Eclipse and CVS Suite

WHITEPAPER 2009.3712 February 2010

March Hare Software Ltd

Legal Notices

march-hare.com

consultants@march-hare.com

Table of Contents

Overview

This document gives a practical and technical overview of how to use Eclipse and CVS Suite 2009 together, and in particular:

- Development integration with Eclipse
- Support for a Dev -> Test -> Production development cycle
- Integration with change management using a defect tracking system such as Bugzilla, Mantis or Atlassian Jira
- Set up Eclipse client for connection to CVS Suite Server
- Use TortoiseCVS on the same workspace/sandbox as Eclipse
- Add an existing Eclipse project to CVS or checkout a CVS project from Eclipse
- Common problems including working with tags and branches

Additionally CVS Suite 2009 provides support for these common Software Change and Configuration Management requirements not directly addressed by this document:

- Failsafe audit to an enterprise database
- Fine grained Access Control
- Event based Build management
- Secure Enterprise Authentication including password-less token based
- Encrypted communications
- Project Management – facilitation of multiple baselines for development
- Concurrency control (to allow reserved or unreserved workflows)
- Team communication with automated e-mail notification
- Automated Backup and disaster recovery
- Centralised management of SCCM system configuration

History of CVSNT and Eclipse

All versions of CVSNT since 2.0.58b have been tested for compatibility with Eclipse. See the Eclipse project page Eclipse Compatibility with CVSNT and Eclipse 3.1 New and Noteworthy - Part 1 (look for the heading *Support for CVSNT*).

CVS Suite 2009 Server supports Eclipse clients.

The following limitations of Eclipse clients are known:

- Cannot restore deleted resources from within Eclipse. This makes it impossible to "undo" a delete from within Eclipse. The workaround is to use TortoiseCVS or CVSNT command line client to restore the deleted file. The topic is covered by Eclipse bug 75912.

Development integration with Eclipse

Eclipse software incorporates some Team features where developers checkin/checkout files that they are working on. Eclipse supports version control systems such as CVS Suite 2009 which can be integrated with the Team features:

- Copies of Project files (java classes, java methods etc) are stored in CVS Suite
- The person making the change using Eclipse is prompted for a comment
- CVS Suite Server and optionally Bugzilla, Mantis or Atlassian Jira defect tracking is automatically updated when a Project file is modified using Eclipse

Support for a Dev -> Test -> Production development lifecycle

CVS Suite supports a promotion model based system for tracking and managing components over the development lifecycle:

- Unlimited flexibility with user definable levels
- Fine grained access control lists manage who can promote to each level and from what lower levels they can promote
- Managers can script actions to occur at specific promotion events, including updating different deployment environments or test environments when project files are promoted

Integration with Change Management using a Defect Tracking System, e.g.: Bugzilla

CVS Suite supports integration with defect tracking systems such as Bugzilla, Mantis and Atlassian Jira to enhance the change management features:

- The defect tracking system allows you to search for changes by change number/job number, username, filename and comments made when the change was committed to the CVS Suite Server
- CVS Suite Server inspects commit or check in comments each time a change is sent to the server for linking to a job or bug
- Commit or check in comments can be used to assign or close bugs in Bugzilla

Installation

This whitepaper is not a comprehensive guide to installing CVS Suite. For a more detailed explanation refer to the documentation that accompanies the software, the ebook: *All About CM & CVS*.

Installed Components

The following software was installed in the PC environment used for this whitepaper:

Software Installation		
	Version	*Location*
CVS Suite Server	2009.3712	`D:\program files\cvs suite\cvsnt\`
Eclipse	3.1 *(other versions also tested)*	`W:\eclipse-3.01\`
Operating System *(for both CVS Suite Server and Eclipse client)*	Windows XP	`D:\windows`
Bugzilla	3.4.5	`/Library/WebServer/Documents/bugzilla` *(on Mac OS X 10.5.8)*
MySQL Server	5.1.43	`/usr/local/mysql` *(on Mac OS X 10.5.8)*

Contacting Sales

If you would like more information please contact us:

United Kingdom

March Hare Software Limited
85-87 Bayham Street
Camden Town
London NW1 0AG
United Kingdom

Ph:	+44 (0)207 692 0712

USA & Canada

March Hare Software LLC
200 Broadhollow Road
Suite 207
Melville NY 11747
United States

Ph:	1-800-653-1501

You can also send messages electronically. To ask for a sales person to contact you please send email to:

sales@march-hare.com

Fundamentals

What is CVSNT and CM Server all about

This software helps computer users keep track of changes to files.

All of the things you create on your computer: Documents, Program Source Code, Web Pages, Pictures, Spreadsheets can be managed using CVS Suite or CM Server.

In addition to tracking the changes, the software also can provide assistance with publishing, reviewing, securing and managing those files and the ability for different computer users to make changes at different stages of the documents life.

CVS Suite was originally designed for tracking changes to files and documents written by computer programmers: computer source code. This is still the primary use of CVS Suite and the focus of this paper, though the same procedures can be used for managing any type of file.

CM Server is a more advanced edition of the software and addresses the requirements of larger teams and organisations.

Can I Install the Software and Read this Later

Effectively managing your computer files and the changes to them largely depends on how you work and what your priorities for management are.

If you attempt to use the software without understanding the theory first then you will almost certainly find it is not optimal for your purposes.

Therefore you are encouraged to read the theory before attempting the practical.

I Don’t Like Version Control

We use the term *Effective Configuration Management* often during our on site consulting. As an organisation it took us at March Hare Software Ltd a long time to discover that there is a difference between Configuration Management and Effective Configuration Management.

Most people who do not like version control feel that way because they have been exposed to it in an ineffective environment. Spending time doing things that are ineffective will lead to an enormous level of frustration.

If you don’t like version control, please read this whitepaper and also our comprehensive guide to SCCM theory: *All About CVS* and look for a process that you would be happy to use to effectively manage your work.

I prefer to use some other tool

CM Suite is the ideal server software for people who prefer to use other tools as it is client agnostic. Client agnostic CM Server is ideal for a heterogeneous CM environment, allowing each person to choose the tool most effective for them whilst not limiting the choices of other people and retaining the ability to comply with audit and management objectives. CM Suite supports most popular Version Control and SCCM client tools.

Theory

Promotion model

A promotion 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 metadata 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 metadata 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.

What are Branches, Magic Branches and Vendor Branches

Branches, Magic Branches and Vendor Branches facilitate software development project management. Often software development managers indicate that they do not require branches in their solution. Take the time to carefully read this section since once they understand what branches, magic branches and vendor branches are and what they can facilitate most customers find that these facilities are very useful and very powerful.

When are Branches, Magic Branches and Vendor Branches Used

Vendor Branches, Magic Branches and Branches are used whenever the evolution of changes to the documents are not sequential.

Example 1

A manager writes the outline of the current product specification for what the business does and gives it to the marketing department so they can develop a "what we do" document for sales people. However the manager has been instructed to also prepare for a new venture in the near future so after sending the document to the marketing department the manager begins to modify it again to bring it up to date with the new plans.

While the manager updates the document, the marketing department begin to change the wording of the document to make it more suitable for a lay audience, adding pictures and changing the formatting. The same document now has *two streams of development*.

Example 2

A software company releases version 1 of their software and immediately begins work on version 2. However the sales department sell the software to a company who discovers that a part of the application has a bug. Version 2 will not be ready for weeks yet, however the customer requires a fix for the broken function much sooner. The software source code now requires *two streams of development*.

Example 3

A freight company uses a software package to track cargo around the country; however the software uses terminology that is different to what the company uses. The names are stored in configuration files that the freight company modified to change the names to more appropriate ones.

However the software company release version 2 with many new features that the freight company want, but it has a new configuration files with a lot of new information.

The configuration files have two independent *streams of evolution*.

Example 4

A web designer manages seven web sites that are identical to each other except for a few files on each. About 90% of the web sites share the same PHP code. The web site has a single stream of development with *multiple variations*.

Example 5

A software vendor manages configuration files for their software as used by the most important 20 customers. The configuration files have a single stream of development with *multiple variations*.

What are the benefits to using Branches, Magic Branches and Vendor Branches

In each of the above examples the organisation finds that the documents do not have a sequential evolution of development but there are *more than one stream of development* (or a primary stream and multiple *variations*). CVS uses branches to track how these changes are made.

However using branches can also offer your organisation some advantages because CVS can automate some of your business activities.

Making the same changes twice

One of the benefits to using good configuration management tools is that if you need to re-apply textual changes made to a document in one stream of development to another stream then it can be automated.

For instance, in example 2 above the software developer can make the “bug fix” in “version 2” and use automated techniques to re-apply the same change to the “maintenance version”.

In the example 3 the freight company can use a vendor branch to apply the vendors changes to the configuration files that they are using and therefore keep their labels.

Changes to binary files (eg: word documents or pictures) cannot be replicated automatically.

Ensure security

If a document is evolving on more than one stream of development then it is possible that one or more have different security requirements. For instance a software vendor may develop enhancements to their application for two customers who are competitors. In this case it may be necessary for the developers making changes for customer A not to be able to see customer B’s changes.

