JSON in the body of the POST request. You can pass a maximum 10 messages, if you provide more than 10 messages only the first 10 are saved. Each message can contain a maximum of 80 characters, any messages with more than 80 characters will be automatically truncated.	array	body	No	

Parameter	Description	Type	Parameter Type	Required	Restriction
url	The CI system url for the testrun, this is the link to the CI that enables you to view the current testrun results.	string	body	No	1 to 1024 characters
uuid	The uuid for the testrun. Used to enable non- <i>admin</i> users to update the testrun. If the requested uuid does not match the review an error message is returned. You must include the review version number at the end of the uuid. For example, to get version 2 of the review, append .v2 to the end of the uuid.	string	body	No	Minimum 32 max 64 characters

## Example usage

Append a testrun for a review using test id 5 for version 3 of review 12345

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm.url/api/v10/reviews/12345/testruns/5"
```

The "mybodyfilename.txt" file contains the testrun details:

```
{
 "change": 12345,
 "version": 3,
 "startTime": 1567895432,
 "status": "running",
 "messages": [
 "Smoke-test running",
 "another message"
],
 "url": "http://my.jenkins.com/projecty/main/1224",
 "uuid": "FAE4501C-E4BC-73E4-A11A-FF710601BC4F.v2"
}
```

P4 Code Review responds with the testrun details:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "testruns": [
 {
 "id": 5,
 "change": 12345,
 "version": 3,
 "test": "test01",
 "startTime": 1567895432,
 "completedTime": null,
 "status": "running",
 "messages": [
 "Smoke-test running",
 "another message"
],
 "url": "http://my.jenkins.com/projecty/main/1224"
 }
]
 }
}
```

**If the request fails**

HTTP/1.1 <response error code>

```
{
 "error": <high level description>,
 "messages": [{"code" : "<code string>", "msg" : "<error message>"}],
 "data" : null
}
```

## Update details for a testrun - PATCH

**Important:**

Only *admin* users can update testrun details for a review.

**Summary**

Update details for a testrun

PATCH /api/v10/reviews/{reviewid}/testruns/{id}

**Description**

Update the details for a testrun.

Here are a few example scenarios related to depot permissions, private projects, and testrun API endpoint.

- If a user is a private project member with access to the depot and has "write" permissions, they can view the review in P4 Code Review. However, the user does not have permissions to run the testRun API.
- If a user is not a member of a private project but can access the depot, and if the user has "write" permissions for the depot, the user will not be able to view the review in P4 Code Review or run the testRun API.
- If a user is not a private project member but has "admin" permissions for the P4 Server while using the testRun API, they can view the review in P4 Code Review, update the test status, and test description.
- If a user is not a private project member but has "write" permissions for the P4 Server, the user cannot see the review in P4 Code Review and they cannot run the testRun API.



## Parameters

Parameter	Description	Type	Parameter Type	Required	Restriction
id	Testrun ID, identifies a specific testrun for the review. Automatically generated by P4 Code Review when the testrun is created.	integer	path	Yes	
reviewid	Review ID.	integer	path	Yes	
version	The version of the review the testrun is being run against.	integer	body	No	
test	Specifies the test used for the testrun and checks that it is a valid name.	string	body	No	1 to 32 characters
startTime	The time the testrun started, expressed as an epoch value.	integer	body	No	> 0

Parameter	Description	Type	Parameter Type	Required	Restriction
completedTime	The time the testrun completed, expressed as an epoch value.	integer	body	No	> 0
status	The status of the testrun, options are: pass, fail, and running	string	body	No	Valid options are pass, fail, and running

Parameter	Description	Type	Parameter Type	Required	Restriction
messages	An array of one or more messages for the testrun. They should be formatted in JSON in the body of the POST request. You can pass a maximum 10 messages, if you provide more than 10 messages only the first 10 are saved. Each message can contain a maximum of 80 characters, any messages with more than 80 characters will be automatically truncated.	array	body	No	

Parameter	Description	Type	Parameter Type	Required	Restriction
url	The CI system url for the testrun, this is the link to the CI that enables you to view the current testrun results.	string	body	No	1 to 1024 characters
uuid	The uuid for the testrun. Used to enable non- <i>admin</i> users to update the testrun. If the requested uuid does not match the review an error message is returned. You must include the review version number at the end of the uuid. For example, to get version 2 of the review, append .v2 to the end of the uuid.	string	body	No	Minimum 32 max 64 characters

## Example usage

### Update testrun 706 for review 12345

```
curl -X PATCH -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://myswarm.url/api/v10/reviews/12345/testruns/706"
```

The "mybodyfilename.txt" file contains the testrun details you want to update.

```
{
 "completedTime": "1567895562",
 "status": "fail",
 "messages": [
 "Test has failed",
 "yet another message"
]
}
```

P4 Code Review responds with the testrun details:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "testruns": [
 {
 "id": 706,
 "change": 12345,
 "version": 2,
 "test": "mytest",
 "startTime": 1567895432,
 "completedTime": 1567895562,
 "status": "fail",
 "messages": [
 "Test has failed",
 "yet another message"
],
 "url": "http://my.jenkins.com/projectx/main/1224"
 }
]
 }
}
```

### If the request fails

HTTP/1.1 <response error code>

