

SDP Developer Guide

Version v2026.1, 2026-09-08

Table of Contents

DRAFT NOTICE	1
Preface	2
Terminology	3
Script Versioning	4
Working in Streams	6
Documentation Builds	8
Code Reviews	9
P4 Code Review and Stream Paths	10
Appendix A: Other Documentation	11
Appendix B: DRAFT NOTICE	12

DRAFT NOTICE



This document is in DRAFT status and should not be relied on yet. It is a preview of a document to be completed in a future release.

Preface

This guide is to aid people who contribute to the P4 Server Deployment Package (P4SDP). This includes Perforce Staff and contributors from the general public.

Terminology

- *Emergency Bug Fix*: A direct edit made in an already-cut release stream that changes something which actually ships in that stream's tarball—a script, not just a doc page—bypassing the normal `dev` → `main` → release-stream flow on purpose, to get a critical fix out faster than a full new Patch Release would allow. Always requires bumping that release stream's `Version` file and republishing its tarball. This is exceptional by design; if it starts feeling routine, that's a signal to cut a real Patch Release instead. See the [Release Process Overview](#)'s "Hot Fixes and Emergency Bug Fixes" section.
- *GA Release*: A General Availability (GA) release is the first release of a new major version. It includes a new SDP tarball as well as updates to files available on the web site with standard/published links.
- *Hot Fix*: A direct edit made in a release stream (or in `main`, ahead of the next Copy Up) that touches only web-facing, non-shipped content—e.g. doc pages or `README.md`—without generating a new SDP tarball. This is the preferred, lower-risk way to make an out-of-band correction, and should be the common case; it's not forbidden to go further, but if the fix needs to touch a script that actually ships, see *Emergency Bug Fix* instead.
- *Merge Down, Copy Up*: A mantra for a well-designed release process. See the [Perforce Directory Standard \(PDS\)](#) for information about this.
- *Patch Release*: A Patch Release is functionally identical to a GA release, in that it includes a new SDP tarball and updates to files on the web site. Any SDP release between major version releases of the P4 Server is a Patch.
- *Ready 5*: The property of a release process that ensures the team can always shift priorities and ship a patch quickly, even as various development tasks are in different states of readiness. Doing work properly and keeping the `mainline` clear of work that is not ready to ship is key to maintaining readiness.

Script Versioning

Many individual SDP scripts are versioned using the **+k** "keyword expansion" file type modifier. This file type modifier causes the P4 Server to update keywords in the content of the versioned file, e.g. our scripts, with a new version identifier each time the script is submitted.

Scripts that use this versioning method support the **-V** (version check) option, which gives results like these examples:

```
$ ccheck.sh -V
ccheck.sh version r26.1.0.33441
```

```
$ ccheck.sh -V
ccheck.sh version DEV_C2S.31580
```

Due to the way we are using Streams, the path to the versioned file contains meaningful information about which version of SDP the script is released with. For example, if the script path starts with `//p4-sdp/r26.1.0`, that script is part of the SDP 2026.1 GA release. Released versions of SDP align with the P4 Server format of `rXX.Y`, with an additional `.Z` patch digit that's always present (never omitted), where `XX` is a year identifier, `Y` increments with each major release in a year, and `Z` is `0` for the GA release and increments with each patch after it (e.g. `r26.1.0` for GA, `r26.1.1` for the first patch). Unreleased versions, such as those being developed and tested, use an ALL-UPPERCASE form of the development stream name, e.g. `DEV_C2S` relates to the `//p4-sdp/dev_c2s` development stream. The uppercase is used to emphasize that version is not released.

The number at the end is the changelist number of the individual change that produced that latest version of the script. For released versions, this changelist will be due to release-process related activities rather than the development code changes.