In example 1 above, the developers from the “version 2” team may not be allowed to make changes to the more stable “version 1”, so it is possible to secure the access of the two groups based on the branch.

What is different between a Branch and a Magic Branch

Magic branches are used where the documents as a collection exist as several *variations*, however as individual documents the majority are all identical. This is common for configuration files or data (or configuration) driven web sites. Two Active Server Pages web sites may exist that are identical except for a few graphics files and the `locals.inc`, which sets the title for the pages and the name of the database to get the data from.

Magic Branches organise development so that the site-specific configuration can be developed separately to the web site framework. Provided that the web site framework is never modified on the branch, the magic branch always will contain the same files as the trunk.

Mixed model – Branching and Promoting

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:

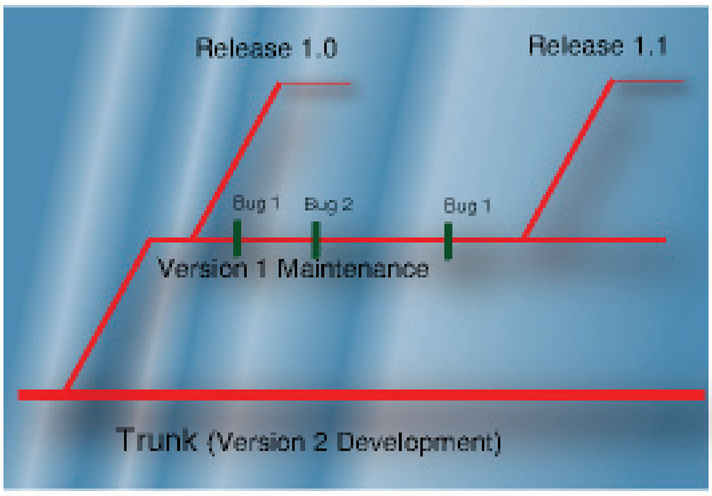
- Version 1
- Fix bug 1
- Fix bug 2
- Release Patch 1
- Fix bug 3
- Release Version 1.1

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.



What is the Repository and the Workspace

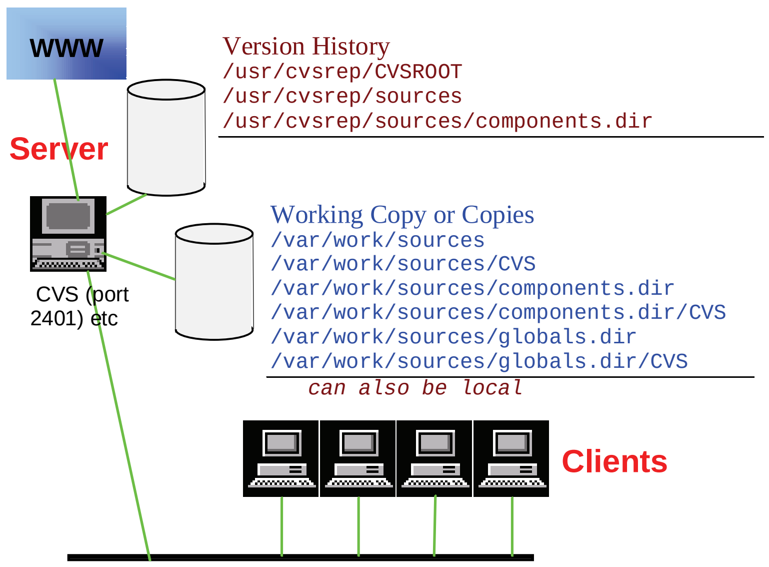
These terms are commonly used in business English but have specific technical meanings for our products and in this whitepaper:

Repository Version History

A repository is a collection of versioned objects with similar business rules managed by our server software in one or more physical locations. E.g.: the method `gencustno` is a versioned object and all instances of it (including dev/test/prod) would be in a single server repository in one or more physical locations.

A repository may contain versioned objects from different projects, eg: `CRM_PROJ`, `ORDER_INV_PROJ`, `STATS_PROJ` etc

Versioned objects from different projects that vary only by role not function (e.g.: `crm_proj_dev` and `crm_proj_test`), will always be stored in a single repository.



Working Copy *or* Workspace

A working copy (often referred to as a workspace) is a collection of instances of versioned objects outside of the repository. Some people refer to a workspace as *checked out copies* or as a *sandbox*.

The most common storage locations for a working copy are a disk (e.g.: `My Documents`) or a shared disk (e.g.: `e$ on myfsserver` or `\\myfsserver\\e$\`).

Whilst a working copy is always *linked* back to a repository, making a change in a working copy is controlled by the native permissions of target device (e.g.: a disk or a network share). Changes to a working copy may be discarded. Changes to a working copy may be kept (i.e.: committed) back to the repository only if the permissions in the repository allow changes by that person and to that instance (e.g.: John may be allowed to commit changes to `custidproj` in dev but not in prod).

CVS Suite and Eclipse Workflow Setup

This section is intended to give the practical workflow when working with one of the popular methodologies supported by CVS Suite.

Workflow Definition

There are many techniques to define the workflow of your organisation in CVS Suite. The following rules have been defined:

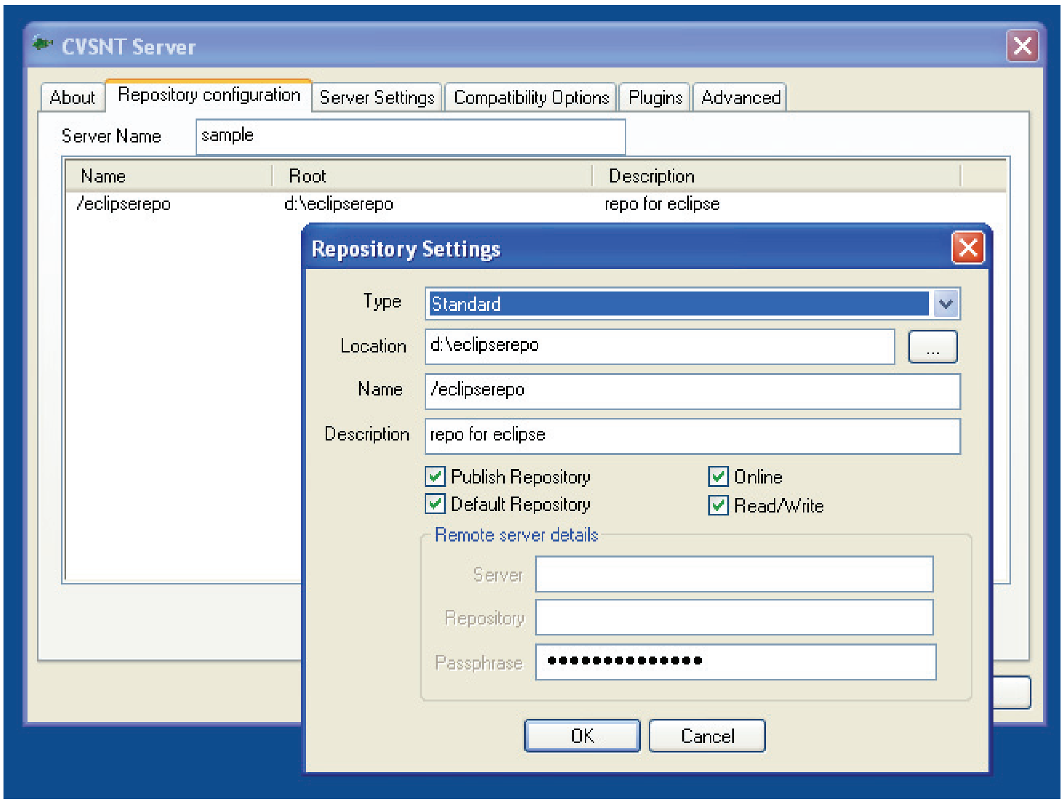
- A Promotion Model of Trunk (Dev) -> Test -> Prod
- Three project workspaces: Example for Dev, Test and Prod
- Two privileged users for updating Test and Prod on promote
- A script to automatically update Test or Prod on promote
- CVS Suite server integration with Ant or Make
- CVS Suite server integration with Bugzilla defect tracking
- CVS Suite client Integration with Eclipse

The build scripts are in reproduced in the appendix of this whitepaper.

