Extracted from:

Pragmatic Version Control

using Subversion, 2nd Edition

Chapter 7

File Locking and Binary Files

A common question when learning about version control is, "But what happens if two people edit the same file? Won't we waste our time undoing each others' changes?" Thanks to the magic of text merging, most of the time it isn't a problem. But what if the file is a picture or CAD model and cannot be merged? Subversion 1.2 introduced optional file locking which can help avoid problems with unmergeable files.

7.1 File Locking Overview

Many projects contain unmergeable files. Sound, graphics, and even many document formats cannot be merged in any meaningful way. If Alice and Bob both decide to edit CurrencyConversionRates.xls *at the same time*, one of them will be first to commit and the other will have to re-do their changes.

Fundamentally, this problem is about your team not communicating effectively. It's unlikely that Alice and Bob should both be editing the project theme song audio file at the same time, but without asking everyone else on the team whether they have that file open there's a chance they might be wasting their time. Subversion provides a mechanism to help the team communicate through optional *file locking*.

file locking

Any file can be set to require a lock before it is edited by setting its needs-lock property (it doesn't matter what the property contains—if it's set, Subversion will enable locking on that file). Any file with locking enabled will be checked out read-

only in the working copy. Most modern editors will refuse to edit a read-only file, or will at least warn that you are doing so, in an effort to remind the user that the file needs to be locked before editing.

We can issue a svn lock command to obtain a lock on the file. The Subversion client will chat with the server ensuring the file isn't already locked, obtain a *lock token*, and then mark the file read-write in the working copy. Additionally we can specify a lock comment informing other users why we locked the file.

lock token

If a file is locked, another user cannot lock the file or commit a change to it without first destroying the original lock (we'll talk more about situations in which this is appropriate later). When the user who locked the file is done and commits their changes, the lock is released.

Let's see how this works in practice. If you want to follow along with these examples, create separate working copies for Alice and Bob similar to those we created in Section 6.7, *Handling Merge Conflicts*, on page 86.

7.2 File Locking in Practice

The Sesame project is moving up in the world. In addition to all of its existing features, our customers now want the software to work in many different countries. As part of this initiative we'll need to convert between local currencies. The real system will use some kind of web service to find out what the exchange rates are, but for testing, Bob would like to use something simple such as an Excel spreadsheet.

Why File Locking is Important

Bob creates a spreadsheet file, CurrencyConversionRates.xls, and adds it to the repository:

```
sesame>svn add CurrencyConversionRates.xls
A  (bin)  CurrencyConversionRates.xls
sesame>svn commit -m "Added conversion rates for testing"
Adding  (bin)  CurrencyConversionRates.xls
Transmitting file data .
Committed revision 32.
```

Subversion automatically detects that the spreadsheet is a binary file when it's added, which also flags it as being non-mergeable. If Alice checks out revision 32 and both Bob and Alice change the file and attempt to commit, only one of them will succeed. Here's what Bob might see:

```
sesame>svn commit -m "Added Norwegian Krona conversion rate"
Sending        CurrencyConversionRates.xls
Transmitting file data .svn: Commit failed (details follow):
svn: Out of date: '/trunk/CurrencyConversionRates.xls'
     in transaction '43-1'
```

Bob's working copy is out of date because Alice snuck her changes in first. If this were a text file, Bob could simply update his working copy and Subversion would merge Alice's committed changes with Bob's pending changes. This doesn't work for the spreadsheet, however; it just produces a conflict:

```
sesame>svn up
C    CurrencyConversionRates.xls
Updated to revision 33.
```

Subversion tells Bob his copy of CurrencyConversionRates.xls is conflicting with the new revision in the repository. Bob has some options now. He can do some detective work with svn log to see who else changed the file, and he can choose to keep his changes, keep Alice's changes, or manually merge the two files. All of this seems to be quite a lot of work.

Enabling Locking on a File

Bob decides he'll throw away his changes and redo them. After all, he only added a single line to the spreadsheet and can quickly re-apply his change to the latest version. He'd like to avoid the same problem in the future, though, so he adds the svn:needs-lock property to the spreadsheet.

```
sesame>svn propset svn:needs-lock true CurrencyConversionRates.xls
property 'svn:needs-lock' set on 'CurrencyConversionRates.xls'
sesame>svn commit -m "Enabled locking for spreadsheet"
Sending        CurrencyConversionRates.xls
Committed revision 34.
```

Bob sets svn:needs-lock to "true" (remember it doesn't actually matter what the property contains; if it is present Subversion enables locking for the file). This property change doesn't take effect until it is committed to the repository. If you're adding an unmergeable file and would like to enable locking, it's good

practice to set the svn:needs-lock property right away. Subversion's autoprops, covered in Section 6.4, *Automatic Property Setting*, on page 74, can help with this.

