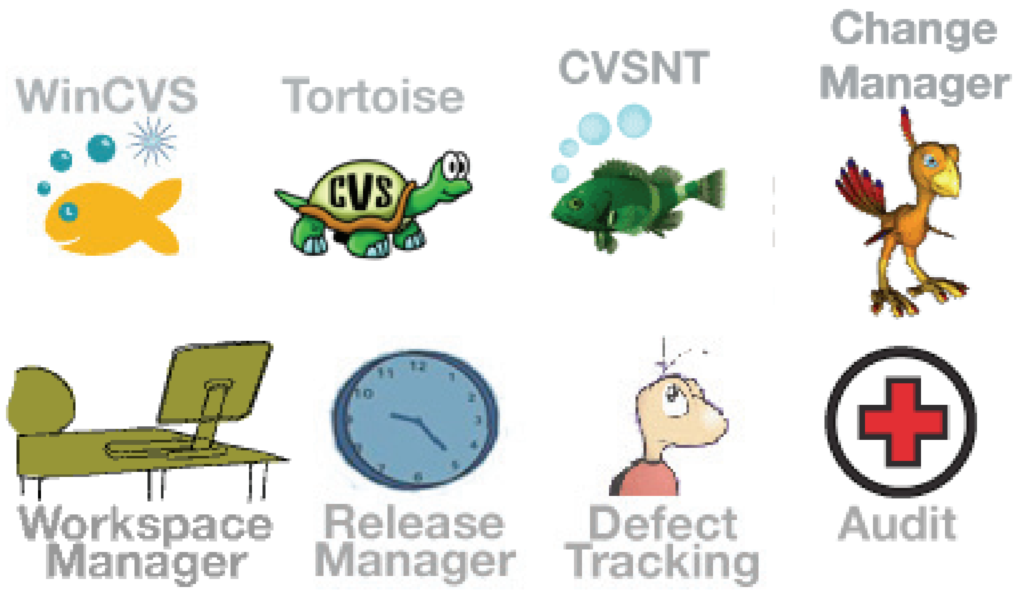
Promotion Model

CVS SUITE QUICK GUIDE **2009** Build 3701 **February 2010**



Optimized for

Microsoft Visual Studio

March Hare Software Ltd

Legal Notices

march-hare.com

sales@march-hare.com

Promotion model

A promotional model for managing changing documents or software is very common in many organizations. Using a promotion model it is easy to ensure that the correct people only “see” the documents and objects that are at the appropriate level of the promotion process.

Example 1

A government department is required to draft a new piece of legislation. The document evolves using a clearly defined model:

- Draft
- Legal Review
- Ministerial Approval
- Parliament

The document may go through several revisions during this entire process, however each time it is *promoted* to the next level it cannot be changed at that level except by authorised people. For example the public servant who authors the document cannot change the *Legal review* copy.

Frequently the document is only ever changed at the lowest level of the hierarchy, however meta data may be added at different levels. For example the legal review may wish to tag certain paragraphs as needing changes, or they may make the changes themselves.

It is important that the department that prints the legislation to table before parliament cannot accidentally print a copy that has not been through the entire promotion process.

Example 2

A software development company releases software four times a year. The software for each release evolves using a clearly defined model:

- Development
- Review
- Test
- Integration Test
- Production

The software goes through several revisions during this entire process, however each time it is *promoted* to the next level it cannot be changed at that level except by authorised people. For example the programmer who authors the bug fix cannot change the *Test* copy.

The software is only ever changed at the lowest level of the hierarchy, however meta data may be added at different levels. For example the code review may wish to tag certain functions as needing changes to comply with company coding standards, or they may make the changes themselves.

It is important that the users only run the Production version of the software and do not accidentally run another version. If a CD is produced and shipped out, the distribution department must have a fail safe way to ensure that they cannot accidentally deploy an untested release.

Mixed model

Mixing up the development models

Most organizations implementing CVS will want to use a mixture of the branching and promotional models. For example:

- Develop Version 1 on the Trunk
- When Version 1 is feature complete Branch Version 1 Maintenance
- Begin work on Version 2 on the Trunk
- Finalise Version 1 on the Branch
- Promote Version 1 Branch to Test
- Fix Version 1 bugs on the Version 1 Branch and promote to test again

Using a mixture of the techniques can lead to a balance with the strengths of both systems.