CVS Suite Server Repository

Creating the CVS Suite Server Repository

The CVS Suite Server Control Panel is used by the Administrator to create the repository on the CVS Suite Server:



Server Name: sample

Name	Root	Description
/eclipserepo	d:\eclipserepo	repo for eclipse

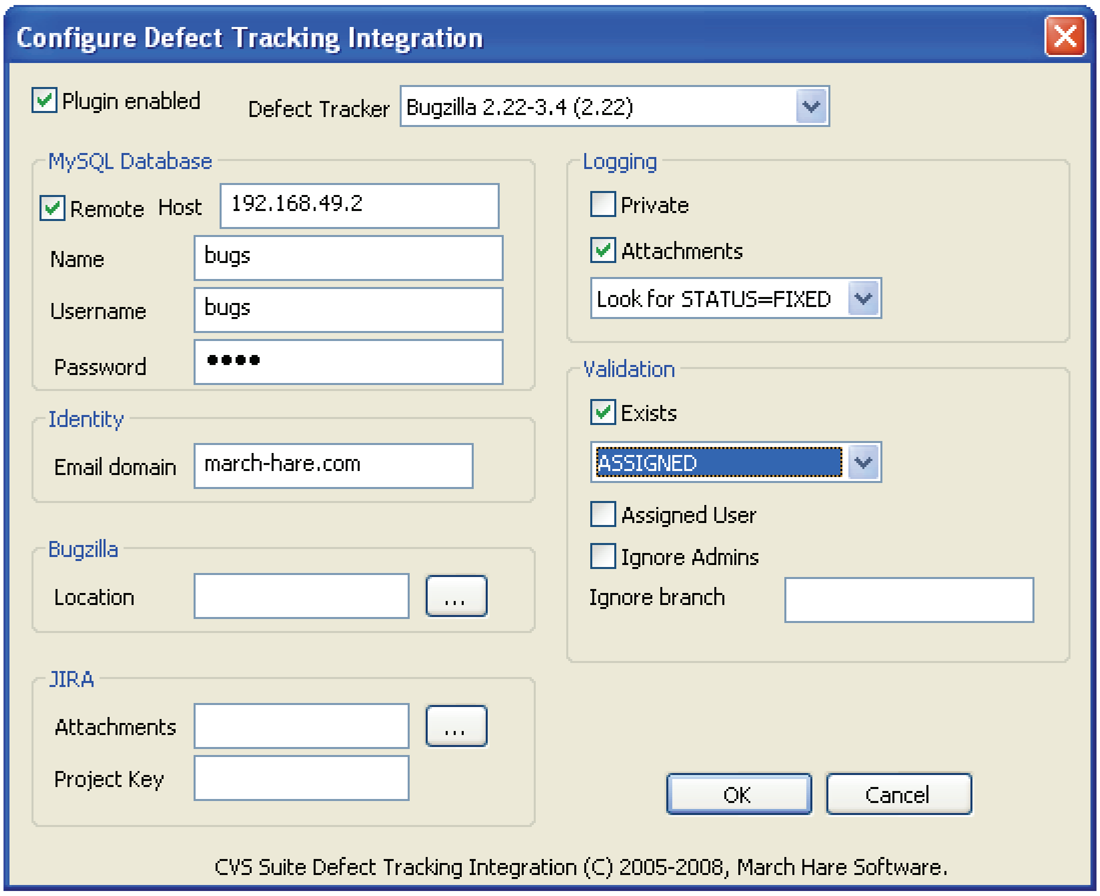
Type: Standard
Location: d:\eclipserepo
Name: /eclipserepo
Description: repo for eclipse
[✓] Publish Repository
[✓] Default Repository
[✓] Online
[✓] Read/Write
Server:
Repository:
Passphrase: •••••••••••••••

An alternative command line interface is available for unix, linux and mac. Refer to the eBook *All About CM & CVS* for more information for both Windows and unix.

Defect Tracking Integration

Activating the CVS Suite Server connection to Bugzilla

The Defect Tracking Plugin, available in the CVS Suite Server Control Panel is used to activate the connection between CVS Suite Server and Bugzilla:



Configure Defect Tracking Integration

[✓] Plugin enabled

Defect Tracker: Bugzilla 2.22-3.4 (2.22)

MySQL Database

[✓] Remote Host	192.168.49.2
Name	bugs
Username	bugs
Password	••••

Identity

Email domain	march-hare.com

Bugzilla

Location	

JIRA

Attachments	
Project Key	

Logging

[ ] Private

[✓] Attachments

Look for STATUS=FIXED

Validation

[✓] Exists

ASSIGNED

[ ] Assigned User

[ ] Ignore Admins

Ignore branch:

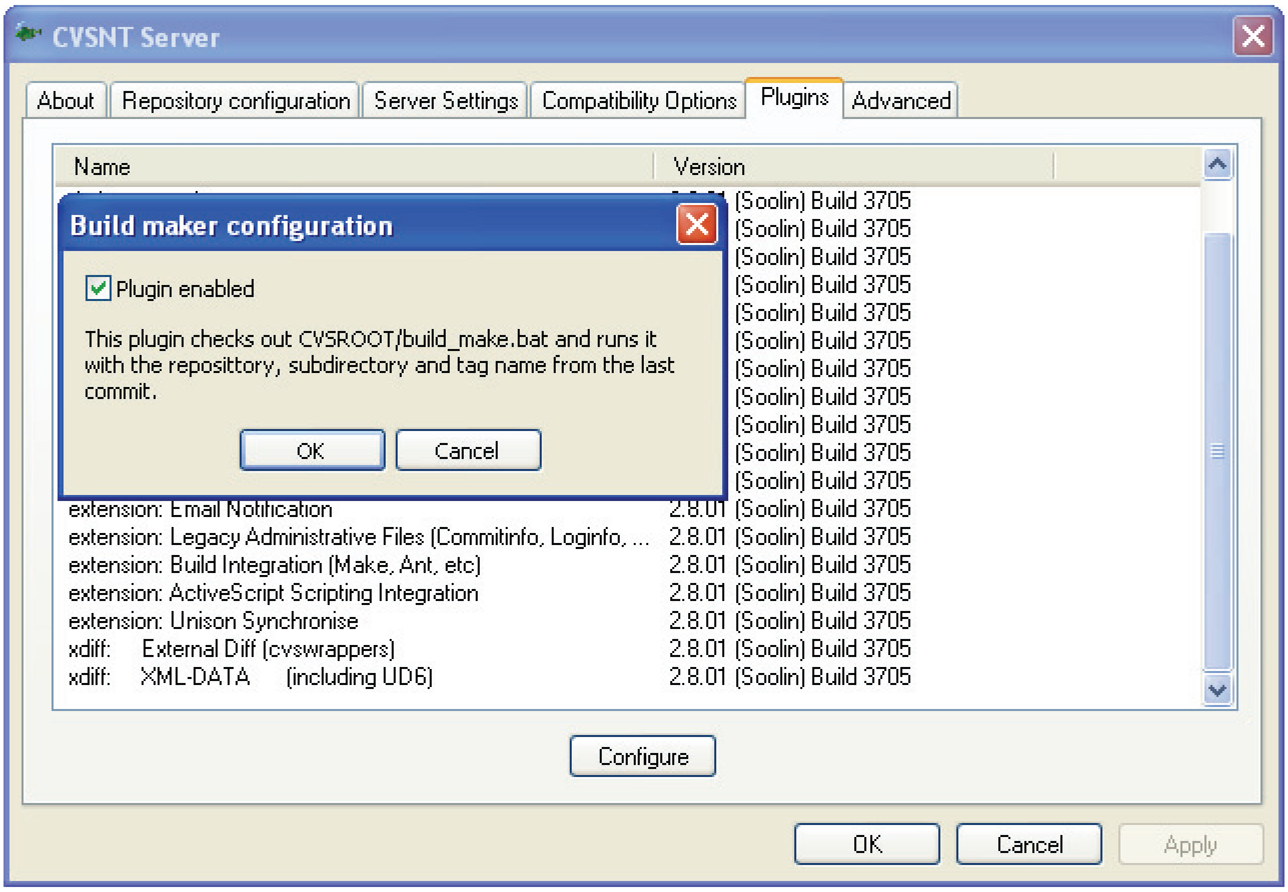
CVS Suite Defect Tracking Integration (C) 2005-2008, March Hare Software.

On unix, linux and Mac systems the configuration is done using the /etc/cvsnt/Bug configuration file. Refer to the eBook *All About CVS* for more information for both Windows and unix.

Build and Promote Integration

Activating the Promotion connection to Ant (or make)

The following steps activate the Promote connection between CVS Suite Server and Ant (or make):



This plugin checks out CVSROOT/build_make.bat and runs it with the reposittory, subdirectory and tag name from the last commit.

Name	Version
extension: Email Notification	2.8.01 (Soolin) Build 3705
extension: Legacy Administrative Files (Commitinfo, Loginfo, ...	2.8.01 (Soolin) Build 3705
extension: Build Integration (Make, Ant, etc)	2.8.01 (Soolin) Build 3705
extension: ActiveScript Scripting Integration	2.8.01 (Soolin) Build 3705
extension: Unison Synchronise	2.8.01 (Soolin) Build 3705
xdiff: External Diff (cvswrappers)	2.8.01 (Soolin) Build 3705
xdiff: XML-DATA (including UD6)	2.8.01 (Soolin) Build 3705

Also see the `build_make.bat` configuration file (for windows CVSNT Server) or the `build_make.sh` configuration file (for unix/linux/mac) in the appendix.

Eclipse Integration Setup

Eclipse is a popular IDE / Editor for use in Java programming environments; it is also the basis of the IBM WebSphere Application Developer (RAD).

Activating the Team connection to CVS Suite

The steps described in this section activate the Team connection between Eclipse and CVS Suite.

