QUICK TIPS

10 Tips for Healthy Check-Ins

The right version management enables high productivity in your development teams, protects your valuable assets and can give you the security to experiment and create great products. Following some simple best practices will enhance your team's productivity.

Here are 10 tips for ensuring healthy check-ins.

1	**Use Atomic Operations**	Ensure that all files involved in a change are successfully committed or none of them are. This is a feature of the VCS, so make sure your tool treats a set of changes as an atomic operation.
2	**Have a Single Intent**	It's tempting to include a number of changes in a single check-in, but that will cause problems later if any of those changes needs to be backed out. Make each change self-contained.
3	**Keep the Scope of a Check-In Narrow**	Break tasks down into smaller chunks (e.g., separate refactoring from introducing a new feature) and commit separately where possible.
4	**Describe the Reason for a Change**	Always use comments and focus on why a change was made and why this particular approach was taken. The details of the how will be apparent by comparing the before and after revisions.
5	**Make Check-Ins Consistent**	Your product should be able to be built (and pass smoke tests) after a check-in. Avoid partial check-ins.
6	**Make Every Check-In Complete**	When checking in a change, make sure all the affected files are included and, for APIs, include test stubs. If teammates pick up the change, they should get everything they need to accept the change without breaking their builds.
7	**Use Code Reviews**	Protect the quality of your mainline by having all code reviewed before committing to the mainline. This step has other benefits, including sharing knowledge across the team.
8	**Ensure That a Check-In Is Traceable**	For security and audit purposes, ensure that every change can be traced and it's clear who made the change, where and why.
9	**Have Reversible Check-Ins**	Following these recommendations should mean it's possible to back out a change and reapply it later if needed.
10	**Have a Single Policy for Each Codeline**	If different projects or branches need different policies, keep them in separate codelines and don't change them during a project.

PERFORCE

Perforce Software Inc.
2320 Blanding Avenue, Alameda, CA 94501, USA | Phone: +1 510.864.7400 | Fax: +1 510.864.5340
Email: info@perforce.com | **perforce.com**