Regardless of the choices you make CVS is always capable of reapplying the changes between any two revisions to another revision – whether it is on the same branch or a different one.

Patch management – getting fixes to customers

The CM model that you choose may be heavily influenced by business concerns such as needing to deliver fixes to current software while also allowing development to continue on newer versions. This is known as *patch management*.

Service Packs

Often organizations deliver stable combinations of patches to customers as a service pack – or a point release. For example, version 1 is released and several bugs are found and fixed, and three months after the first release version 1.1 (or version 1 service pack 1) is released containing all bug fixes.

This service pack example can also be described in a time line similar to:

- Version 1
- Fix bug 1
- Fix bug 2
- Fix bug 3
- Release Version 1.1

Patches

What happens if one of those bugs seriously effects the customer’s current day-to-day operations? In this case it may be necessary to release one of the bug fixes immediately. Since most customers are not affected the release version 1.1 is not created earlier – but a patch is produced.

This patching example can also be described in a time line similar to:

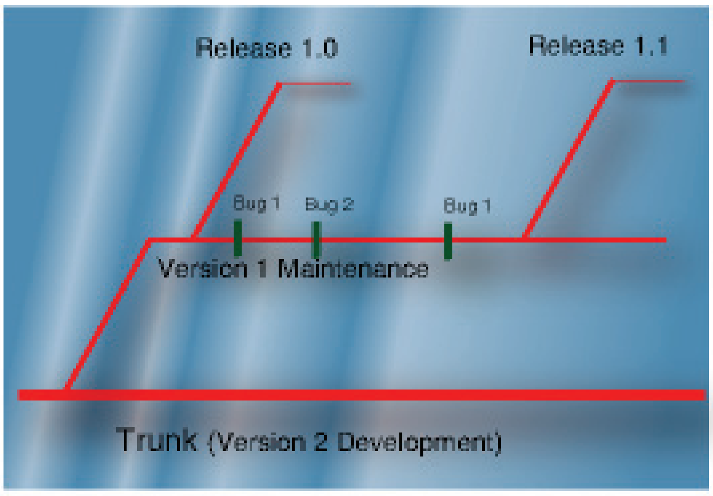
- Version 1
- Fix bug 1
- Fix bug 2
- Release Patch 1
- Fix bug 3
- Release Version 1.1

However the customer does not want any changed functionality *except* their bug fix. In the list above it can be seen that bug fix 1 and bug fix 2 have already been completed and a combined Release Patch 1 contains both fixes. A typical development environment may have made 50 changes, and the customer does not want the responsibility of testing all those fixes to get their environment working again.

The software company needs to balance the customer's requirements with their own. Specifically they need to ensure that the change is only made once but it is then applied to release 1.1 and also later to release 2.0.

Careful consideration of the business requirements is necessary to design an effective CM process. CVS is technically capable of supporting all these decisions.

In this particular example the choices the software organization would make would depend largely on the frequency and the billing methods. If these *individual patch releases* are rare or are charged to the customer then they will be designed as an exception. If they are common then the SCCM solution will be designed with some level of automation for reliability and reproducibility.



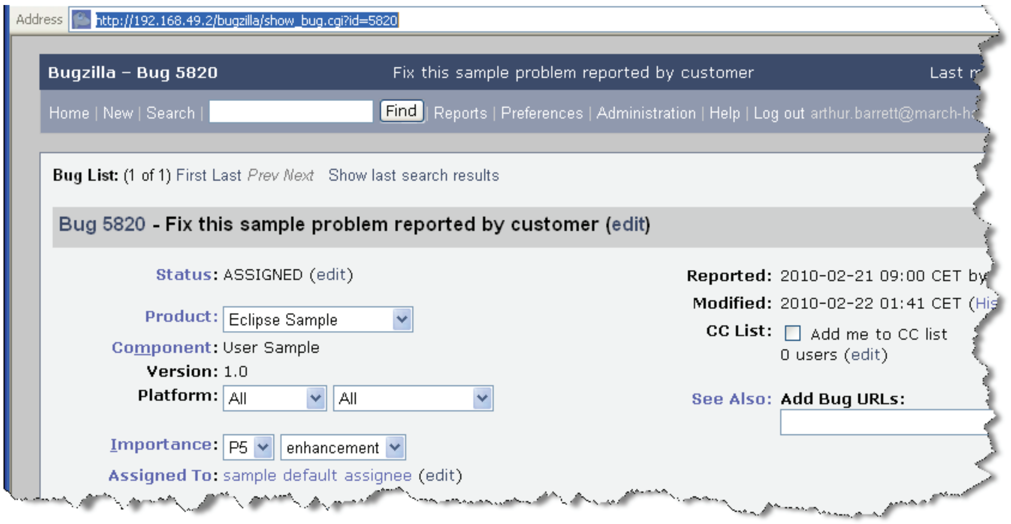
Bug fixing workflow with Promote

A typical request from QA managers is how to track what files have changed for each job or bug. The following section describes an example of this:

- Make a change to source code using Eclipse and commit to repository.
- View changes via Bugzilla
- View changes via SQL Query on Audit database or Bugzilla database
- View changes via CVS Suite command line client

Make changes to source code using Eclipse and commit to repository

In a typical commercial software development environment a programmer is given a task number, job number or bug number before beginning work. This bug number is then used to track the changes from requirements gathering through to release:



Address http://192.168.49.2/bugzilla/show_bug.cgi?id=5820

Bugzilla – Bug 5820 Fix this sample problem reported by customer Last m

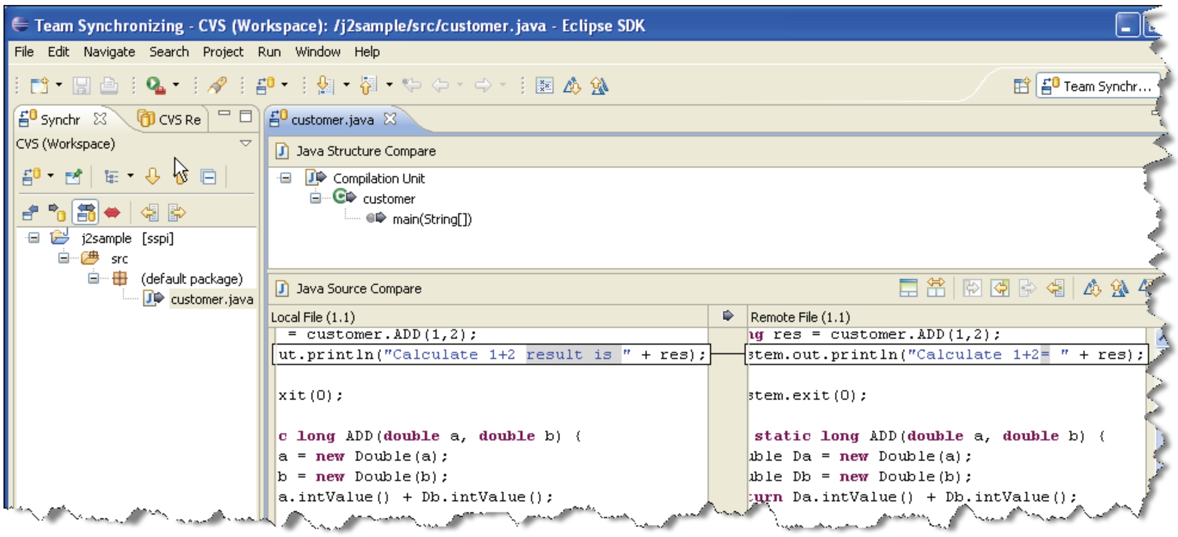
Home | New | Search | Find | Reports | Preferences | Administration | Help | Log out arthur.barrett@march-h

Bug List: (1 of 1) First Last Prev Next Show last search results

Bug 5820 - Fix this sample problem reported by customer (edit)

Status:	ASSIGNED (edit)	Reported:	2010-02-21 09:00 CET by
Product:	Eclipse Sample	Modified:	2010-02-22 01:41 CET (His
Component:	User Sample	CC List:	Add me to CC list 0 users (edit)
Version:	1.0		
Platform:	All / All	See Also:	Add Bug URLs:
Importance:	P5 / enhancement		
Assigned To:	sample default assignee (edit)		

The developer can use the synchronize view to visually compare the changes in the eclipse workspace with the CVS Suite Server repository:



Team Synchronizing - CVS (Workspace): /j2sample/src/customer.java - Eclipse SDK

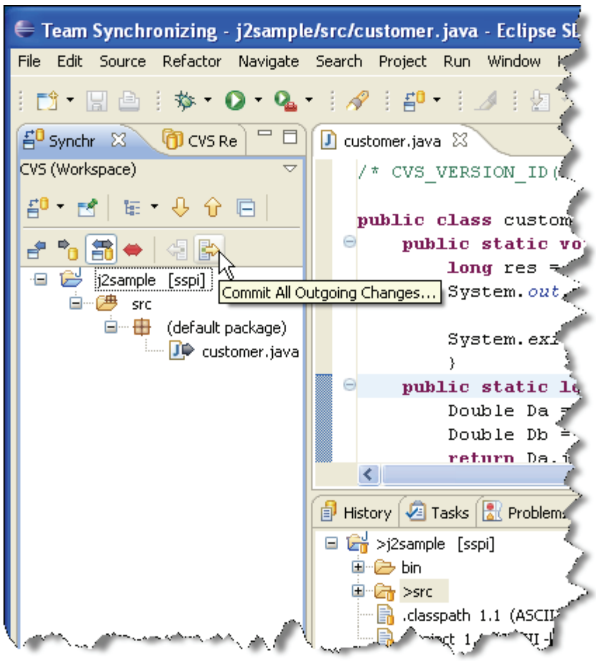
Local File (1.1):

```
 = customer.ADD(1,2);
ut.println("Calculate 1+2 result is " + res);

xit(0);

c long ADD(double a, double b) {
a = new Double(a);
b = new Double(b);
a.intValue() + Db.intValue();
```

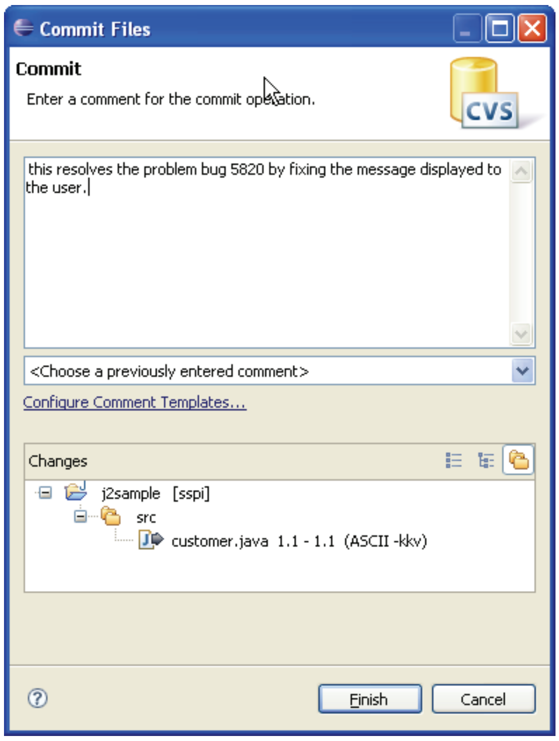
Remote File (1.1):

```
ng res = customer.ADD(1,2);
stem.out.println("Calculate 1+2= " + res);

stem.exit(0);

 static long ADD(double a, double b) {
uble Da = new Double(a);
uble Db = new Double(b);
urn Da.intValue() + Db.intValue();
```

When the developer is finished with the changes they can commit them back to the repository and link them to the job / bug using the bug number:


Team Synchronizing - j2sample/src/customer.java - Eclipse SD
File Edit Source Refactor Navigate Search Project Run Window
Synchr
CVS Re
CVS (Workspace)
j2sample [sspi]
Commit All Outgoing Changes...
src
(default package)
customer.java
History
Tasks
Problems
>j2sample [sspi]
bin
>src
.classpath 1.1 (ASCII

```
/* CVS_VERSION_ID (
public class custom
    public static vo
        long res =
        System.out
        System.exi
        }
    public static lo
        Double Da =
        Double Db =
        return Da.i
```