```
{
 "error": <high level description>,
 "messages": [{"code" : "<code string>", "msg" : "<error message>"}],
 "data" : null
}
```

## Update details for a testrun - PUT

### Summary

Update details for a testrun

PUT /api/v10/reviews/{reviewid}/testruns/{id}

### Description

Update the details for a testrun. All values must be provided in the request.

#### Important:

Only *admin* users can update testrun details for a review.

### Parameters

Parameter	Description	Type	Parameter Type	Required	Restriction
id	Testrun ID, identifies a specific testrun for the review. Automatically generated by P4 Code Review when the testrun is created.	integer	path	Yes	
reviewid	Review ID.	integer	path	Yes	

Parameter	Description	Type	Parameter Type	Required	Restriction
version	The version of the review the testrun is being run against.	integer	body	Yes	
test	Specifies the test used for the testrun and checks that it is a valid name.	string	body	Yes	1 to 32 characters
startTime	The time the testrun started, expressed as an epoch value.	integer	body	Yes	> 0
completedTime	The time the testrun completed, expressed as an epoch value.	integer	body	Yes	> 0
status	The status of the testrun, options are: pass, fail, and running	string	body	Yes	Valid options are pass, fail, and running

Parameter	Description	Type	Parameter Type	Required	Restriction
messages	An array of one or more messages for the testrun. They should be formatted in JSON in the body of the POST request. You can pass a maximum 10 messages, if you provide more than 10 messages only the first 10 are saved. Each message can contain a maximum of 80 characters, any messages with more than 80 characters will be automatically truncated.	array	body	Yes	

---



Parameter	Description	Type	Parameter Type	Required	Restriction
url	The CI system url for the testrun, this is the link to the CI that enables you to view the current testrun results.	string	body	Yes	1 to 1024 characters
uuid	The uuid for the testrun. Used to enable non- <i>admin</i> users to update the testrun. If the requested uuid does not match the review an error message is returned. You must include the review version number at the end of the uuid. For example, to get version 2 of the review, append .v2 to the end of the uuid.	string	body	Yes	Minimum 32 max 64 characters

## Example usage

### Update testrun 706 for review 12345

```
curl -X PUT -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm.url/api/v10/reviews/12345/testruns/706"
```

The "mybodyfilename.txt" file must contain all of the testrun details:

```
{
 "id": 706,
 "change": 12345,
 "version": 2,
 "test": "mytest",
 "startTime": 1567895432,
 "completedTime": 1567895562,
 "status": "fail",
 "messages": [
 "Test has failed",
 "and another message"
],
 "url": "http://my.jenkins.com/projectx/main/1224",
 "uuid": "FAE4501C-E4BC-73E4-A11A-FF710601BC3F.v2"
}
```

P4 Code Review responds with the testrun details:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "testruns": [
 {
 "id": 706,
 "change": 12345,
 "version": 2,
 "test": "mytest",
 "startTime": 1567895432,
 "completedTime": 1567895562,
 "status": "fail",
 "messages": [
 "Test has failed",
 "and another message"
],
 "url": "http://my.jenkins.com/projectx/main/1224"
 }
]
 }
}
```

```
}
}
```

**If the request fails**

HTTP/1.1 <response error code>

```
{
 "error": <high level description>,
 "messages": [{"code" : "<code string>", "msg" : "<error message>"}],
 "data" : null
}
```

## Rerun a specific test

**Summary**

Rerun a specific test.

POST /api/v10/reviews/{reviewid}/testruns/{testrunid}/run

**Description**

Used to rerun a specific test.

**Example usage**

Rerun testrun 706 for review 12345

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket"
"https://myswarm.url/api/v10/reviews/12345/testruns/706/run"
```

P4 Code Review responds with the testrun details:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "testruns": [
 {
 "id": 706,
 "change": 12345,
 "version": 2,
 "test": "mytest",
 "startTime": 1567895432,
 "completedTime": 1567895562,

```

```
 "status": "running",
 "messages": null,
 "url": "http://my.jenkins.com/projecty/main/1224"
 "title": "Swarm-main-codeSniff"
 }
]
}
}
```

#### If the request fails

HTTP/1.1 <response error code>

```
{
 "error": <high level description>,
 "messages": [{"code" : "<code string>", "msg" : "<error message>"}],
 "data" : null
}
```

## Users: P4 Code Review users

### Get a list of users

#### Summary

Get a list of users.

GET /api/v10/users

#### Description

Returns a list of users in P4 Code Review.

## Parameters

Parameter	Description	Type	Parameter Type	Required
ids	An optional single string or an array of users to display. Omitting this parameter or passing an empty value shows all users.	string	query	No
ignoreExcludeList	Determines if the list of users has the <code>user_exclude_list</code> filter applied or not. Add the parameter to ignore the <code>user_exclude_list</code> filter.	boolean	query	No

## Example usage

### Get a list of users

```
curl -u "username:ticket" "https://myswarm-url/api/v10/users"
```

P4 Code Review responds with all users in P4 Code Review:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "users": [
 {
 "id": "bruno",
 "type": "standard",
 "email": "bruno@perforce.com",
 "update": "2013/10/11 21:05:15",
```

```
 "access": "2021/01/21 16:55:38",
 "fullName": "Bruno First",
 "jobView": null,
 "password": null,
 "authMethod": "perforce",
 "reviews": []
 },
 ...
 <other users formatted as above>
 ...
}
]
}
}
```

#### Fetch a specific user

To fetch details for bruno:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/users?ids=bruno"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "users": [
 {
 "id": "bruno",
 "type": "standard",
 "email": "bruno@perforce.com",
 "update": "2013/10/11 21:05:15",
 "access": "2021/01/21 16:55:38",
 "fullName": "Bruno First",
 "jobView": null,
 "password": null,
 "authMethod": "perforce",
 "reviews": []
 }
]
 }
}
```

#### Fetch a number of specific users

To fetch details for bruno and alice:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/users?ids[]=bruno&ids[]=alice"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "users": [
 {
 "id": "bruno",
 "type": "standard",
 "email": "bruno@perforce.com",
 "update": "2013/10/11 21:05:15",
 "access": "2021/01/21 16:55:38",
 "fullName": "Bruno First",
 "jobView": null,
 "password": null,
 "authMethod": "perforce",
 "reviews": []
 },
 {
 "user": "alice",
 "type": "standard",
 "email": "asmith@perforce.com",
 "update": "2018/09/11 18:17:14",
 "access": "2020/12/24 16:55:38",
 "fullName": "Alice Smith",
 "jobView": null,
 "password": null,
 "authMethod": "perforce",
 "reviews": []
 }
]
 }
}
```

If a request fails

<error code>:

**401** Incorrect or missing credentials

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",
```

```
"text" : "<error message>"
}},
"data" : null
}
```

## Get information for a specific user

### Summary

Get information for a specific user.