Eclipse 3.1, 3.3.2, 3.4.2 (Ganymede) and 3.5.x (Galileo)

These examples are created using Eclipse SDK 3.1. Eclipse 3.3.2 has also been tested using these instructions. Eclipse 3.4.2 (Ganymede) and 3.5.x (Galileo) have also been tested using these instructions.

Eclipse 2.1.3

Eclipse 2.1.3 has also been tested using these instructions (see notes on versions of Eclipse prior to 3.1). Specific instructions are not provided for using CVS with Eclipse 2.1.3,

Checkout out a project using Eclipse 2.1.3

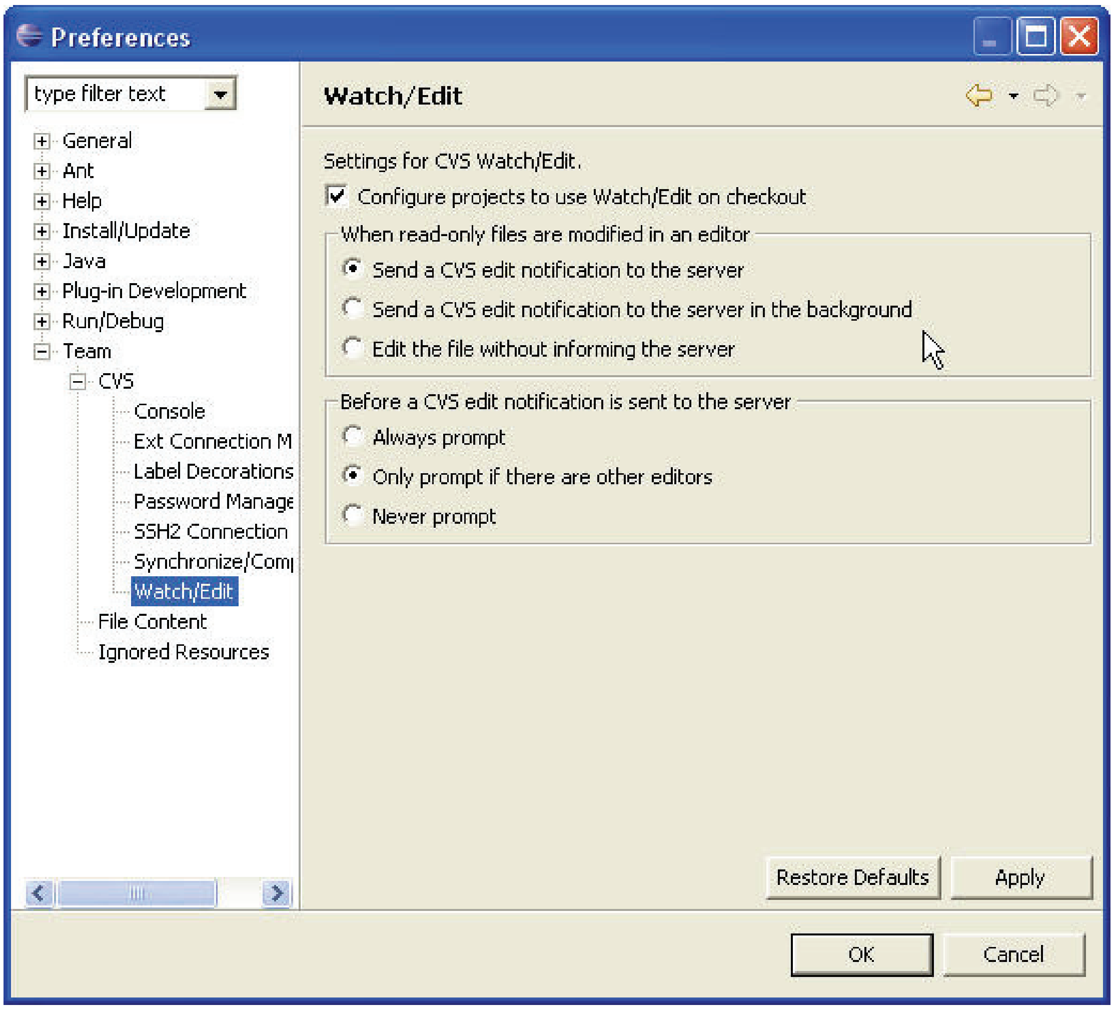
To checkout a project in Eclipse 2.1.3 requires using Window->Show View->Other, then selecting the CVS->CVS repositories view, then defining a New Repository Location, then opening the “HEAD”, then browsing to the project then right click to get the check out project option.

IBM Rational Application Developer for WebSphere Software (IBM WebSphere Studio Application Developer)

IBM Rational Application Developer for WebSphere Software (previously known as *IBM WebSphere Studio Application Developer* or simply *RAD*) is Eclipse and works identically to non-IBM releases as described in this document.

Eclipse Workflow

The Eclipse client is designed primarily to work with an Unreserved Distributed model of versioning. It is possible to configure it to work more co-operatively by enabling watch/edit in the preferences (Team->CVS->Watch Edit):



Watch/Edit

Settings for CVS Watch/Edit.

☑ Configure projects to use Watch/Edit on checkout

When read-only files are modified in an editor
- [✓ Send a CVS edit notification to the server]
- Send a CVS edit notification to the server in the background
- Edit the file without informing the server

Before a CVS edit notification is sent to the server
- Always prompt
- [✓ Only prompt if there are other editors]
- Never prompt

Eclipse Rename Workflow

Eclipse does not allow files to be renamed, the Refactor->rename function deletes the old file and creates new ones. After refactoring the Team system will prompt you to add the renamed files to version control, these files will be new and have no historical connection to your previous files.

An alternative is to refactor/rename the code in Eclipse then using the operating system to rename the physical file(s) to their original names. After that you can use another CVSNT client to rename the file within CVSNT.

Even then the results may not be what you expect. You wont be able to update the new filename to a tag that was placed on the previous filename, because that tag had the other filename. However you would be able to see (with cvs log) the history of the name change and the tags that each name has.

Versions of Eclipse prior to 3.1

If you are using a version of Eclipse earlier than 3.1 then you will need to configure the non-CVSNT client compatibility options in the CVSNT Server. Without these settings - versions of Eclipse prior to 3.1 may refuse to connect to a CVSNT server.

CVSNT servers since 2.0.35 have a configuration flag which when set forces the CVSNT server to impersonate a Unix CVS 1.11 server. This impersonation allows Eclipse to work with CVSNT (you must also use a repository prefix for this to work - Eclipse does not support drive letters).

See the section below *Common Problems with Eclipse* for detailed information on how to set the compatibility options for CVS Suite Server on Windows, Unix, Linux and Mac.

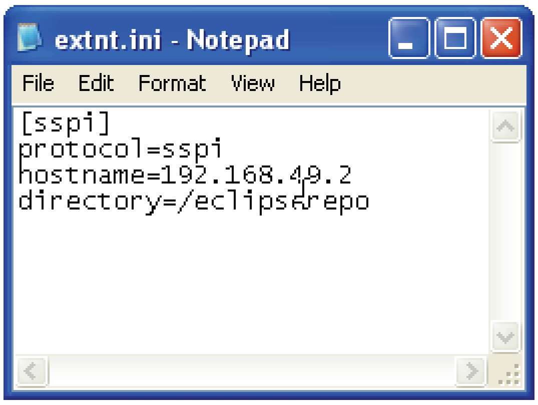
Connecting to Windows CVS Suite Server

To use Eclipse with a Windows based CVSNT server it is best to use the SSPI protocol. See the section *Authentication with SSPI* in the *Administration* section of the eBook *All About CM & CVS* for more information.

The built in Eclipse CVS client cannot connect to a Windows based CVSNT server using SSPI, so it is necessary to install the CVSNT command line client on the same workstation as Eclipse (CVSNT is available for HPUX, Solaris, Mac OS X, Linux and Windows). Eclipse then uses the CVSNT command line client via the EXT protocol.

