CollabNet Subversion 1.9 for Developers

Enterprise Lab Exercises

Command Line

COLLABNET

4000 Shoreline Court, Suite 400
South San Francisco, California 94080 U.S.A.

888.778.9793 toll free
650.228.2500 voice
650.228.2501 fax
www.collab.net
E-mail info@collab.net

Table of Contents

1. Lab 1 – Essential Concepts 1

1.1. Standard Work Cycle – Part 1

Steps	Comments
1. Create 2 working copies based on the repository's trunk in directories named *JoeDev* and *JillDev*	
2. Using *JoeDev*, create and add a new directory under the directory *EasySVN* by the name *FirstLessons* and a file within it named First.txt	Structural change
3. Delete an existing file called *DeleteMe*	Structural change
4. Rename the existing file RenameMe.txt to *ChangedName.txt*	Structural change
5. Choose an existing file EditMe.txt and change *"anything"* to *"something"* on the 2nd line and save your changes	Content change
6. Use Subversion to examine all your changes done from step 2 to step 5	(High level checking)
7. Choose the file *EditMe.txt* whose content was changed and compare with the previous revision	Examine changes (Low level checking)
8. Undo the rename you did earlier	Reverting changes
9. Bring in changes made and committed by others	Potential content change
10. Resolve any conflict detected	Resolve conflicts
11. Commit your changes to the repository	Commit changes
12. Bring in changes made and committed by others	Remove mixed revisions state

1.2. Standard Work Cycle – Part 2

Steps	Comments
1. Update the *JillDev* working copy	Content change
2. Create and add a new directory under *EasySVN* by name *SecondLessons* and add a file within it named Second.txt	Structural change
3. Move the directory *MoveMe* to the directory *AddMe*	Structural change
4. Get a lock on the existing file *EditMe.txt*.	Lock
5. Edit the existing file *EditMe.txt* and change *"line"* to *"sentence"* on the 2nd line.	Content change
6. Examine all your changes done from step 2 to 4	(High level checking)
7. Bring in changes made and committed by others	Same as step 1
8. Resolve any conflict detected	Resolve conflicts
9. Commit your changes to the repository	Commit changes
10. Bring in changes made and committed by others	Remove mixed revision state

1.3. Examine History

Steps	Comments
1. Using *JillDev*, show the history for the trunk	
2. Show the history for the specific file – *EditMe.txt*	
3. Check the difference between any two sequential revisions of the file, *EditMe.txt*, which was modified in Exercises 1.1 and 1.2	
4. Check *EditMe.txt* to find who last changed line number 3	

1.4. Property Changes

Steps		Comments
1.	Using JillDev, add a property to enforce locking on the image file, “Subversion Logo.jpg”	Adding a property
2.	Check that the Read only attribute is not set on the file in the Windows properties	
3.	Commit your changes	Committing a property
4.	Update your working copy	
5.	Check that the Read only attribute is now set on the file in the Windows properties	
6.	Get a lock on the image file, “Subversion Logo.jpg”	
7.	Check that the Read only attribute is not set on the file in the Windows properties	
8.	Release the lock	

2. Lab 2 – Essential Concepts 2

2.1. Traversing

Steps	Comments
1. Using JoeDev, update EditMe.txt to revision 2	Moving back in time
2. Note that the update records a modification made to that file. Examine the file to see that your changes are gone	
3. Update the directory EasySVN to revision 1	
4. Note the changes recorded by the update	
5. Update your working copy to point to the HEAD on the trunk	

2.2. Tree Conflict

Steps	Comments
1. Using *JillDev*, update your working copy to point to the HEAD on the trunk.	Notice the speed of creating a branch
2. Using *JoeDev*, move the directory *SecondLessons* to the directory *AddMe*	
3. Commit your changes	
4. Using *JillDev*, edit the file *Second.txt* in the directory *SecondLessons* and enter the text “Tree conflict looming”	
5. Update your working copy	Notice that you have a tree conflict
6. Use a Subversion operation to identify the type of tree conflict	
7. Fix the conflict by getting your edited content of the file *Second.txt* into that file in its new location.	
8. Inform Subversion that you resolved the conflict	
9. Commit your changes.	

2.3. Merging

Steps	Comments
1. Using *JoeDe*v, create a branch *"Demo"* (from HEAD of the trunk) in the branches subdirectory. Choose to stay on the trunk	Notice the speed of creating a branch
2. Update your working copy	
3. Modify *EditMe.txt* on the trunk. Change the last word from *"Subversion"* to *"SVN"*	
4. Commit your changes	
5. Switch your working copy to the newly created branch, *Demo*	Notice that the only change is the modified file
6. Modify *EditMe.txt* changing the last word from *'Subversion"* to *"Smart Subversion"*	
7. Commit your changes.	
8. Update your working copy	
9. Perform a merge between the trunk and the branch. Choose to resolve the resulting conflict by accepting the trunk revision.	Interactive conflict resolution.
10. Commit your changes.	