GET /api/v10/users/{id}

### Description

Returns information for a specific user in P4 Code Review.

### Parameters

Parameter	Description	Type	Parameter Type	Required
<u>id</u>	User ID	string	path	Yes

### Example usage

#### Get information for a specific user

To fetch information for bruno:

```
curl -u "username:ticket" "https://myswarm-url/api/v10/users/bruno"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "users": [
 {
 "id": "bruno",
 "type": "standard",
 "email": "bruno@perforce.com",
 "update": "2013/10/11 21:05:15",
```



```

 "access": "2021/01/21 16:55:38",
 "fullName": "Bruno First",
 "jobView": null,
 "password": null,
 "authMethod": "perforce",
 "reviews": []
 }
]
}
}

```

#### If a request fails

<error code>:

- **401** Incorrect or missing credentials
- **404** User does not exist

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code": "<code string>",
 "text": "<error message>"
 }],
 "data": null
}

```

## Check for user deletion or add user for deletion in P4 Code Review

### Summary

Check if a user has been successfully deleted from the user profile, test definitions, workflows, projects, or groups.

POST /api/v10/users/{id}/cleanup

### Description

When a user is deleted from the P4 Server, P4 Code Review automatically removes the deleted user from the user configuration, test definitions, workflows, projects, or groups. An admin user can use this API endpoint to determine if a user is successfully deleted from P4 Code Review or if a user is not deleted, P4 Code Review adds the cleanup action for that user in the queue . For more information about how Swarm handles deleted users, see ["How P4 Code Review handles deleted users " on page 736](#).

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	User ID	string	path	Yes

**Example usage****Check if a user has been successfully deleted across all entities in P4 Code Review**

To check if user bruno is successfully deleted from P4 Code Review:

```
curl -u "username:ticket" -X POST "https://myswarm-url/api/v10/users/bruno/cleanup"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": {
 "code": 200,
 "text": {
 "testdefinition": "The requested user bruno does not exist in any testdefinition",
 "workflow": "The requested user bruno does not exist in any workflow",
 "project": "Your request to clean up project for the deleted user bruno has been queued",
 "group": "Your request to clean up group for the deleted user bruno has been queued",
 "followers": "Your request to clean up followers for the deleted user bruno has been
queued"
 }
 },
 "data": []
}
```

**Check if a user has been successfully deleted from all test definitions in P4 Code Review**

To check if user bruno is successfully deleted from all test definitions in P4 Code Review:

```
curl -u "username:ticket" -X POST "https://myswarm-url/api/v10/users/bruno/cleanup/testdefinition"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": {
```

```

 "code": 200,
 "text": "The requested user bruno does not exist in any test definition"
 },
 "data": []
}

```

#### Check if a user has been successfully deleted from all workflows in P4 Code Review

To check if user bruno is successfully deleted from all workflows in P4 Code Review:

```
curl -u "username:ticket" -X POST "https://myswarm-url/api/v10/users/bruno/cleanup/workflow"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": {
 "code": 200,
 "text": "Your request to clean up workflow for the deleted user bruno has been queued"
 },
 "data": []
}

```

#### Check if a user has been successfully deleted from all projects in P4 Code Review

To check if user bruno is successfully deleted from all projects in P4 Code Review:

```
curl -u "username:ticket" -X POST "https://myswarm-url/api/v10/users/bruno/cleanup/projects"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```

{
 "error": null,
 "messages": {
 "code": 200,
 "text": "The requested user bruno does not exist in any project"
 },
 "data": []
}

```

#### Check if a user has been successfully deleted from all groups in P4 Code Review

To check if user bruno is successfully deleted from all groups in P4 Code Review:

```
curl -u "username:ticket" -X POST "https://myswarm-url/api/v10/users/bruno/cleanup/groups"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": {
 "code": 200,
 "text": "Your request to clean up group for the deleted user bruno has been queued"
 },
 "data": []
}
```

#### Check if a user has been successfully deleted from the user configuration in P4 Code Review

To check if user bruno is successfully deleted from the user configuration in P4 Code Review:

```
curl -u "username:ticket" -X POST "https://myswarm-url/api/v10/users/bruno/cleanup/config"
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": {
 "code": 200,
 "text": "The requested user bruno does not exist in any config"
 },
 "data": []
}
```

#### If a request fails

<error code>:

- **400** User is active and not yet deleted

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Workflows : P4 Code Review workflows

### Get workflows

#### Summary

Gets workflows

GET /api/v10/workflows/

#### Description

Get all workflows

#### Parameters

Parameter	Description	Type	Parameter Type	Required
fields	An optional comma-separated list (or array) of fields to show for each workflow. Omitting this parameter or passing an empty value shows all fields.	string	query	No
noCache	If provided and has a value of 'true' a query will always be performed and the cache of workflows is ignored. Otherwise the cache will be used if it exists.	boolean	query	No

## Usage example

### Get a list of workflows

```
curl -u "username:ticket" "https://myswarm-url/api/v10/workflows"
```

JSON Response:

**Tip:**

The global workflow has an id of 0.

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "workflows": [
 {
 "on_submit": {
 "with_review": {
 "rule": "no_checking",
 "mode": "default"
 },
 "without_review": {
 "rule": "no_checking",
 "mode": "policy"
 }
 },
 "name": "Global Workflow",
 "description": "This is the Global workflow",
 "shared": "false",
 "owners": [
 "swarm"
],
 "end_rules": {
 "update": {
 "rule": "no_checking",
 "mode": "default"
 }
 },
 "auto_approve": {
 "rule": "never",
 "mode": "default"
 },
 "counted_votes": {
 "rule": "anyone",
 "mode": "default"
 }
 },
]
 }
}
```

```

 "group_exclusions":{
 "rule": [],
 "mode": "policy"
 },
 "user_exclusions":{
 "rule": [],
 "mode": "policy"
 },
 "tests": [],
 "id": "0"
 },
 {
 "on_submit":{
 "with_review":{
 "rule": "no_checking",
 "mode": "inherit"
 },
 "without_review":{
 "rule": "no_checking",
 "mode": "inherit"
 }
 },
 "name": "myWorkflow 1",
 "description": "Another description",
 "shared": "true",
 "owners": [
 "user3",
 "user4"
],
 "end_rules":{
 "update":{
 "rule":"no_revision",
 "mode": "inherit"
 }
 },
 "auto_approve":{
 "rule": "votes",
 "mode": "inherit"
 },
 "counted_votes": {
 "rule": "members",
 "mode": "inherit"
 },
 "tests": [],
 "id": "1"
 },
]
}

