Perforce Streams Adoption Guide

Perforce Consulting Guide

Take advantage of Perforce Streams' built-in productive workflow process. Learn how to move an existing project into streams and how to address any legacy issues in the planning phase.

Table of Contents

Executive Summary

Perforce Streams are an innovative way to manage concurrent development, dependencies, and other common branching and release activities. Streams provide projects with a workflow framework based on best practices observed over many years. Streams are flexible enough to accommodate many branching and development models, including the mainline branching model and a promotion model suitable for website development.

Moving an existing project into streams requires some planning. Although the mechanical aspects of moving the data into a stream depot are straightforward, other areas must be addressed, including the impact on users and other tools.

This document highlights preparatory steps to consider before moving an existing project to streams. Although some planning is required, moving to streams presents an opportunity to take advantage of its built-in productive workflow process and address legacy bottlenecks.

Understanding Streams

Before reading this paper in detail, it is helpful to have a basic understanding of Perforce Streams: what they are, how they work, and the associated commands and tools. To get a firm grasp of the mainline branching model that underpins Perforce Streams, the book **Practical Perforce** (O'Reilly, 2005) is the best starting point. Also useful is the **Perforce Directory Standard**, which has been updated to cover streams.

Some key streams concepts are covered in the following sections, as they directly impact migration planning. To learn more, see Additional Resources at the end of this paper.

Altered Directory Structure

Prior to streams, directory structure was used to convey important contextual information about the intended use and relative stability of branches. For example, consider the Jam project, which has a main branch and two release branches. A common directory structure is shown in Figure 1 (left), where a REL container directory indicates that the subdirectories are release maintenance branches.

After moving the Jam project to streams (and adding a couple of new branches), the directory structure is flat (see Figure 1 (R)).
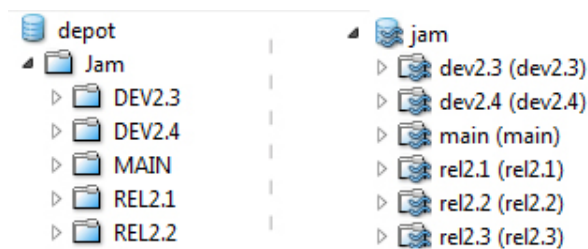

Figure 1: Jam branches (L) and the Jam branches as streams (R).

Branch Stability and Use

According to the mainline model, a release stream follows a particular flow of change pattern, governing when changes are merged back to the main branch. Before moving to streams, the REL container level in the Jam project conveyed important information about the branches in that container. The branches were release maintenance branches, implying a higher level of stability than development branches or the main branch. With streams, this information is captured in the stream metadata and presented visually in the Stream Graph (see Figure 2).

Stream Composition and Inheritance

Streams, like workspaces and branches, have views. A product architect can use the stream view to define the set of modules or components in a stream. In other words, a stream view defines which files are actually branched for work, which are imported or excluded from the parent, and which are imported from other parts of the repository.

The stream view is inherited by all child streams and workspaces, which simplifies the start-work process for new project users. When a new user creates a workspace from a stream, the workspace view is generated automatically. Workspace views are also updated automatically when moving from stream to stream.
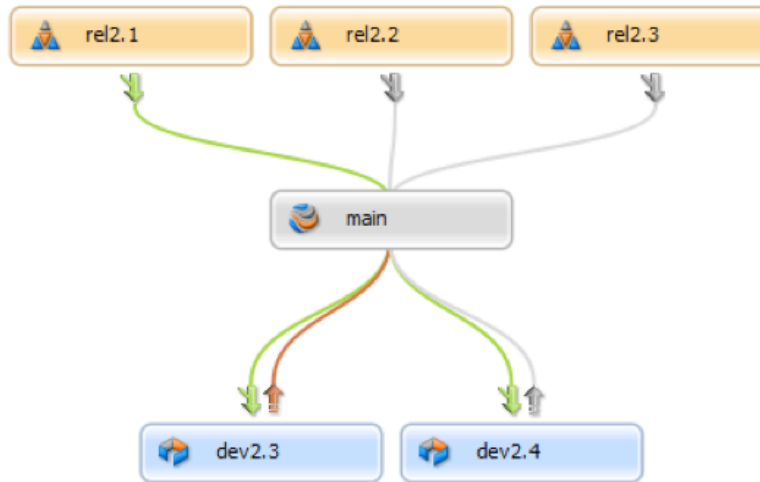
Figure 2: Stream Graph conveys branch stability and the intended flow of change.