Basic File Locking

Once Alice and Bob update their working copies, Subversion will make the CurrencyConversionRates.xls file read-only. The idea behind making the working copy read-only is that next time a user edits the file, they will be reminded they are attempting to change a read-only file and remember to lock the file before continuing. Depending on the application used to edit the file you may or may not get a warning. Excel will happily open a read-only file and let you change it without giving any warning—it's only when you come to *save* the modified spreadsheet that you're prompted for a new filename. This isn't usually too much of a problem, though, as many users instinctively hit "save" quite often. Other applications such as graphics or sound editors may treat read-only files differently. You will have to experiment with your particular application to find out.

After setting svn:needs-lock, Bob decides to add the Norwegian Krona exchange rate again. This time, he locks the file before editing it.[1]

```
sesame>svn lock CurrencyConversionRates.xls          \
          -m "Adding Norwegian Krona"
'CurrencyConversionRates.xls' locked by user 'bob'.
```

It's advisable to always include a comment indicating why you are locking the file. Subversion can provide the lock comment to other users and it's a good way to improve communication. Bob can now examine the file and see that it's locked:

```
sesame>svn info CurrencyConversionRates.xls
Path: CurrencyConversionRates.xls
Name: CurrencyConversionRates.xls
URL: svn://olio/sesame/trunk/CurrencyConversionRates.xls
Repository Root: svn://olio/sesame
Repository UUID: 63a31077-dc47-8e48-8372-099aabc6682c
Revision: 34
Node Kind: file
Schedule: normal
Last Changed Author: bob
```

[1] Subversion will only let you lock a file if it is up to date—you cannot lock an old revision.

```
Last Changed Rev: 34
Last Changed Date: 2006-03-06 15:31:04 -0700 (Mon, 06 Mar 2006)
Text Last Updated: 2006-03-06 15:29:58 -0700 (Mon, 06 Mar 2006)
Properties Last Updated: 2006-03-06 15:30:40 -0700 (Mon, 06 Mar 2006)
Checksum: 7cd95b6dcf6b3ce39baf073f56253e20
Lock Token: opaquelocktoken:42eef8ec-0355-d548-805b-16b5a8e830aa
Lock Owner: bob
Lock Created: 2006-03-06 17:10:17 -0700 (Mon, 06 Mar 2006)
Lock Comment (1 line):
Adding Norwegian Krona
```

There's a lot of information here, but the stuff we're interested in is the final five lines. We can see that Bob's working copy has a *lock token* and that Bob is the lock owner. We can also see when the lock was created and Bob's comment explaining why he locked the file.

If Alice now attempts to lock the file, she'll receive an error.

```
sesame>svn lock CurrencyConversionRates.xls       \
          -m "Adding Euro conversion rate"
svn: warning: Path '/trunk/CurrencyConversionRates.xls'
     is already locked by user 'bob'
     in filesystem '/home/svnroot/sesame/db'
```

Subversion lets her know that Bob has already locked the file. If Alice runs svn info on her working copy she won't see a lock token, indicating she doesn't have a lock (she couldn't, Bob has a lock already). For Alice to find out more about why the file is locked, she can ask Bob directly (improving team communication) or she can ask the Subversion server. Running svn info with the full URL for the file yields more information:

```
sesame>svn info svn://olio/sesame/trunk/CurrencyConversionRates.xls
Path: CurrencyConversionRates.xls
Name: CurrencyConversionRates.xls
URL: svn://olio/sesame/trunk/CurrencyConversionRates.xls
Repository Root: svn://olio/sesame
Repository UUID: 63a31077-dc47-8e48-8372-099aabc6682c
Revision: 34
Node Kind: file
Last Changed Author: bob
Last Changed Rev: 34
Last Changed Date: 2006-03-06 15:31:04 -0700 (Mon, 06 Mar 2006)
Lock Token: opaquelocktoken:42eef8ec-0355-d548-805b-16b5a8e830aa
Lock Owner: bob
Lock Created: 2006-03-06 17:10:17 -0700 (Mon, 06 Mar 2006)
Lock Comment (1 line):
Adding Norwegian Krona
```

Alice needs to use the full URL to find out about Bob's lock—if she uses just the file name, Subversion shows information about her working copy. Alice's working copy doesn't have the

lock, so she needs to ask the server about the latest version of the file.