```

**If a request fails**

<error code>:

**501** Workflows are not enabled. To enable workflows, see "[workflow](#) " on page 743

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get a workflow by id

**Summary**

Gets a workflow by id

GET /api/v10/workflows/{id}

**Description**

Gets a workflow by id. The global workflow id is 0.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
<code>fields</code>	An optional comma-separated list (or array) of fields to show for each workflow. Omitting this parameter or passing an empty value shows all fields.	string	query	No

---



## Example usage

### Get a workflows by id

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/workflows/1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "workflows": [
 {
 "on_submit": {
 "with_review": {
 "rule": "no_checking",
 "mode": "inherit"
 },
 "without_review": {
 "rule": "no_checking",
 "mode": "inherit"
 }
 },
 "name": "myWorkflow 1",
 "description": "Another description",
 "shared": "true",
 "owners": [
 "user3",
 "user4"
],
 "end_rules": {
 "update": {
 "rule": "no_revision",
 "mode": "inherit"
 }
 },
 "auto_approve": {
 "rule": "votes",
 "mode": "inherit"
 },
 "counted_votes": {
 "rule": "members",
 "mode": "inherit"
 },
 "tests": [],
 "id": "1"
 }
],
 },
}
```

```
]
}
```

**If a request fails****<error code>:**

- **401** Insufficient permissions to view the workflow
- **404** Workflow does not exist
- **501** Workflows are not enabled. To enable workflows, see ["workflow " on page 743](#)

HTTP/1.1 &lt;response error code&gt;

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Get workflow details by workflow name or test id

**Summary**

Gets workflow details by workflow name or for workflows associated with a test id

GET /api/v10/workflows

**Description**

Gets workflow details by workflow name or for workflows associated with a test id. You can enter multiple workflow names or test ids in the request if required.

**Tip:**

If the name and testdefinitions parameters are both in the request, only name is used for the request and the testdefinitions parameter is ignored.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
name	Specify the workflow you want to return details for with the workflow name parameter. To request details for multiple workflows, enter the workflow names as a comma separated list.	string	query	No
testdefinitions	Get details of workflows associated with a test id with the testdefinitions parameter. To request multiple test ids, enter the test ids as a comma separated list.	string	query	No

**Example usage****Get details for myWorkflow 1**

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/workflows?name=myWorkflow 1"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
```

```
"workflows": [
 {
 "on_submit":{
 "with_review":{
 "rule": "no_checking",
 "mode": "inherit"
 },
 "without_review":{
 "rule": "no_checking",
 "mode": "inherit"
 }
 },
 "name": "myWorkflow 1",
 "description": "Another description",
 "shared": "true",
 "owners": [
 "user3",
 "user4"
],
 "end_rules":{
 "update":{
 "rule":"no_revision",
 "mode": "inherit"
 }
 },
 "auto_approve":{
 "rule": "votes",
 "mode": "inherit"
 },
 "counted_votes": {
 "rule": "members",
 "mode": "inherit"
 },
 "tests": [],
 "id": "1"
 },
]
```

Get details for workflows associated with test id 1 or test id 2

```
curl -u "username:ticket" "https://my-swarm-host/api/v10/workflows?testdefinitions=1,2"
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "error": null,
```

```

"messages": [],
"data": {
 "workflows": [
 {
 "on_submit":{
 "with_review":{
 "rule": "no_checking",
 "mode": "default"
 },
 "without_review":{
 "rule": "no_checking",
 "mode": "policy"
 }
 },
 "name": "Our Workflow",
 "description": "This is the our workflow and it is good",
 "shared": "true",
 "owners": [
 "swarm"
],
 "end_rules":{
 "update":{
 "rule": "no_checking",
 "mode": "default"
 }
 },
 "auto_approve":{
 "rule": "never",
 "mode": "default"
 },
 "counted_votes":{
 "rule": "anyone",
 "mode": "default"
 },
 "group_exclusions":{
 "rule": [],
 "mode": "policy"
 },
 "user_exclusions":{
 "rule": [],
 "mode": "policy"
 },
 "tests": [
 {
 "id": 1,
 "event": "onUpdate"
 "blocks": "approved"
 },

```

```
{
 "id": 2,
 "event": "onSubmit"
 "blocks": "nothing"
}
],
"id": "3"
},
{
 "on_submit":{
 "with_review":{
 "rule": "no_checking",
 "mode": "inherit"
 },
 "without_review":{
 "rule": "no_checking",
 "mode": "inherit"
 }
 },
 "name": "Another Workflow",
 "description": "Yet another workflow description",
 "shared": "false",
 "owners": [
 "user3",
 "user4"
],
 "end_rules":{
 "update":{
 "rule": "no_revision",
 "mode": "inherit"
 }
 },
 "auto_approve":{
 "rule": "votes",
 "mode": "inherit"
 },
 "counted_votes": {
 "rule": "members",
 "mode": "inherit"
 },
 "tests": [
 {
 "id": 2,
 "event": "onSubmit"
 "blocks": "approved"
 },
 {
 "id": 5,
```

```

 "event": "onSubmit"
 "blocks": "approved"
 }
],
 "id": "6"
 },
]
}

```

**If a request fails****<error code>:**

- **401** Insufficient permissions to view the workflow
- **404** Workflow or id does not exist
- **501** Workflows are not enabled. To enable workflows, see ["workflow " on page 743](#)

HTTP/1.1 &lt;response error code&gt;

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Create a workflow

**Summary**

Create a workflow

POST /api/v10/workflows/

**Description**

Create a new workflow.