Configure EXTNT

Use SSPI or SSERVER to connect using the EXT protocol. Open the configuration file extnt.ini which is located in the directory where you installed CVSNT.



```
[sspi]
protocol=sspi
hostname=192.168.49.2
directory=/eclipse/repo
```

Specify the following in the extnt.ini file:

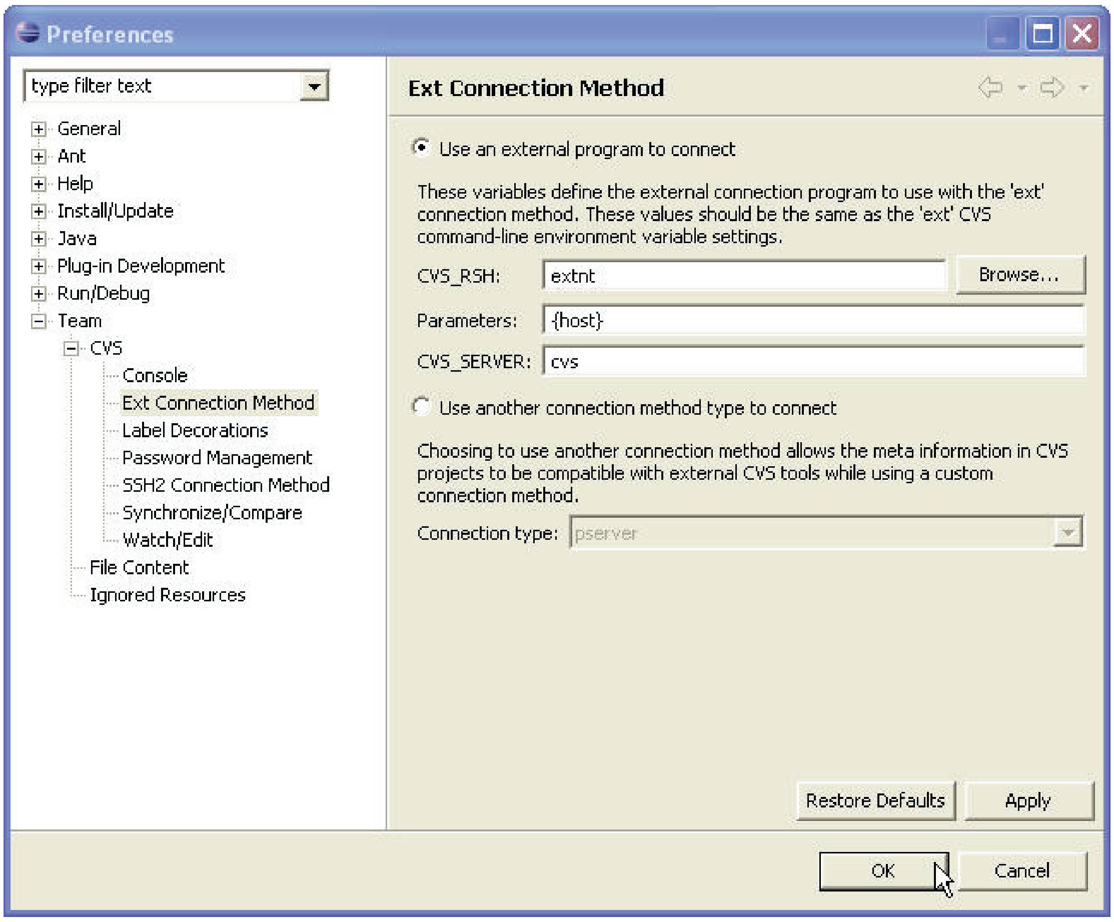
- `[server name]` for use from Eclipse (this does not have to be the actual server name where the repository is stored)
- `protocol=` This is the authentication technique to use with server (usually SSPI or SSERVER)
- `hostname=` This is the CVSNT server host, and is not the same as the *server name*.
- `directory=` This is the repository name on the server

Eclipse Preferences for Ext Connection Method

Eclipse stores the preferences for the EXT connection method with each workspace.

To set the preferences for the EXT connection, run Eclipse and from the pulldown menu choose Window->Preferences to open the preferences window.

A navigation tree at the left of the preferences window allows you to choose the page Team->CVS->Ext Connection Method:



(●) Use an external program to connect

These variables define the external connection program to use with the 'ext' connection method. These values should be the same as the 'ext' CVS command-line environment variable settings.

CVS_RSH: extnt

Parameters: {host}

CVS_SERVER: cvs

( ) Use another connection method type to connect

Choosing to use another connection method allows the meta information in CVS projects to be compatible with external CVS tools while using a custom connection method.

Connection type: pserver

If you are connecting using the protocol SSPI (defined in extnt.ini) then the parameters should be:

```
{host}
```

If you are connecting using the protocol SSERVER (defined in extnt.ini) and your user has a password then the parameters should be (note eclipse has a limitation where space characters cannot be used in a password):

```
{host} –l "{user}" –P {password}
```

If you are connecting using the protocol SSERVER (defined in extnt.ini) and your user does not have any password then the parameters should be:

```
{host} –l "{user}"
```

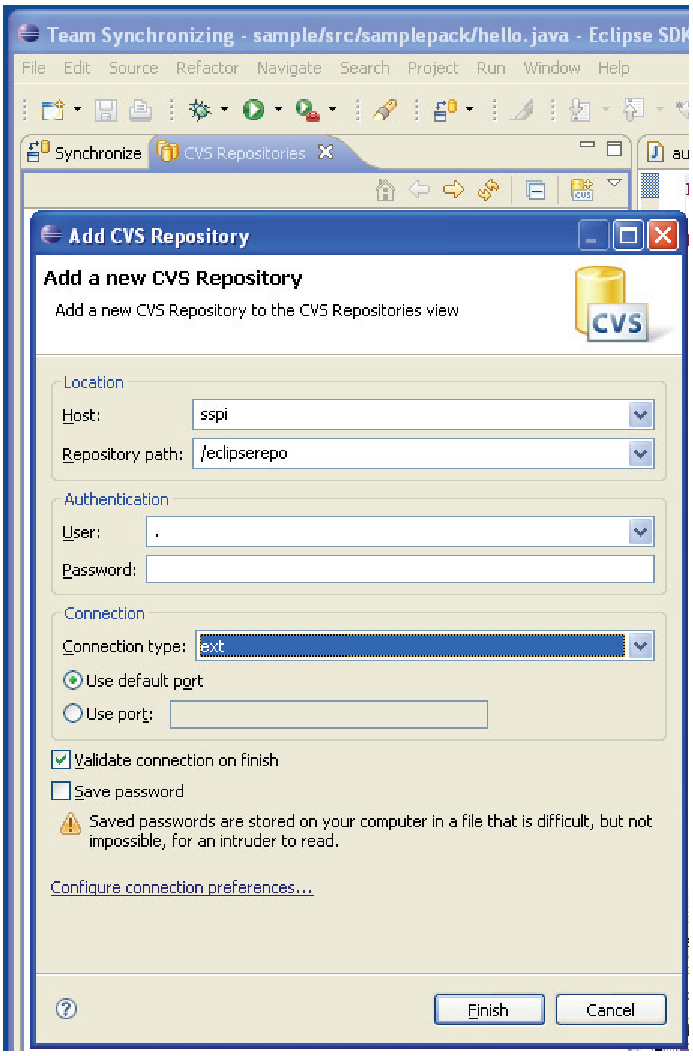
Eclipse Repository Location Information

When `EXTNT.INI`, and the Eclipse EXT Preferences are configured you can then specify the repository location using the standard CVS dialogs in Eclipse. Eg: in the CVS Repository Exploring Perspective (see below *Creating a new repository location*).

Creating a new repository location

A new repository location is defined in Eclipse when you have not checked in or out of this repository before. Specify the Host (using the server name from the EXTNT.INI file), The repository path (using the directory name from the EXTNT.INI file), the user name (if using SSPI then specify the username as a single period character, ie: ".") and password (leave blank for SSPI, or if there is no password).

Choose the Connection type of "ext", and then press Next.


Team Synchronizing - sample/src/samplepack/hello.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Synchronize
CVS Repositories
Add CVS Repository
CVS
Finish
Cancel

Add a new CVS Repository

Add a new CVS Repository to the CVS Repositories view

Location

Host: sspi

Repository path: /eclipserepo

Authentication

User: .

Password:

Connection

Connection type: ext

(●) Use default port

( ) Use port:

[✓] Validate connection on finish

[ ] Save password

Saved passwords are stored on your computer in a file that is difficult, but not impossible, for an intruder to read.

Configure connection preferences...