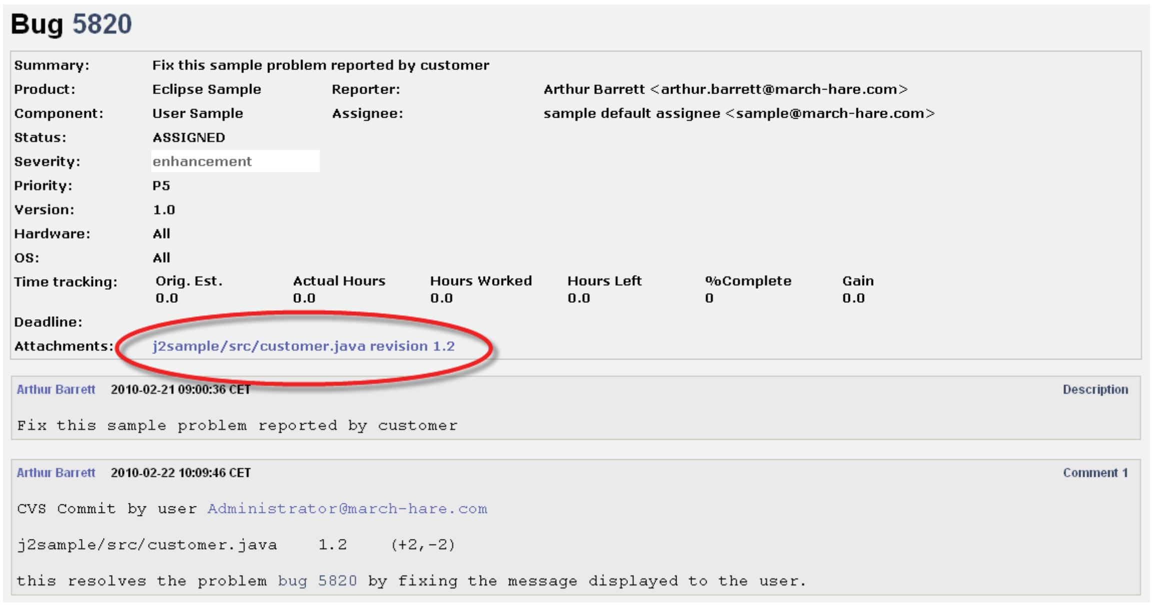

Commit Files
Commit
Enter a comment for the commit operation.
CVS
<Choose a previously entered comment>
Configure Comment Templates...
Changes
j2sample [sspi]
src
Finish
Cancel

this resolves the problem bug 5820 by fixing the message displayed to the user.

customer.java 1.1 - 1.1 (ASCII -kkv)

View changes in Bugzilla

The QA manager can see at a glance in Bugzilla all the files modified by any bug:



Bug 5820

Summary:	Fix this sample problem reported by customer		
Product:	Eclipse Sample	Reporter:	Arthur Barrett <arthur.barrett@march-hare.com>
Component:	User Sample	Assignee:	sample default assignee <sample@march-hare.com>
Status:	ASSIGNED		
Severity:	enhancement		
Priority:	P5		
Version:	1.0		
Hardware:	All		
OS:	All		

	Orig. Est.	Actual Hours	Hours Worked	Hours Left	%Complete	Gain
Time tracking:	0.0	0.0	0.0	0.0	0	0.0

Deadline:

Attachments: j2sample/src/customer.java revision 1.2

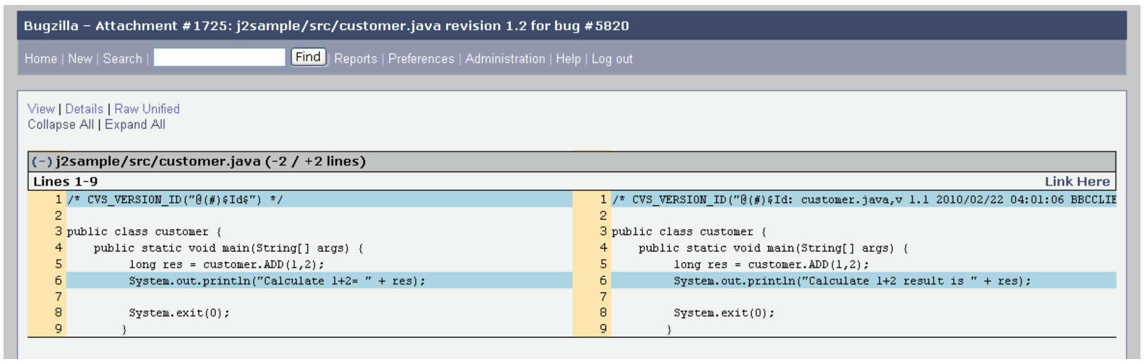
Arthur Barrett 2010-02-21 09:00:36 CET — Description

```
Fix this sample problem reported by customer
```

Arthur Barrett 2010-02-22 10:09:46 CET — Comment 1

```
CVS Commit by user Administrator@march-hare.com

j2sample/src/customer.java    1.2    (+2,-2)

this resolves the problem bug 5820 by fixing the message displayed to the user.
```