**Tip:**

The global workflow is present by default so it can only be edited.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
name	The workflow name. Will be compared against other workflows and rejected if not unique.	string	body	Yes
description	Description for the new workflow.	string	body	Yes
shared	Whether this workflow is shared for other users that do not own it. Defaults to not shared.	boolean	body	Yes
owners	A list owners for the workflow. Can be users or group names (prefixed with swarm-group-). Users and group names must exist or the workflow will be rejected.	array (of strings)	body	Yes
mode	Each workflow rule must have a mode parameter. The only valid value for mode is inherit.	string	body	Yes



Parameter	Description	Type	Parameter Type	Required
on_submit	Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without_review are no_checking, auto_create, reject.	array	body	Yes
end_rules	Data for rules when changes are submitted. Valid values are no_checking, no_revision.	array	body	Yes
auto_approve	Data for rules when changes are submitted. Valid values are votes, never.	array	body	Yes
counted_votes	Data for rules when counting votes up. Valid values are anyone, members.	array	body	Yes

Parameter	Description	Type	Parameter Type	Required
tests	<p>A list of test ids for the workflow. Each test id has an event that determines when the test is run and a blocks value that determines whether a test failure blocks review approval.</p> <ul style="list-style-type: none"><li>Valid values for event are onUpdate, onSubmit.</li><li>Valid values for blocks are nothing and approved.</li></ul>	array	body	No

### Example usage

#### Create a workflow

```
curl -X POST -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt" "https://myswarm-url/api/v10/workflows/"
```

The "mybodyfilename.txt" file contains the information for the new workflow:

```
{
 "name": "My workflow"
 "description": "This is my workflow."
 "shared": "true"
 "owners": "Francois_Piccard,Anna_Schmidt"
 "on_submit":{
 "with_review":{
 "rule":"no_checking",
 "mode": "inherit"
 },

```

```
 "without_review":{
 "rule":"no_checking",
 "mode": "inherit"
 }
 },
 "end_rules": {
 "update":{
 "rule":"no_revision",
 "mode": "inherit"
 }
 },
 "auto_approve":{
 "rule": "never",
 "mode": "inherit"
 },
 "counted_votes": {
 "rule": "members",
 "mode": "inherit"
 },
 "tests": [
 {
 "id": 2,
 "event": "onUpdate"
 "blocks": "nothing"
 },
 {
 "id": 5,
 "event": "onSubmit"
 "blocks": "approved"
 }
]
}
```

P4 Code Review responds with:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "workflows": [
 {
 "on_submit":{
 "with_review":{
 "rule": "no_checking",
 "mode": "inherit"
 },
 "without_review":{
```

```
 "rule": "no_checking",
 "mode": "inherit"
 }
},
"name": "My workflow",
"description": "This is my workflow.",
"shared": "true",
"owners": [
 "Francois_Piccard",
 "Anna_Schmidt"
],
"end_rules": {
 "update": {
 "rule": "no_revision",
 "mode": "inherit"
 }
},
"auto_approve": {
 "rule": "votes",
 "mode": "inherit"
},
"counted_votes": {
 "rule": "members",
 "mode": "inherit"
},
"tests": [
 {
 "id": 2,
 "event": "onUpdate"
 "blocks": "nothing"
 },
 {
 "id": 5,
 "event": "onSubmit"
 "blocks": "approved"
 }
],
"id": "2"
},
]
```

**If a request fails**

`<error code>`:

- **400** Invalid parameter data specified
- **401** Insufficient permissions to create a workflow
- **501** Workflows are not enabled. To enable workflows, see ["workflow " on page 743](#)

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}
```

## Update a workflow

### Summary

Update a workflow

PUT /api/v10/workflows/{id}

### Description

Update a workflow. All values should be provided in the request. If not provided any missing values are reverted to default.

### Parameters

Parameter	Description	Type	Parameter Type	Required
id	The id of the workflow being updated	integer	path	Yes
name	The workflow name. Will be compared against other workflows and rejected if not unique	string	body	Yes

Parameter	Description	Type	Parameter Type	Required
owners	A list owners for the workflow. Can be users or group names (prefixed with swarm-group-). Users and group names must exist or the workflow will be rejected	array (of strings)	body	Yes
description	Description for the new workflow	string	body	Yes
shared	Whether this workflow is shared for other users that do not own it. Defaults to not shared	boolean	body	Yes

Parameter	Description	Type	Parameter Type	Required
mode	<p><b>Workflow rules:</b></p> <p>Each workflow rule must have a mode parameter. The only valid value for mode is inherit.</p> <p><b>Global workflow rules only:</b></p> <ul style="list-style-type: none"><li>■ default applies the workflow rule setting to projects and project branches that don't have an associated workflow. If a project or project branch has an associated Swarm workflow, the global rule is ignored.</li><li>■ policy applies a minimum workflow rule setting to all</li></ul>	string	body	Yes

Parameter	Description	Type	Parameter Type	Required
	<p>projects and project branches. If a project or project branch has an associated workflow, the global rule is merged with the workflow rule and the most restrictive setting is used. Displayed as <b>Enforce on</b> in the P4 Code Review UI, see "<a href="#">Global workflow</a>" on <a href="#">page 744</a>.</p>			
on_submit	<p>Data for rules when changes are submitted. Valid values for with_review are no_checking, approved, strict. Valid values for without_review are no_checking, auto_create, reject</p>	array	body	Yes



Parameter	Description	Type	Parameter Type	Required
end_rules	Data for rules when changes are submitted. Valid values are no_checking, no_revision.	array	body	Yes
auto_approve	Data for rules when changes are submitted. Valid values are votes, never.	array	body	Yes
counted_votes	Data for rules when counting votes up. Valid values are anyone, members.	array	body	Yes
tests	<p>A list of test ids for the workflow. Each test id has an event that determines when the test is run and a blocks value that determines whether a test failure blocks review approval.</p> <ul style="list-style-type: none"> <li>Valid values for event are onUpdate, onSubmit.</li> <li>Valid values for blocks are nothing and approved.</li> </ul>	array	body	No