3. Lab 3 – Enterprise Features

3.1. Advanced Merge –Manual Merge

Steps	Comments
1. Using *JillDev*, create a new branch by the name of *Feature1* but keep your working copy referencing the trunk	Creating a branch for a new feature
2. Edit *First.txt* and change its contents however you wish	
3. Commit your changes	
4. Switch to the branch, *Feature1*	
5. Execute a manual merge between the trunk and the branch	
6. Examine *First.txt* to see that the changes were not made	
7. Commit	
8. Examine the mergeinfo data on the top directory	

3.2. Sparse Checkouts

Steps	Comments
1. Create a new working copy by checking out only the files beneath the trunk	Creating a branch for a new feature
2. Update the root directory to get all files and directories, but nothing below that	
3. Update to get the full structure beneath of the *Background* folder	
4. Add the full structure of the SecondLessons folder	

4. Solutions for Lab 1 – Essential Concepts 1

4.1. Suggested Solution for Standard Work Cycle – Part 1

Steps	Comments
1. 'svn co REPO_URL JoeDev' 2. 'svn co REPO_URL JillDev' 3. You may be prompted for authentication	SVN checkout fetches all the contents from the specified URL into the local directory. An example of URL is http://localhost/svn/LearningLab/trunk
4. Navigate to the *EasySVN* folder in the JoeDev working copy, use mkdir to create new directory, *FirstLessons* 5. Navigate into the *FirstLessons* directory and use an editor to create a new text document by the name *First.txt* 6. Navigate into the EasySVN directory 7. 'svn add *FirstLessons*'	Note that First.txt is also listed as added
8. 'svn del *DeleteMe.txt'*	Do not use the OS Delete as will not be changed as well
9. 'svn ren *RenameMe.txt ChangedName.txt'*	Do not use the OS Rename for the same reason as noted above for delete
10. Use your preferred editor to edit *EditMe.txt* and change "anything" to "something" on line 2 and save the file	Content change
11. Navigate to the top of your working copy and execute 'svn status'	High level checking
12. Navigate to the EasySVN directory and execute 'svn diff *EditMe.txt'*	
13. 'svn revert *ChangedName.txt'* and 'svn revert RenameMe.txt'	• Remember a Rename or copy is implemented as a combination of a delete operation and an add operation • You can choose Revert to undo any operation which has not been committed
14. 'svn update'	This operation will report the changes made to your working copy after updating it with the latest changes from the repository
15. At this point there are no conflicts, if there were the steps would be the same as defined for the merge process later	

16. 'svn commit –m "YOUR CONCISE AND USEFUL MESSAGE"'	• The commit will show all the changes that are being committed to the repository • Note the incremented global revision number after a successful commit
17. 'svn update'	This is to ensure that your working copy does not have mixed revisions.

4.2. Suggested Solution for Standard Work Cycle – Part 2

Steps	Comments
1. Navigate to the *JillDev* working copy and execute 'svn update'	You should always begin work with the latest revision from the repository
2. Navigate into the *EasySVN* folder and create a new directory, *SecondLessons* 3. Navigate into *SecondLessons* and create a new text document, *Second.txt* 4. Navigate up into EasySVN and execute 'svn add *SecondLessons*'	
5. 'svn mv *MoveMe AddMe'*	
6. 'svn lock *EditMe.txt* -m "Cause"'	Locking should normally be used for file types that can't be merged, but it may be used for other purposes
7. Edit the file "EditMe.txt" using your preferred editor to change "line" to "sentence" on the 2nd line and save the file	
8. 'svn status'	High level checking
9. 'svn update'	
10. Now there are no conflicts, if there were the same process would be used as defined for merge later	
11. Navigate to the top of the working copy and execute 'svn commit –m "more changes"'	• The commit will show all changes that are committed • Note the incremented global revision number after a successful commit
12. 'svn update'	This is to ensure that your working copy does not have mixed revisions

4.3. Suggested Solution for Examine History

Steps	Comments
1. 'svn log'	The entire trunk directory structure's (any revision where something changed) history will be displayed
2. Navigate to EasySVN and execute 'svn log *EditMe.txt*'	Show log will display only the history of the selected file
3. 'svn diff –r OLD:NEW *EditMe.txt*'	Diff will show the modifications made with respect to the two previous revisions
4. 'svn blame *EditMe.txt*'	Blame (or annotation) shows what revision and author last changed each line of a text file