How to Move: Seeding the Streams

Mechanically moving data into a stream depot for an existing project is straightforward. Although a baseline-and-branch import style (BBI)1 approach could be used, in most cases it will be sufficient to simply integrate the tips of the relevant branches into new streams. Perforce tracks and respects the indirect branching history between the new streams.

At a high level, here are the steps involved:

Define a new stream depot for the project (administrative access is required).

Choose the relevant branches to copy to the stream depot. All branches could be included, or only those branches that are still actively being used.

For each branch, define an equivalent stream. Start with the main stream and keep in mind the following:

- **Consider which stream type to use.** A mainline stream is the main branch and there is usually only one mainline per stream depot. Release streams are assumed to be more stable than the parent. Development streams are less stable than the parent.
- **Consider the allowed flow of change, which is set in a stream's definition.** Most development branches allow a bidirectional flow of change, while release maintenance branches usually do not accept changes from the parent.

- **Choose stream names carefully.** Remember that the stream depot directory structure is flat. The stream metadata captures a lot of the important information about streams, but a good naming convention will help your users.
- **Carefully determine the practical implications of the parent-child relationship between the streams.** Consider how you perform merges between branches. For example, if you normally merge bug fixes from oldest to newest release through to the mainline, you may want the oldest release to use the next oldest release as its parent, and so on. In a more advanced branch model, an older 1.0 release receives bug fixes first, followed by the 1.5 release, and then the main branch (see Figure 3).
- **Capture any relevant information about stream composition in the stream views.** This information may be currently stored in a branch spec, or may be implied. For instance, in Figure 3 the dev-db stream imports two modules from the integration stream, as shown in the stream paths.

The 2011.1 Perforce integration engine handles renames without relying on branch mappings if the `p4 move` command was used.
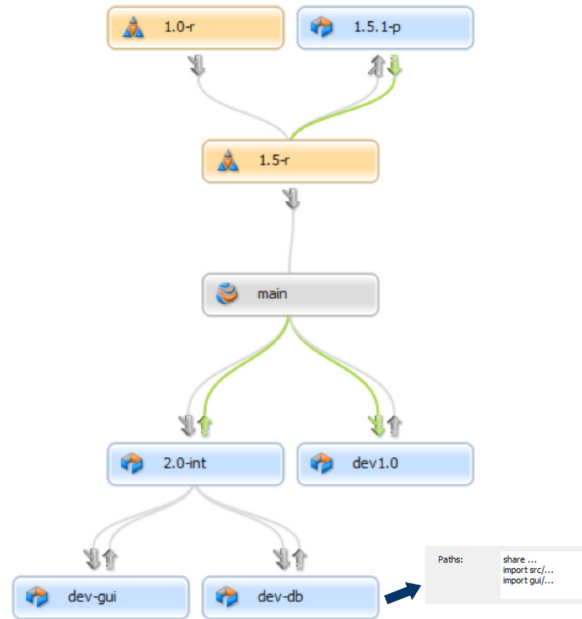
Figure 3: A more advanced branch model in the Stream Graph.

Copy each branch from its original location to the equivalent stream using the `p4 copy` command. For example:

```
p4 copy -v //depot/Jam/MAIN/... //jam/
main/...
p4 submit -d "seeding Jam stream
mainline"
```

The Revision Graph for a file shows that its pre-streams history is readily accessible (see Figure 4).

In most cases, due to the indirect history between the newly copied streams, merging between two streams will return results similar to merging between two original branches. Having copied them from their legacy location, the streams may diverge going forward, but their merge history will not change.

For example, say there is one bug fix waiting to be merged from the Jam REL2.1 branch back to MAIN. After integrating the current state of MAIN and REL2.1 to streams, to show the pending bug fix, run the command:

```
p4 integ -S //jam/rel2.1
```

In cases with more complex merge history, it would be wise to preview any merge operations immediately after moving to streams, to ensure that the results are as expected.
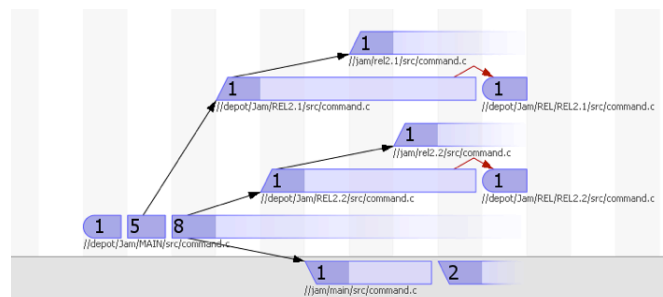

Figure 4: Revision Graph showing pre-Streams history of the Jam branch.