## Example usage

### Update a workflow

```
curl -X PUT -H "Content-Type: application/json" -u "username:ticket" -d "@mybodyfilename.txt"
"https://myswarm-url/api/v10/workflows/2"
```

The "mybodyfilename.txt" file contains the updated workflow information:

```
{
 "name": "My workflow",
 "description": "This is my workflow and I have changed the description of it.",
 "shared": "true",
 "owners": ["Francois_Piccard,Anna_Schmidt"],
 "on_submit":{
 "with_review":{
 "rule":"no_checking",
 "mode": "inherit"
 },
 "without_review":{
 "rule":"no_checking",
 "mode": "inherit"
 }
 },
 "end_rules": {
 "update":{
 "rule":"no_revision",
 "mode": "inherit"
 }
 },
 "auto_approve":{
 "rule": "never",
 "mode": "inherit"
 },
 "counted_votes": {
 "rule": "members",
 "mode": "inherit"
 },
 "tests": [{
 "id": 2,
 "event": "onUpdate",
 "blocks": "nothing"
 },
 {
 "id": 5,
 "event": "onSubmit",
 "blocks": "approved"
 }
]
```

JSON Response:

HTTP/1.1 200 OK

```
{
 "error": null,
 "messages": [],
 "data": {
 "workflows": [{
 "on_submit": {
 "with_review": {
 "rule": "no_checking",
 "mode": "inherit"
 },
 "without_review": {
 "rule": "no_checking",
 "mode": "inherit"
 }
 },
 "name": "My workflow",
 "description": "This is my workflow and I have changed the description of it.",
 "shared": "true",
 "owners": [
 "Francois_Piccard",
 "Anna_Schmidt"
],
 "end_rules": {
 "update": {
 "rule": "no_revision",
 "mode": "inherit"
 }
 },
 "auto_approve": {
 "rule": "votes",
 "mode": "inherit"
 },
 "counted_votes": {
 "rule": "members",
 "mode": "inherit"
 },
 "tests": [
 {
 "id": 2,
 "event": "onUpdate",
 "blocks": "nothing"
 },
 {
 "id": 5,
 "event": "onSubmit",

```

```

 "blocks": "approved"
 }
],
 "id": "2"
 }]
}
}

```

#### If a request fails

<error code>:

- **400** Invalid parameter data specified
- **401** Insufficient permissions to update the workflow
- **404** Workflow does not exist or you do not have permission to update the workflow
- **501** Workflows are not enabled. To enable workflows, see ["workflow " on page 743](#)

HTTP/1.1 <response error code>

```

{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
 "data" : null
}

```

## Delete a workflow

### Summary

Delete a workflow

DELETE /api/v10/workflows/{id}

### Description

Delete a workflow for the provided id. This call must be authenticated and the user must have permission to edit the workflow. If the workflow is in use it cannot be deleted and an error message will be returned. The global workflow cannot be deleted.

**Parameters**

Parameter	Description	Type	Parameter Type	Required
id	The id of the workflow being deleted	string	form	Yes

**Example usage**

Delete a workflow that is not in use by a project

```
curl -u "username:ticket" -X DELETE "https://my-swarm-host/api/v10/workflows/1"
```

JSON Response

HTTP/1.1 200

```
{
 "error": null,
 "messages": [
 "Workflow [1] was deleted"
],
 "data": null
}
```

If a request fails

<error code>:

- **401** Insufficient permissions to delete the workflow
- **403** Cannot delete the global workflow
- **404** Workflow does not exist or you do not have permission to view this workflow
- **501** Workflows are not enabled. To enable workflows, see ["workflow " on page 743](#)

HTTP/1.1 <response error code>

```
{
 "error": <error code>,
 "messages": [{
 "code" : "<code string>",
 "text" : "<error message>"
 }],
}
```

```
"data" : null
}
```

# Glossary

## A

---

### access level

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

### admin access

An access level that gives the user permission to use privileged commands.

### archive depot

A special depot into which versioned files (also known as archive files) can be transferred. See also 'depot'.

### archive files

Versioned files that users submitted to a depot.

## B

---

### base

1. For files: The file revision that contains the most common edits or changes among the file revisions in the source file and target file paths. 2. For checked out streams: The public 'have' version from which the checked out version is derived. See also 'have list'.

### binary file type

A file type assigned to a non-text file. By default, the contents of each binary revision are stored in full.

### branch

(noun) A set of related files that exist at a specific location in the P4 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. To associate code reviews in P4 Code Review (formerly Helix Swarm) with the projects they are part of, add the 'branch' paths in the P4 Code Review project. (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 view**

A specification of the branching relationship (branch mapping) 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.

### **broker**

A P4 component that can apply rules and scripts based on incoming commands before passing them to a designated P4 Server.

## **C**

---

### **central server**

The one server that is innermost in a multi-server deployment. In the server specification form field for Services, the central server might be specified as "standard" or "commit-server". If edge servers are part of the multi-server deployment, the central server must be a commit server. See also 'upstream server'.

### **change review**

The process of sending email to users who have registered that they want to know about or watch changelists that include the specified files.

### **changelist**

The changes to files or stream specifications along with metadata, such as the list of changed files, their version numbers, the submitter of the changelist, and the submitter's description of the changes. A changelist is the unit of versioned work.

### **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 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 P4 files that is not streams-based. Uses the Perforce file revision model, not the graph model. The default depot name is depot. See also '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 root**

The topmost directory of a client workspace.

**client syntax**

The way of indicating the location of files in the client workspace that, unlike 'depot syntax', begins with //workspaceName/

**client workspace**

Directories on your machine where you work on file revisions that are managed by P4 Server.

**code review**

A process in P4 Code Review (formerly 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, such as a release codeline from a development codeline.

**comment**

Feedback provided in P4 Code Review on a on all or part of a changelist, review, or job.