Once Bob is done editing the file he can commit. When you commit a file or directory, Subversion automatically releases any locks you are holding.[2]

Breaking Locks

The owner of a lock can always use svn unlock to release it. But what if the lock owner isn't currently available? Suppose Alice tries to lock the exchange rates file to add some data. Subversion warns her the file is already locked, so she does a bit of investigation:

```
sesame>svn info \
  svn://olio/sesame/trunk/CurrencyConversionRates.xls | grep Lock
Lock Token: opaquelocktoken:42eef8ec-0355-d548-805b-16b5a8e830aa
Lock Owner: bob
Lock Created: 2006-03-06 17:10:17 -0700 (Mon, 06 Mar 2006)
Lock Comment (1 line):
```

Alice can see that Bob has a lock on the file, but he created it yesterday and still hasn't released the lock. Since Bob is off sick today, Alice decides to break the lock so she can make her change. She also makes a mental note to remind Bob not to leave important files locked for too long in the future.

The svn unlock command can be used to release someone else's lock on a file. Alice needs to pass the --force option because she doesn't own the lock herself:

```
sesame>svn unlock svn://olio/sesame/trunk/CurrencyConversionRates.xls
svn: warning: User 'alice' is trying to use a lock owned
     by 'bob' in filesystem '/home/svnroot/sesame/db'

sesame>svn unlock --force                          \
           svn://olio/sesame/trunk/CurrencyConversionRates.xls
'CurrencyConversionRates.xls' unlocked.
```

If Alice forgets to tell Bob that she broke his lock, when he tries to commit the change Subversion will let him know he no longer has a matching lock token. The token in Bob's working copy corresponds to the lock that Alice broke, so Bob will have to attempt to lock the file again before he can commit it.

[2] Subversion releases locks for *all* files in your working copy, not just those you are committing. This encourages users to release locks as quickly as possible, but might catch you by surprise the first few times you do it.

```
sesame>svn commit -m "Added Norwegian Krona"
Sending        CurrencyConversionRates.xls
Transmitting file data .svn: Commit failed (details follow):
svn: Cannot verify lock on path '/trunk/CurrencyConversionRates.xls';
     no matching lock-token available
```

Given that Alice can just forcibly unlock the file, you might think that Subversion's file locking is a bit pointless. Bob is in the same situation as if there were no locking at all—he has to decide whether to throw away his changes or overwrite Alice's. What we have achieved, however, is better communication between people editing the file. Alice knew she was breaking Bob's lock, and did so for a good reason—Bob wasn't at work that day and Alice needed to get on with the project.

Subversion allows you to restrict who can lock and unlock files, and who can break locks, through the use of special hook scripts. The *pre-lock* and *pre-unlock* hooks run before a file is locked or unlocked (respectively). These hooks can examine whether a file is already locked and, depending on site policy, restrict lock breaking operations to certain users.

The *post-lock* and *post-unlock* hooks can be used, for example, to send email after a lock is broken. That way Alice can't forget to tell Bob she broke his lock, there will be an email waiting for him to let him know.

After forcibly releasing Bob's lock, Alice should then lock the spreadsheet so she can make modifications to it. But there's a small chance someone else will lock the file in between Alice typing those two commands, so Subversion also provides the ability to *steal* the lock from another user.

Using the --force option with svn lock will steal the lock without giving anyone else the chance to lock the file.

```
sesame>svn lock --force CurrencyConversionRates.xls
'CurrencyConversionRates.xls' locked by user 'alice'.
sesame>svn info CurrencyConversionRates.xls | grep Lock
Lock Token: opaquelocktoken:b8c434ce-98ef-1046-a553-7479abccdbca
Lock Owner: alice
Lock Created: 2006-03-07 14:44:26 -0700 (Tue, 07 Mar 2006)
```

It's important to note that a lock is specific to a working copy as well as being owned by a particular user. If Bob locks a file using his office computer, then works from home the next day on his laptop, the lock is still stored on the office machine. If

he wants to edit the file at home he'll have to break the lock held by the office working copy.