4.4. Suggested Solution for Property Changes

Steps	Comments
1. 'svn propset svn:needs-lock * "Subversion logo.jpg"'	
2. Check the read/write access on the file	The read only attribute will not be set at this point, since the property is not yet committed
3. Navigate to the top of the working copy and execute 'svn commit –m "Make it require a lock"'	• The commit will show the change that was committed • Note the incremented global revision number after the commit
4. 'svn update'	This ensures that your working copy does not have mixed revisions
5. Check the read/write access on the file	Note that it is read-only now
6. 'svn lock "SubversionLogo.jpg" –m "Need it"'	This informs the Subversion server that you reserve the right to create the next revision of this file on the trunk
7. Check the read/write access on the file	Note that it is writeable
8. 'svn unlock "SubversionLogo.jpg"'	

5. Solutions for Lab 2 – Essential Concepts 2

5.1. Suggested Solution for Traversing

Steps	Comments
1. Navigate to JoeDev and execute “svn update –r 2 EditMe.txt’	
2. User your preferred editor to check the contents of the file to observe that the changes you made after revision 2 are not included	
3. ‘svn update –r 1 .’	
4. Observe the changes listed by the update operation’s output	
5. ‘svn update’	Content change

5.2. Suggested Solution for Tree Conflict

Steps	Comments
1. Navigate into the working copy JillDev and execute ‘svn update’	
2. Navigate into JoeDev, into EasySVN, and execute ‘svn mv *SecondLessons AddMe*’	
3. ‘svn commit –m “Moved”’	
4. Navigate to JillDev/EasySVN/SecondLessons and user your preferred editor to edit *Second.txt* 5. Type in “Tree conflict looming” and save the file	
6. Navigate to JillDev and execute ‘svn update’	Note the output shows a tree conflict on EasySVN\FirstLessons
7. Select P to get out of interactive conflict resolution 8. ‘svn status EasySVN\FirstLessons’	Note the output
9. Copy EasySVN\FirstLessons\First.txt to EasySVN\AddMe\FirstLessons (overwrite what’s there)	Resolution of tree conflicts isn’t fully automated as in this case you need to make a manual fix before utilizing the automated step provided for this type of tree conflict.

10. ‘svn resolve’ and enter r	
11. ‘svn commit –m “Change Added”’	

5.3. Suggested Solution for Merging

Steps	Comments
1. Navigate to JoeDev 2. ‘svn copy REPO_URL/trunk REPO_URL/branches/DEMO –m “New Demo branch”’	Note this does not change the reference point of the JoeDev working copy
3. ‘svn update’	
4. Navigate to EasySVN 5. Using your preferred editor modify *EditMe.txt* by changing the last word from “Subversion” to “SVN”	
6. Navigate to the top of the working copy and execute ‘svn commit –m “Fixed”’	
7. ‘svn switch REPO_URL/branches/DEMO’	Note that the only reported action is an update of the EditMe.txt file
8. Navigate to EasySVN and using your preferred editor change modify *EditMe.txt* by adding the word “Smart” before the last word “Subversion”	
9. Navigate to the top of the working copy and execute ‘svn commit –m “Modified”’	
10. ‘svn update’	
11. ‘svn merge REPO_URL/trunk’ 12. Select tc (their side of conflict)	
13. ‘svn commit –m “No conflict”’	

6. Solutions for Lab 3 – Enterprise Feature

6.1. Suggested Solution for Manual Merge

Steps	Comments
1. Navigate to JillDev 2. 'svn copy REPO_URL/trunk REPO_URL/branches/Feature1 –m "Branch to develop Feature 1"	
3. Using your preferred editor modify *First.txt* however you wish	
4. 'svn commit –m "My Changes"'	
5. 'svn switch REPO_URL/branches/Feature1'	
6. 'svn merge --record-only REPO_URL/trunk'	
7. Use cat on *First.txt* to see that the changes from the trunk were not applied	
8. 'svn commit –m "No change added"'	
9. 'svn propget svn:mergeinfo .'	Note the value of the svn:mergeinfo property showing the manual merge

6.2. Suggested Solution for Sparse Checkouts

Steps	Comments
1. Navigate to the parent folder of JoeDev and JillDev 2. 'svn co –depth=files REPO_URL/trunk JimDev'	Verify that you only get your new working copy directory with a .svn directory and the files directly under trunk
3. Navigate into JimDev and execute 'svn update --set-depth=immediates'	This option specifies to selectively include files at the top level of the branch and empty (excluding the .svn directories) folders beneath Verify that you got the expected results
4. 'svn update --set-depth=infinity Background'	• Verify that you got the expected results
5. 'svn update –set-depth=infinity SecondLessons'	• Verify that you got the expected results