The diffs can even be code reviewed directly in Bugzilla:



Bugzilla – Attachment #1725: j2sample/src/customer.java revision 1.2 for bug #5820

Home | New | Search | Find | Reports | Preferences | Administration | Help | Log out

View | Details | Raw Unified
Collapse All | Expand All

(-) j2sample/src/customer.java (-2 / +2 lines)

Lines 1-9 Link Here

```
/* CVS_VERSION_ID("@(#)$Id$") */

public class customer {
    public static void main(String[] args) {
          long res = customer.ADD(1,2);
          System.out.println("Calculate 1+2= " + res);

          System.exit(0);
        }
```

```
/* CVS_VERSION_ID("@(#)$Id: customer.java,v 1.1 2010/02/22 04:01:06 BBCCLIE

public class customer {
    public static void main(String[] args) {
          long res = customer.ADD(1,2);
          System.out.println("Calculate 1+2 result is " + res);

          System.exit(0);
        }
```

View changes via SQL Query on Audit database or Bugzilla database

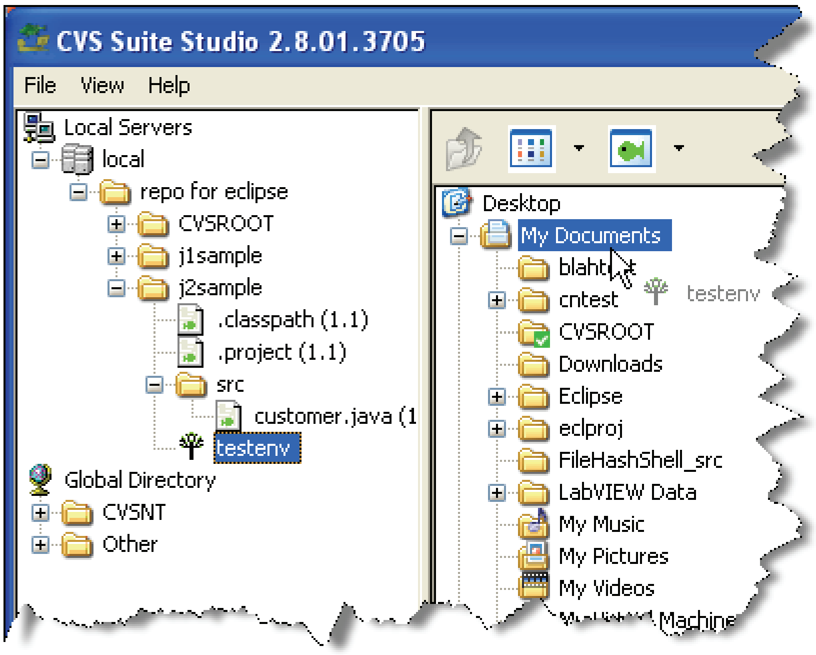
An SQL client (eg: Microsoft Excel) can query the audit database or the Bugzilla database to find all the files changed by a bug or all of the bugs affected by a file.

Promote to test or production by bug number

The CVS Suite tools all understand the bug number, and you can promote to any defined promotion level using a bug number, eg: promoting to test:

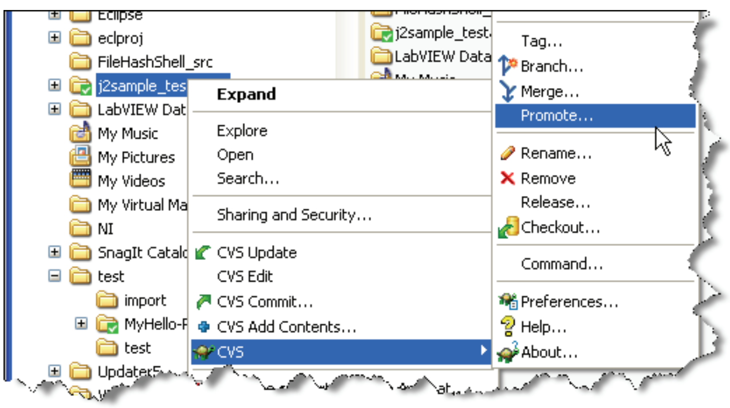
Firstly the QA manager creates a disk area to perform the promote:

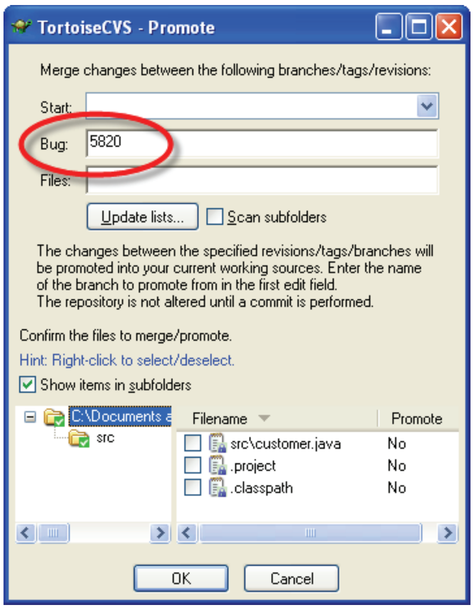
Drag the promotion level to a secure location:



Secondly use the right click promote menu

Right click the new promotion directory and choose the bug number to promote – there is no need to know what files are affected by the bug – all files affected are promoted:




TortoiseCVS - Promote

Merge changes between the following branches/tags/revisions:

Start:

Bug: 5820

Files:

Update lists... ☐ Scan subfolders

The changes between the specified revisions/tags/branches will be promoted into your current working sources. Enter the name of the branch to promote from in the first edit field.
The repository is not altered until a commit is performed.

Confirm the files to merge/promote.
Hint: Right-click to select/deselect.
☑ Show items in subfolders

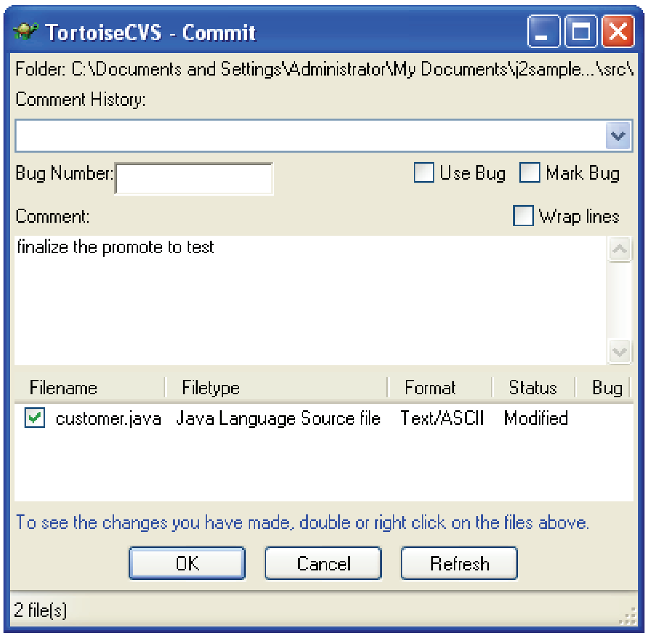
C:\Documents a
src

Filename	Promote
☐ src\customer.java	No
☐ .project	No
☐ .classpath	No

OK Cancel

Lastly use the right click commit menu

Right click the new promotion directory and choose commit – you can use the same bug number, or a new bug number (eg: a new bug that combines several other bug fixes into a release or service pack):


TortoiseCVS - Commit

Folder: C:\Documents and Settings\Administrator\My Documents\j2sample...\src\
Comment History:

Bug Number: ☐ Use Bug ☐ Mark Bug

Comment: ☐ Wrap lines

finalize the promote to test

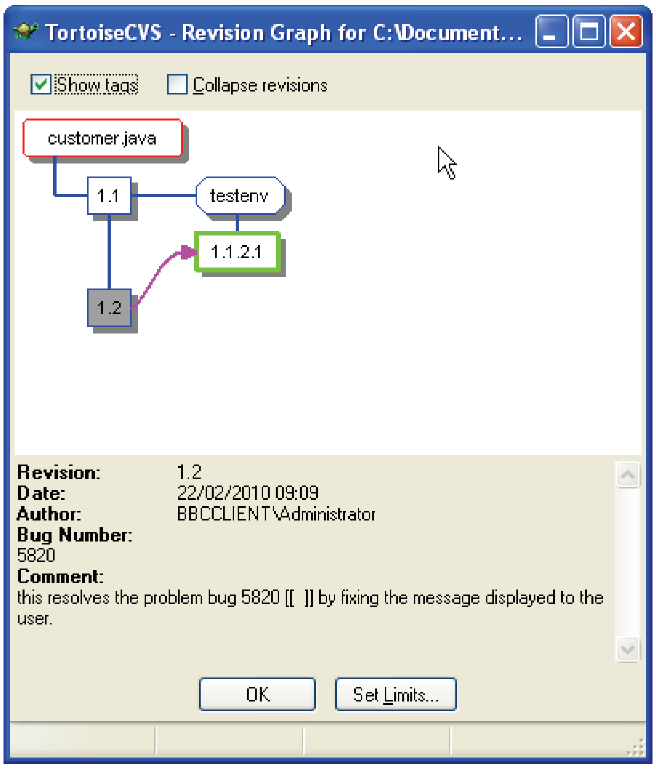
Filename	Filetype	Format	Status	Bug
☑ customer.java	Java Language Source file	Text/ASCII	Modified	

To see the changes you have made, double or right click on the files above.

OK Cancel Refresh

2 file(s)

You can use the revision graph feature to visually confirm the promotion:

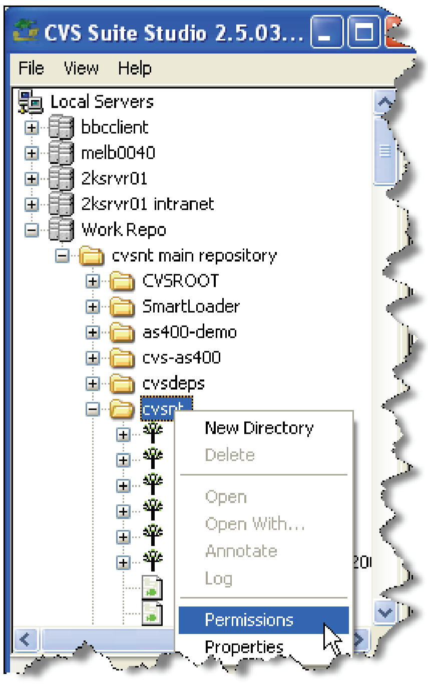


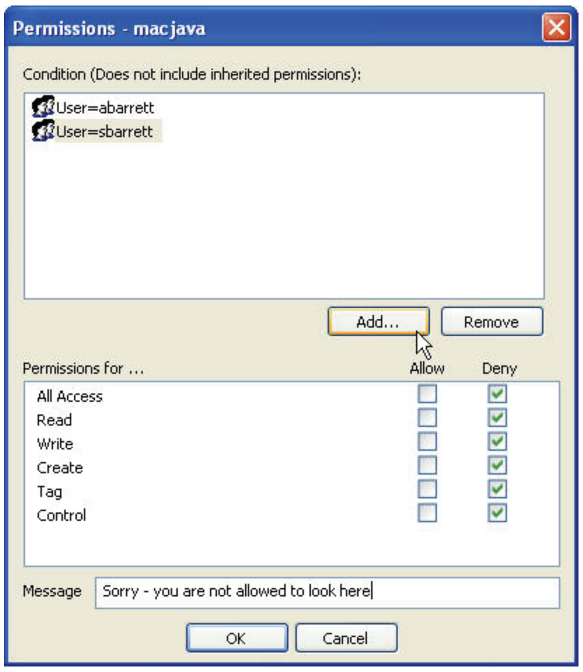
Revision: 1.2
Date: 22/02/2010 09:09
Author: BBCCLIENT\Administrator
Bug Number:
5820
Comment:
this resolves the problem bug 5820 [[ ]] by fixing the message displayed to the user.

Set and Manage ACL's

CVS Suite Studio is the recommended tool graphical front end for CVSNT and provides a range of powerful commands for performing the most common CVSNT tasks in a clear and simple user interface.

To use ACL's you first must configure the ACLMode, see the Administration section for more information. To deny access by default set `aclmode = normal` (if you do not set this CVSNT will allow access by default). Once the repository wide default is set, you can then use CVS Suite Studio to add or remove permissions to folder on the server:





Condition (Does not include inherited permissions):

User=abarrett
User=sbarrett

Permissions for ...	Allow	Deny
All Access	☐	✓
Read	☐	✓
Write	☐	✓
Create	☐	✓
Tag	☐	✓
Control	☐	✓

Message: Sorry - you are not allowed to look here