**commit server**

The innermost P4 Server server in a topology with one or more edge servers.

**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.

**counter**

A persistent numeric or textual variable that records details relevant to the functioning of a P4 Server instance or scripts that access that instance. See 'real-time performance counters'.

**D**

---

**database**

The set of db.\* files in the P4ROOT directory that contain metadata that the P4 Server uses to operate on versioned files, users, protections, streams, changelists, and more.

**default changelist**

The unnumbered pending changelist that is automatically created when a file is opened for add, edit, or delete.

**deleted file**

A file with the head revision marked as deleted. Previous revisions of the file are still available. See also 'head revision' and 'obliterate'.

**delta transfer**

A P4 Server feature that might boost performance by sending over the network only the parts of a file that have changed.

**depot**

A file repository hosted on the P4 Server. A depot is the top-level unit of storage for versioned files, which are also known as depot files, archive files, or source files. It contains all versions of all files ever submitted to the depot. Except for obliterated files, any version of any file can be restored, including deleted files, but not obliterated files. An installation can have multiple depots, and they might be of different types, such as a local depot and a stream depot.

**depot root**

The topmost directory for a depot.

**depot side**

The first part of a client view mapping, specifying the location of files in a depot. See client side.

**depot syntax**

The syntax for specifying the location of files in the P4 Server depot as opposed to in the user's client workspace. Depot syntax begins with //depotName/

**diff**

1: A set of lines that do not match when two files, or stream versions, are compared. 2: To compare the contents of files or file revisions, or of stream versions. See conflict.

**distribution server**

A P4 Server that allows a limited set of content and history to be exposed to third parties, without exposing the main server or the internal network.

---

**E**

---

**edge server**

A server that is part of a commit-edge environment that can independently support work in progress for locally-bound clients, thereby reducing the load on the commit server.

### **exclusionary mapping**

A view mapping that excludes specific files or directories.

### **extension**

Custom logic that runs in a Lua engine embedded into the P4 Server. Extensions are an alternative to triggers. See the P4 Extensions Documentation.

## **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. A file conflict can occur when multiple users have the same file opened for edit.

### **file pattern**

A definition of a set of files that is made up of a file spec that might include wildcards and a revision specification. See 'revision specification'.

### **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 (#). Example: `testfile#3`

### **file specification**

Syntax that enables you to specify a set of files. A file spec might include wildcards and special characters for a range of dates, revisions, or changelists. File specs are also used to define the mappings of a client workspace and to specify a label.

### **file type**

An attribute that determines how the server stores and diffs a particular file. Examples of file types: binary, text, and unicode.

### **form**

A screen displayed by certain P4 Server commands to create a specification (spec) by assigning values to fields. For example, you use the client form to define a client workspace mappings. Commonly used forms include Branch, Change, Client, Depot, Group, Label, Server, Stream, and User.

**forwarding replica**

A P4 server that processes reporting (read) commands but forwards update (write) commands to the central server. Forwarding replicas can reduce the workload of the central server and be used as a backup server for disaster recovery.

**G**

---

**Git Connector**

A component of P4 Server that enables you mirror, cache, or replicate a Git repository into a P4 Server repo in a graph depot. See also 'hybrid workspace'.

**graph depot**

A depot of type graph that is used to store Git repos managed by P4 Server. See also Git Connector and classic depot.

**H**

---

**have list**

The list of file revisions most recently synced from the depot into the workspace. In other words, the list of files that the client workspace has.

**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**

For failover, a process that allows one server to monitor another server, such as a standby server monitoring its upstream server so that it is prepared to take over the role of the upstream server. See the 'p4 heartbeat' command.

**hybrid workspace**

A client workspace that supports both repos of type graph (see also 'Git Connector') and the classic P4 file revision model.

## I

---

### **integrate**

To compare two sets of files and determine which changes to propagate. A typical use case is to integrate between a development branch and a release branch.

## J

---

### **journal**

A file containing a record of every change made to the metadata since the time of the last checkpoint. This file grows as each transaction is logged.

### **journal notes**

Entries added to journals and checkpoints that provide information for subsequent users of the journal or checkpoint. For example, replica servers might take action based on the journal notes they process.

### **journal rotation**

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

### **journaling**

The process of recording changes made to the server database. See also 'metadata'.

## L

---

### **label**

A special filespec that gives a meaningful name to a set of file revisions. For example, the @Release2.3-August-2025 label is easier to remember than //depot/project/...@48296

### **lazy copy**

A method to conserve disk space by referencing files instead of duplicating their content.

**librarian**

A subsystem of the server that stores, manages, and provides the archive files to other subsystems of the server. See also 'archive files'.

**license file**

A file that ensures that the number of 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 syntax**

The way of indicating the location of files that, unlike 'depot syntax', is specific to the local operating system. This syntax is used as the second part of the client workspace mapping, after the depot syntax. Examples: /staff/maria/myworkspace/file.c for Linux and c:\staff\maria\myworkspace\file.c for Windows.

**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**

P4 Server supports the standard log, which is human-readable, as well as structured logs in a comma-separated value (CSV) format. The P4LOG environment variable specifies the log file for events, including errors. The human-readable P4AUDIT environment variable specifies the log file that records file transfers to users. Structured log files are typically processed by external tools. To learn more, see 'Logging' in P4 Server Administration Documentation.

**M**

---

**mapping**

A single line in a view definition, consisting of a left side, a blank space, and a right side. The mappings 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'.

## **merge**

1. To create new files from existing files, preserving their history when 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, such as P4 Merge.

## **merge file**

A file generated from two conflicting file revisions.

## **metadata**

Information that P4 Server maintains, such as who created file revisions in the depot, whether the file is a 'lazy copy', the current state of client workspaces, protections, groups, users, labels, streams, and branches. Metadata is stored in the server database and is separate from the 'archive files' that users submit from their client workspace into the depot.

## **N**

---

### **nonexistent revision**

An empty revision of a file that results from deleting a file or syncing to the #none revision.

### **numbered changelist**

A changelist to which the server has assigned a number. This is distinct from the default of an unnumbered pending changelist.