Impact Analysis of Streams: What to Expect

For most people, streams will be easy and straight-forward to use. However, streams are a departure from classic Perforce branching. Advance training and documentation should be provided. Here are some ways that streams may impact your users.

Impact on Users

Of particular interest is the new, simplified workflow that streams support and the related commands and GUI functions, such as:

- Simplified workspace creation and management.
- Stream, or branch, creation.
- The merge down/copy up paradigm, supported by the merge and copy commands.
- Stream view management.
- In-place branching and fast workspace switching.

Aside from learning the new workflow and commands, the decision to start working in streams will directly impact your users. New workspaces and other tools and scripts may need to be updated, and users will need to understand the new directory locations and the structure of streams.

Impact on Release and Project Management: Who and When to Move

Streams simplify many release management activities. The flow of change is guided by the streams framework, and the stream view allows for dependency management and code re-use. Build and release scripts and tools will need updating to take advantage of these new features.

If the previous branching model was well laid out, with good supporting scripts and tools, then the overall workflow will feel familiar. An inefficient branch and release model will become more visible in the streams framework; users will find themselves working outside of the expected guidelines frequently. Moving to streams is thus a good opportunity to identify and fix any problems in the legacy model.

Another important choice is how many teams to move over, and when they should move. Moving over the entire organ-ization to streams at once simplifies some of the planning, since legacy tools and processes could be retired. However, doing so would require very careful planning and management.

Moving over a pilot team, and then moving over additional teams incrementally, allows more time for new processes and tools to be fleshed out. If there are strong dependencies between the work done by different teams, additional work will be required so that the team using classic Perforce can still collaborate with the team using streams, and vice versa. The migration schedule for various teams may depend on the level of collaboration, schedule of releases, and other important milestones, as well as the ability of each team to adapt to new processes.

In the course of planning, you may find that some teams should not move to streams. Teams that simply use Perforce as a document repository, who perform little or no branching and parallel work, would not benefit much from the streams workflow. Some teams may already have in place a comprehensive set of scripts and tools to support their unique development process, in which case moving to streams may prove disruptive. Streams provide built-in workflow based on observed best practices. If your needs are currently well-met, you may not realize any gains by moving to streams.

Impact on Other Tools

There are many ALM tools that interact heavily with Perforce: code review tools, defect trackers, and continuous integration engines all interface with Perforce to some degree. How streams will impact these tools should be analyzed and understood in advance.

Be aware that every tool that relies on knowing and understanding the product directory layout will need to be changed when you migrate to streams. The directory structure will be flat, so tools and scripts cannot rely on the same directory conventions in use to convey structural context. Streams capture the information in a different way, so the tools that rely on this information must account for this change.

Impact on Administration

Using Perforce Streams requires a 2011.1 Perforce Server and streams-enabled clients. Administrators will need to manage a server upgrade and make sure that all affected users have appropriate client software.

Using streams will most likely require the creation and use of more depots (one per product) than before. Creating these depots and setting the appropriate permissions will be an administrative task.

Conclusion

This document details, at a very high level, some of the considerations involved when moving existing users and projects to Perforce Streams. Planning and preparation are key to a successful process transition and moving to streams is no exception, particularly if you want to realize the potential productivity gains.

Learn More

Read these selected articles or all of the articles listed in the **Streams category** from the Perforce Blog:

- Introduction to Streams
- High Level Streams Overview
- Streams Tutorial
- Streams and the Flow of Change
- Stream Views Explained

- Streams FAQ
- Origins of Streams
- Streams Solve Developer Challenges
- Streams Live!
- Streams Applied: Document Management Workflow
- Streams Applied: Mainline Development
- Perforce Directory Standard and Streams

For more information on Perforce Streams:

- Visit our website: **perforce.com/streams**
- Try Streams: **perforce.com/downloads**
- Take a self-paced **eLearning** course on Streams
- For assistance on adopting or migrating to Perforce Streams, email consulting@perforce.com

PERFORCE

Version everything.