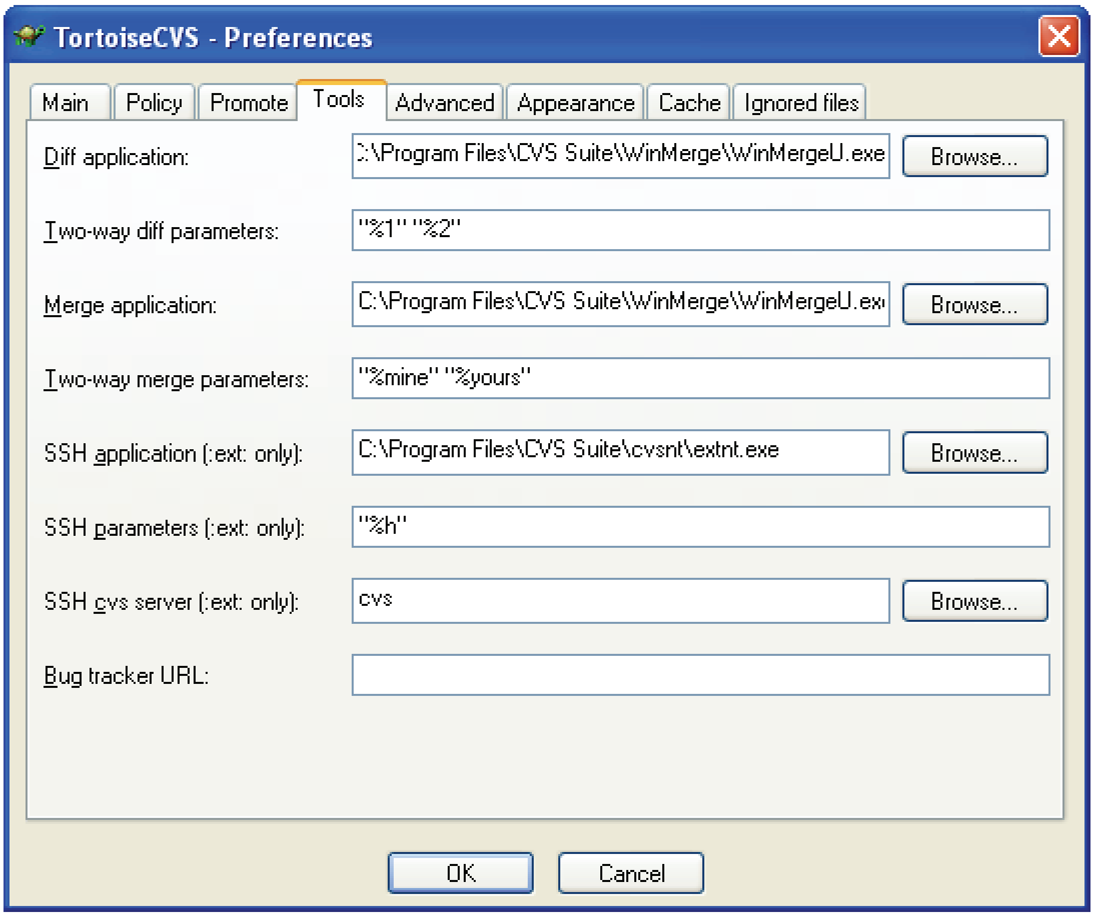
Using Tortoise with Eclipse Workspaces

To use Tortoise with an Eclipse Workspace you must do the following:

- Configure Tortoise to use the EXT method
- Use the "real" server name and repository name in both Eclipse and the extnt.ini configuration file.

Configure Tortoise to use the EXT method

Configure Tortoise to also use the EXT method. Open the Tortoise Preferences and set the "EXT" protocol program to extnt.exe:



Diff application: :\Program Files\CVS Suite\WinMerge\WinMergeU.exe
Two-way diff parameters: "%1" "%2"
Merge application: C:\Program Files\CVS Suite\WinMerge\WinMergeU.exe
Two-way merge parameters: "%mine" "%yours"
SSH application (:ext: only): C:\Program Files\CVS Suite\cvsnt\extnt.exe
SSH parameters (:ext: only): "%h"
SSH cvs server (:ext: only): cvs
Bug tracker URL:

Using the "real" names in extnt.ini configuration file.

The extnt.ini file maps the server name that you specify in Eclipse to a server, protocol and repository name that will be "really" used. Ideally these should be the same for both Tortoise and Eclipse, i.e.: my Eclipse CVSROOT is:

Host: sspi
Repository path: /eclipserepo
Connection type: ext
User: .
Password:

And the `extnt.ini` would match it:

```
[sspi]
protocol=sspi
hostname=servername
directory=/eclipserepo
```

Using CVSNT Command Line Client with Eclipse Workspaces

To use CVSNT command line client with an Eclipse Workspace you must do the following:

– Configure CVSNT to use the EXT method
– Use the "real" server name and repository name in both Eclipse and the extnt.ini configuration file.

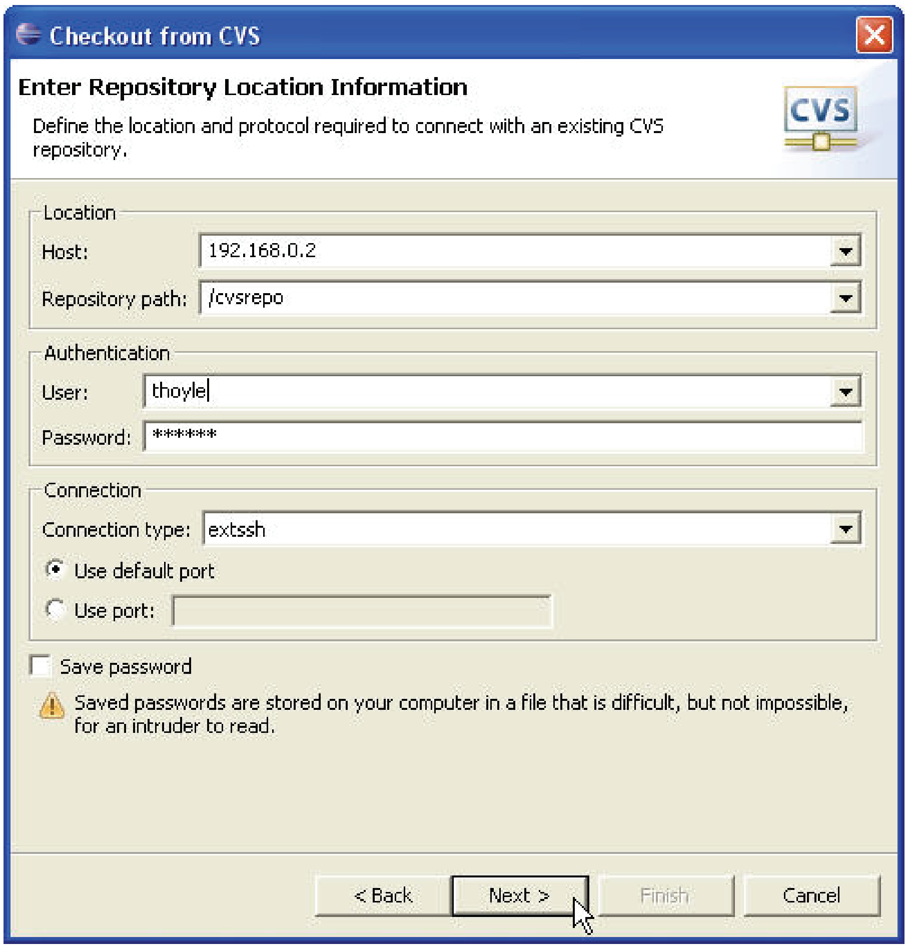
Configure CVSNT Command Line to use the EXT method

Configure CVSNT to use the EXT connection method by setting the environment variable `CVS_EXT`, similar to how you would in Tortoise or Eclipse:

```
CVS_EXT=extnt %h
```

Connecting to Unix, Linux or Mac OS X CVS Suite Server

If the CVSNT server is NOT ON WINDOWS then the Eclipse client will usually use EXTSSH to connect using the SSH protocol. This protocol is internal to Eclipse so therefore the settings in the Preferences do not affect it.



Checkout from CVS

Enter Repository Location Information

Define the location and protocol required to connect with an existing CVS repository.

Location
Host: 192.168.0.2
Repository path: /cvsrepo

Authentication
User: thoyle
Password: ******

Connection
Connection type: extssh
(●) Use default port
( ) Use port:

[ ] Save password

Saved passwords are stored on your computer in a file that is difficult, but not impossible, for an intruder to read.

Limitations to using Unix, Linux or Mac OS X CVSNT Server with EXTSSH

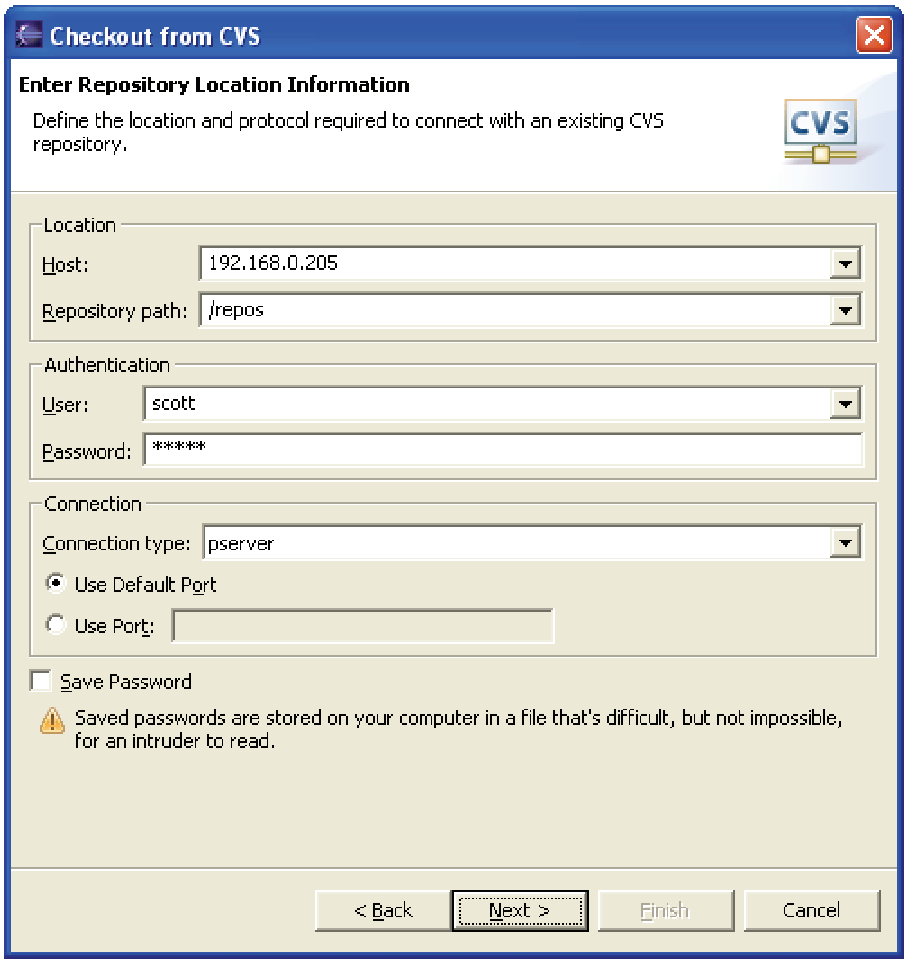
Use EXTSSH to connect using the SSH protocol to the server will only work unless the cvs command is on the default PATH on the server. The SSH connection bypasses the users .profile environment, so unless the cvs server command is on the default path then the EXTSSH connection will fail.