## **O**

---

### **obliterate**

Permanent removal of files and their history from the depot. The 'p4 obliterate' command makes it impossible to recover the file, unlike the 'p4 delete' command, which preserves the previous revisions.

### **offline database**

A copy of a P4 Server database that is kept up to date by manual or scripted processes that replay journals from a P4 Server.



**opened file**

A file in your client workspace that you have checked out because you intend to edit, add, delete, or branch the file. Opening a file from your operating system file browser is not tracked by P4 Server.

**owner**

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

**P**

---

**P4 Code Review**

A web-based code review tool for P4. Contributors share files, comment, suggest tasks, vote up or down, and submit final work. (Formerly Helix Swarm)

**P4 Server**

The program that manages the depot files, the metadata associated with those files, as well as the permissions the administrator grants to specific users and groups.

**p4d**

The binary file (p4d.exe on Windows) that runs P4 Server to manage depot files (versioned files) and the metadata of the server database.

**pending changelist**

A unit of versioned work that is not submitted. A changelist is created when you check out any file. A changelist is pending until you submit or revert the change.

**protections**

The permissions stored in the P4 Server protections table. User permissions can be managed with the 'p4 protect' command and the P4Admin application.

**proxy server**

A P4 server application that caches versioned files and can deliver them to clients on behalf of a P4 Server.

## R

---

### **real-time performance counters**

Various server-wide counters providing performance data in real time.

### **remote depot**

A depot located on a different P4 Server that the current P4 Server can access.

### **replica**

A P4 Server that automatically maintains a full or partial copy of the central server's metadata and that might contain related file content. The replica copies by using 'p4 pull' or 'p4 journalcopy'. A replica can be used as a backup server for disaster recovery.

### **repo**

The container of files that the Git Connector caches or mirrors from Git users. See also 'graph depot'.

### **resolve**

The process you use to manage the differences between two revisions of a file, or two versions of a stream spec.

### **revert**

To discard changes you made to a file in your client workspace without submitting the changes to the depot.

### **revision number**

An integer indicating a specific file revision. For example, if the file name is readme.txt and the revision number is 5, the file name would include a pound sign prefix and the number 5, such as readme.txt#5

### **revision range**

A range of revision numbers for a file, specified as the low and high end of the range. For example, myfile.txt#5,7 specifies revisions 5 through 7 of myfile.txt.

**revision specification**

A suffix to a file name that specifies one or more file revisions. Revision specifiers can be revision numbers, a revision range, change numbers, label names, date/time specifications, or client names.

**S**

---

**shelve**

The process of temporarily storing files in the server without checking in a changelist. Shelves are often used for reviews.

**stream**

An enhanced type of branch with a dedicated view for clients of the stream (see 'stream view') and built-in rules that determine how changes flow across the stream depot (see 'stream hierarchy'). A stream specification defines a stream. In P4V, stream specs are visible in the Streams Graph and the Streams tab.

**stream depot**

A depot used with streams and stream clients. A stream depot has structured branching, unlike the free-form branching of other depot types. See also 'classic depot' and 'graph depot'.

**stream hierarchy**

The set of parent-to-child relationships between streams in a stream depot.

**stream view**

A dedicated view shared by all clients of the stream. A stream view is defined by the Paths, Remapped, and Ignored fields of the stream specification. A stream view can be seen with the 'p4 stream -ov' command.

**submit**

To send a pending changelist from the client workspace to the depot.

**super access**

The highest access level, which gives this user permission to run every command of P4 Server, including commands that set protections, install triggers, shut down the service for maintenance, as well as perform failover and fallback.

## **sync**

To copy file revisions from the depot to your client workspace.

## **T**

---

### **target server**

The immediately upstream server for replica servers, edge servers, standby servers, proxies and brokers. See also 'upstream server' and 'central server'.

### **topology**

Data about the set of P4 services deployed in a multi-server installation, which might include a commit server, edge servers, standby servers, proxies, and brokers. P4Admin's Topology tab visualizes the topology and supports exporting the details, which might help for troubleshooting.

### **trigger**

A script to customize server behavior when specific conditions are met.

### **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 which you assign file types to files. For example, pdf is typically mapped to binary as opposed to text.

## **U**

---

### **unload depot**

A special depot for infrequently used metadata or to facilitate reloading workspaces that have been moved to a different server.

### **upstream server**

Any server in the inward direction, that is, toward the central server. For example, in an edge-to-edge configuration with a commit, edge1, and edge2, both edge1 and the commit server are upstream servers for edge2. See also 'central server'.

**user**

The identifier that P4 Server uses to determine who is performing an operation. The three types of users are standard (also includes the user with super access), service, and operator. Service users and operator users are restricted to a narrow subset of commands but do not consume a license.

**V**

---

**versioned files**

Source files stored in the depot, including one or more revisions to each file. Also known as archive files, archives, and depot files. Versioned files typically use the naming convention 'filename,v' or '1.changelist.gz'.

**view**

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

**W**

---

**workspace**

See client workspace.

**workspace view**

A set of mappings that specifies the correspondence between file locations in the depot and the client workspace. Each row in the workspace view consists of a pair of filespecs, first a depot filespec, then white space, and finally a filespec that shows a workspace location relative to the workspace root.

# Getting help

**Tip:**

Perforce Support cannot help you with user account creation or with resetting your password, please contact your company's Perforce administrator for help with this.

We look forward to hearing about your experiences with P4 Code Review, positive or negative, including *must-haves* or *it would be great if P4 Code Review....* Please feel free to contact us.

Please visit the [Perforce Support portal](#) for product support and contact information:

- Search our knowledge base
- Post on the Perforce forum
- Submit a Support request
- Call us for support
- View our video tutorials
- View our training options
- View the documentation and release notes
- Download Perforce software

Complete contact information is available on the [Perforce web site](#).

# License statements

For the licensing information of the third-party software included in this Perforce product, see the license file on the P4 Code Review server in [<swarm\\_root>/readme/LICENSE.txt](#).