7.3 When to use Locking

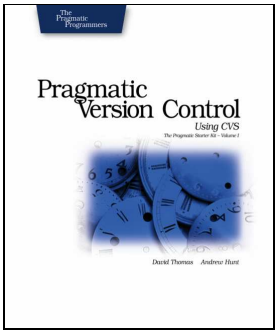
Subversion's optional locking is very useful for controlling access to unmergeable files. Adding svn:needs-lock to a file can help your team communicate more effectively about who is working on the file, and can help prevent wasted effort.

Try to lock as few files as possible for as short a time as possible. Don't be like Bob—locking a file and then going home for the night. The longer a file is locked, the greater the chance someone else has to wait around before they can make their changes. In the worst case, Alice might be waiting for Bob to finish work on a file while Bob waits for Alice to finish work on a different file. The two of them will wait forever for the other's lock to be released. Developers moving to Subversion from systems such as Visual Source Safe will be familiar with this "deadlock" situation.

If your files are text, such as program code, Subversion can usually merge changes for you and you don't need to lock them. There can be cases where it seems attractive to start locking mergeable files, such as an important source code file that developers update quite often and which seems to encounter a lot of conflicts. Usually the best solution isn't to start locking the file, it's to figure out how to split the file into several logical pieces so the whole team doesn't need to continually trip over each other when making changes.

Pragmatic Starter Kit

Version Control. **Unit Testing**. **Project Automation**. Three great titles, one objective. To get you up to speed with the essentials for successful project development. Keep your source under control, your bugs in check, and your process repeatable with these three concise, readable books from The Pragmatic Bookshelf.

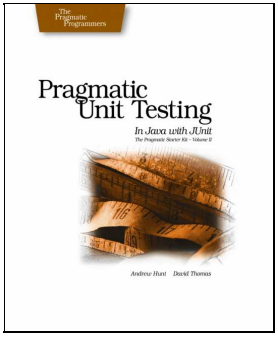


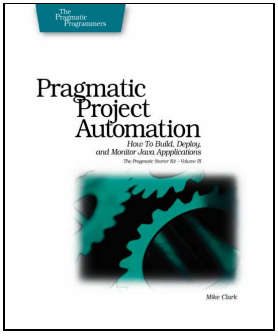
The CVS companion to this book • Keep your project assets safe—never lose a great idea • Know how to UNDO bad decisions—no matter when they were made • Learn how to share code safely, and work in parallel • See how to avoid costly code freezes • Manage 3^{rd} party code • Understand how to go back in time, and work on previous versions.

Pragmatic Version Control using CVS
Dave Thomas and Andy Hunt
(176 pages) ISBN: 0-9745140-0-4. $29.95

• Write better code, faster • Discover the hiding places where bugs breed • Learn how to think of all the things that could go wrong • Test pieces of code without using the whole project • Use JUnit to simplify your test code • Test effectively with the whole team.

Pragmatic Unit Testing
Andy Hunt and Dave Thomas
(176 pages) ISBN: 0-9745140-1-2. $29.95
(Also available for C#, ISBN: 0-9745140-2-0)





• Common, freely available tools which automate build, test, and release procedures • Effective ways to keep on top of problems • Automate to create better code, and save time and money • Create and deploy releases easily and automatically • Have programs to monitor themselves and report problems.

Pragmatic Project Automation
Mike Clark
(176 pages) ISBN: 0-9745140-3-9. $29.95

Visit our secure online store: `http://pragmaticprogrammer.com/catalog`

Pragmatic Starter Kit

Version Control. **Unit Testing**. **Project Automation**. Three great titles, one objective. To get you up to speed with the essentials for successful project development. Keep your source under control, your bugs in check, and your process repeatable with these three concise, readable books from The Pragmatic Bookshelf.

Visit Us Online

Pragmatic Version Control using Subversion
`pragmaticprogrammer.com/titles/svn`
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
`pragmaticprogrammer.com/updates`
Be notified when updates and new books become available.

Join the Community
`pragmaticprogrammer.com/community`
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
`pragmaticprogrammer.com/news`
Check out the latest pragmatic developments in the news.

Buy the Book

If you liked this PDF, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: `pragmaticprogrammer.com/\HomePageUrl`.

Contact Us

Phone Orders:	1-800-699-PROG (+1 919 847 3884)
Online Orders:	`www.pragmaticprogrammer.com/catalog`
Customer Service:	`orders@pragmaticprogrammer.com`
Non-English Versions:	`translations@pragmaticprogrammer.com`
Pragmatic Teaching:	`academic@pragmaticprogrammer.com`
Author Proposals:	`proposals@pragmaticprogrammer.com`