Usually CVS clients use the variable CVS_SERVER to specify the location of the server command, e.g. `CVS_SERVER=/apps/cvs/bin/cvsnt`

In Eclipse you can only set the CVS_SERVER variable in the preferences if you are using the EXT protocol (not the EXTSSH protocol). If you cannot add the cvs command to your servers path you must use EXT protocol instead with the preferences set for CVS+RSH = `TortoisePlink.exe`, Parameters = `-l "{user]" {host}` and CVS_SERVER = `/apps/cvs/bin/cvsnt`.

Connecting to CVS Suite Server using Insecure PServer Server

Use PServer to connect using the insecure PServer protocol (commonly used over the internet). This protocol is internal to Eclipse.



Enter Repository Location Information

Define the location and protocol required to connect with an existing CVS repository.

Location

Host: 192.168.0.205

Repository path: /repos

Authentication

User: scott

Password: *****

Connection

Connection type: pserver

(●) Use Default Port

( ) Use Port:

☐ Save Password

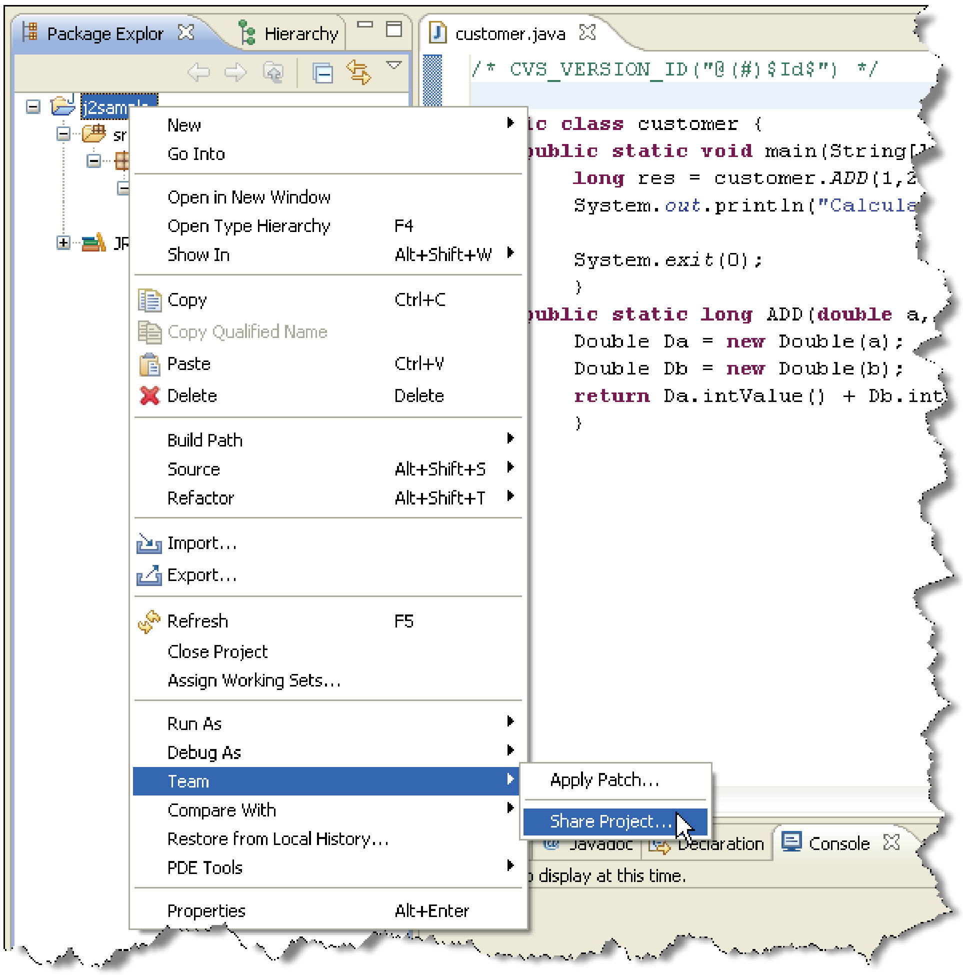
Saved passwords are stored on your computer in a file that's difficult, but not impossible, for an intruder to read.

Workflow – CVS Suite in an Eclipse Team

Using the Eclipse Integration

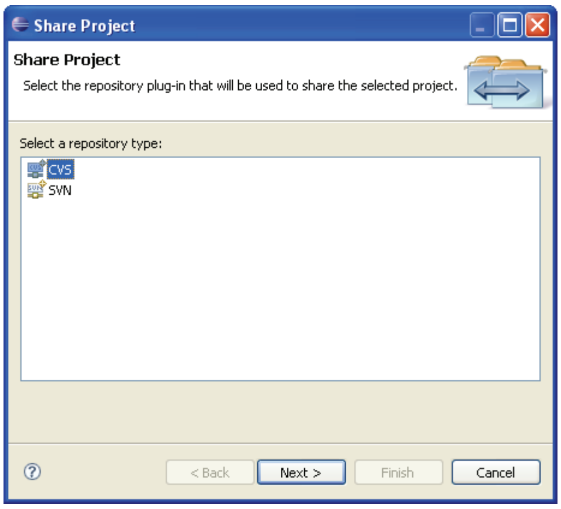
<u>Importing a new project into CVS Suite from Eclipse</u>

To import a new project into CVSNT, right click on the project and select Team->Share Project.



```
/* CVS_VERSION_ID("@(#)$Id$") */
ic class customer {
ublic static void main(String[
    long res = customer.ADD(1,2
    System.out.println("Calcula

    System.exit(0);
    }
ublic static long ADD(double a,
    Double Da = new Double(a);
    Double Db = new Double(b);
    return Da.intValue() + Db.int
    }
```

Choose CVS as the repository type and select the repository location you defined earlier:

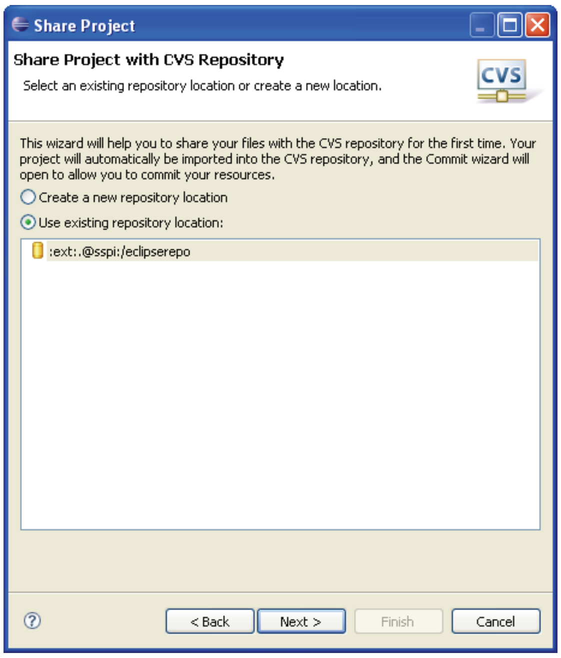


Share Project

Select the repository plug-in that will be used to share the selected project.

Select a repository type:

- CVS
- SVN



Share Project with CVS Repository

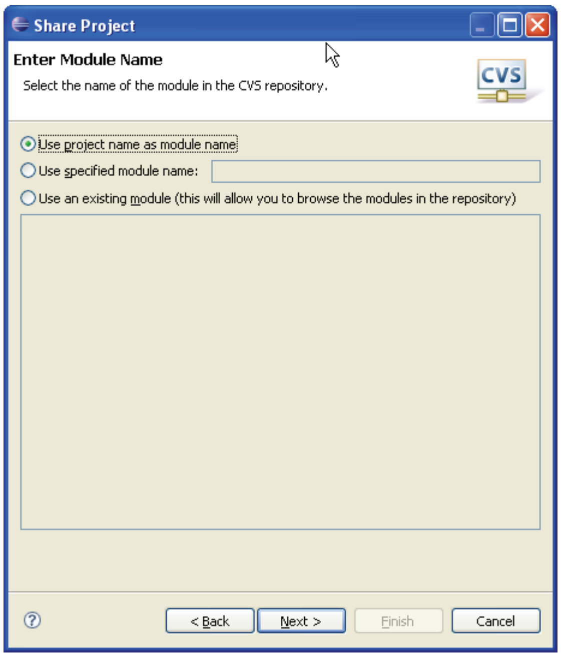
Select an existing repository location or create a new location.