The following standard block of code (bash in this example) illustrates how the keyword is used to automatically update the version number with each submit.

```
# Version ID Block. Relies on +k filetype modifier.
#-----
# shellcheck disable=SC2016
declare VersionID='${Id: //p4-sdp/r26.1.0/doc/gen/gen_script_man_pages.sh#2 $ $Change:
31472 $'
declare VersionStream=${VersionID#*//}; VersionStream=${VersionStream#*/};
VersionStream=${VersionStream%/*};
declare VersionCL=${VersionID##*: }; VersionCL=${VersionCL% *}
declare Version=${VersionStream}.${VersionCL}
[[ "$VersionStream" == r* ]] || Version="${Version^^}"
```

The line that defines `VersionID` is modified by the P4 Server upon submit, with the `'$Id:$`` and `$Change:$` tags being replaced. This ensures that the version is reliably updated each time the script changes. Note that in this bash example, single quotes are used rather than double quotes to

prevent the bash shell from interpreting `$Id` and `$Change` as bash script variables. The `shellcheck disable=SC2016` comment silences as ShellCheck warning about accidental usage of single quotes suppressing expansion of variables. In this case, that is exactly the intent.

Working in Streams

Different types of work are done in different streams:

Stream Name	Type	Description of Work
//p4-sdp/r* Examples: //p4-sdp/r26.1.0 //p4-sdp/r26.1.1	release	Release process activities are done in release streams, such as updating the Version file and generating final versions of docs and Release Notes. Each release — the initial GA and every subsequent patch — gets its own freshly-cut release stream from main ; release streams are never patched or otherwise modified in place after they ship. See doc/ReleaseProcessOverview.md for the full process.
//p4-sdp/main	mainline	Reflects whatever was most recently released. Regression test suites target dev (the release candidate), not main — main only receives content via Copy Up from dev as part of cutting a release. Humans should do very little direct work in this stream outside of the release process itself.
//p4-sdp/dev	development	This is the default development stream. Work on features that are committed to be in the next release can be done directly in this stream, such as straightforward bug fixes and small, low-risk features. Anything submitted to this stream <i>must</i> be in a state where, if released today due to a need to ship an urgent patch possibly unrelated to the current change being submitted, it would be a Good Thing. Submitting something to the default dev branch is in effect saying, "Pending verification by regression test suites, this change is good enough to be shipped." Don't submit something in the default dev stream unless you intend to do any necessary iteration (e.g. based on regression test suite results) in short order.

Stream Name	Type	Description of Work
//p4-sdp/dev_* Examples: //p4-sdp/dev_c2s //p4-sdp/dev_rebrand //p4-sdp/dev_SDP-1265	development or sparsedev	Development tasks are done in feature streams if it is not certain they are ready or whether they will be included in the next release. The tag is a short tag name referencing the work, e.g. "c2s" for "Classic to Streams development work", or the tag can even be a JIRA issue tag, e.g. SDP-1265. Types of work done in development streams may include: • Projects with uncertainty in their development time frames, possibly large projects and/or those requiring significant iteration. • Prototype or Research and Development work that may never be released. Work in dev* streams can terminate stream and never be promoted, or deferred indefinitely. When choosing development vs. sparsedev stream type, consider these factors: • Use development streams if a plan on using push/fetch to work offline; as fetching doesn't a sparsedev stream only fetches stream-resident files, not the whole workspace. • Use sparsedev if you're working on focused changes, e.g. to just a few scripts or files.

Documentation Builds

HTML and PDF documentation are generated from AsciiDoc/Markdown sources via **make** in the **doc/** directory (see the [Release Process Overview](#)'s doc-generation steps).



Regenerate PDFs only as part of the release process itself, or when specifically checking PDF rendering/formatting—never as a routine step alongside an ordinary doc edit. PDFs are large, heavy files with almost no incremental value during development; regenerating one for every small doc change wastes disk space and adds bulk to changelists for no benefit. HTML is cheap to regenerate and should be kept current with its source; PDF generation is deliberately deferred to release time.

Code Reviews

Code reviews can occur in any stream. Both pre- and post-commit reviews are allowed. Reviews in the default dev stream can be done for changes initiated directly in the default dev stream, as well as Copy Up changes from dev* streams, thus reviewing the sum of a series of iterative changes in a lower dev* stream in a single review. More granular reviews can also occur in directly in dev* streams.

P4 Code Review and Stream Paths

Appendix A: Other Documentation

See Also:

- [SDP Release Process Overview](#).

Appendix B: DRAFT NOTICE



This document is in DRAFT status and should not be relied on yet. It is a preview of a document to be completed in a future release.