This wizard will help you to share your files with the CVS repository for the first time. Your project will automatically be imported into the CVS repository, and the Commit wizard will open to allow you to commit your resources.

Create a new repository location [✓ Use existing repository location:]

:ext:.@sspi:/eclipserepo

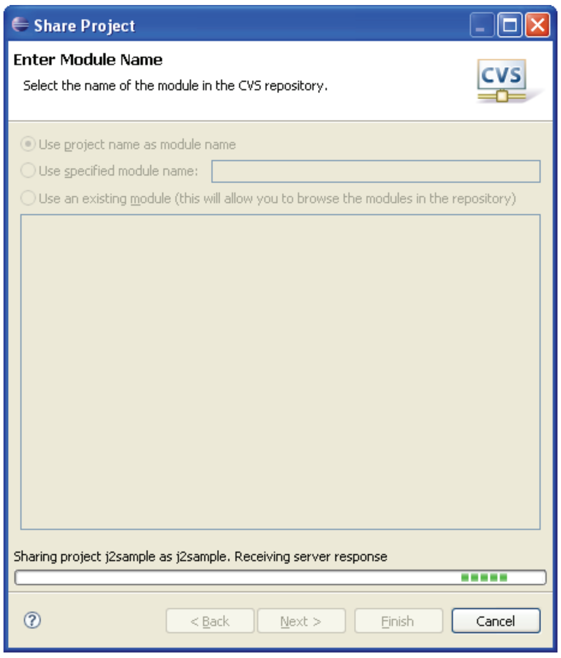
Finally to finish importing existing Eclipse projects to the CVS Suite Server select a module name (the default is usually sufficient):



Enter Module Name

Select the name of the module in the CVS repository.

[✓ Use project name as module name]
Use specified module name:
Use an existing module (this will allow you to browse the modules in the repository)



Enter Module Name

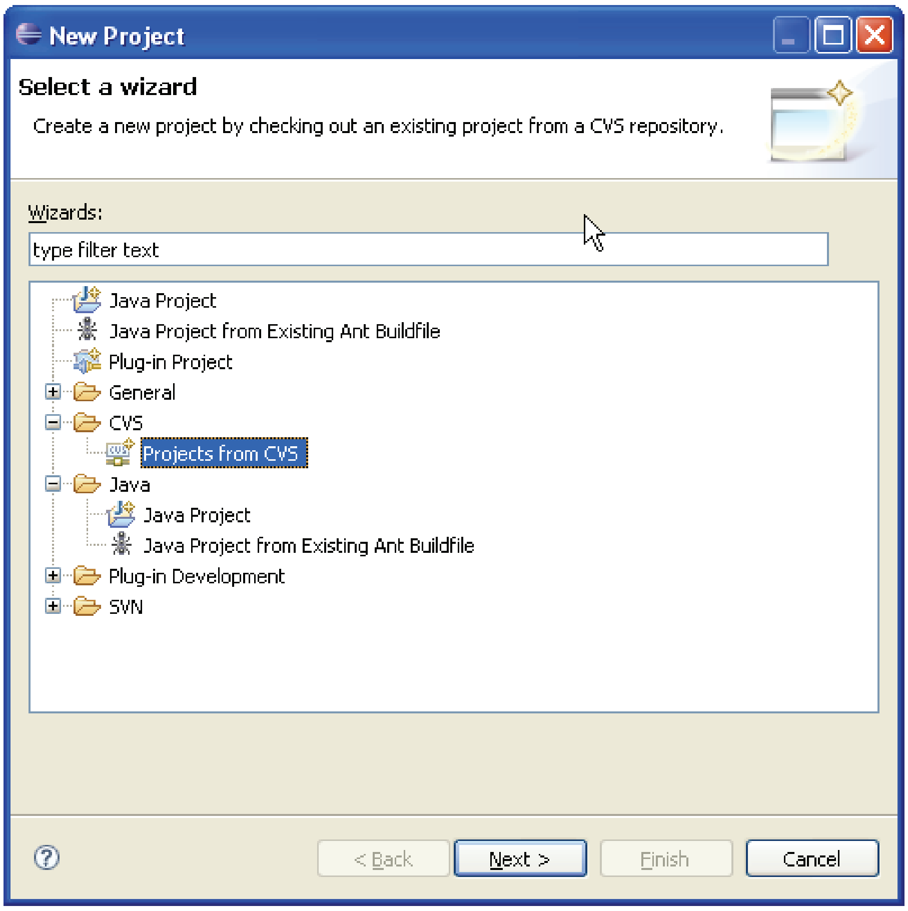
Select the name of the module in the CVS repository.

[✓ Use project name as module name]
Use specified module name:
Use an existing module (this will allow you to browse the modules in the repository)

Sharing project j2sample as j2sample. Receiving server response

Checkout a project from CVS Suite Server using Eclipse

If the project already exists in CVSNT, choose File->New->Project from the pulldown menu and use the CVS->Checkout Projects from CVS wizard.



New Project

Select a wizard

Create a new project by checking out an existing project from a CVS repository.

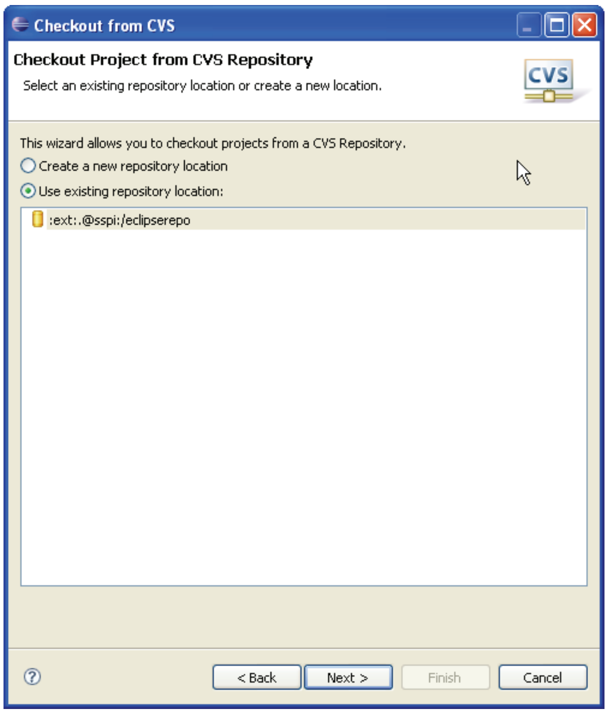
Wizards:

type filter text

- Java Project
- Java Project from Existing Ant Buildfile
- Plug-in Project
- General
- CVS
  - Projects from CVS
- Java
  - Java Project
  - Java Project from Existing Ant Buildfile
- Plug-in Development
- SVN

< Back | Next > | Finish | Cancel

On the following screens select ‘Use existing repository location’ and ‘Use an existing module’ to browse and choose from an existing module in the repository:



Checkout from CVS

Checkout Project from CVS Repository

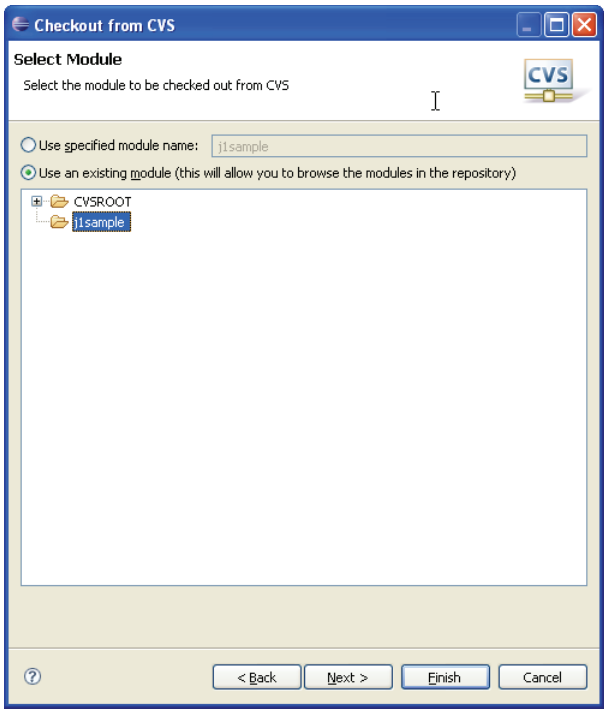
Select an existing repository location or create a new location.

This wizard allows you to checkout projects from a CVS Repository.

Create a new repository location [✓ Use existing repository location:]

:ext:.@sspi:/eclipserepo

< Back | Next > | Finish | Cancel



Checkout from CVS

Select Module

Select the module to be checked out from CVS

Use specified module name: j1sample

[✓ Use an existing module (this will allow you to browse the modules in the repository)]

- CVSROOT
- j1sample

< Back | Next > | Finish | Cancel

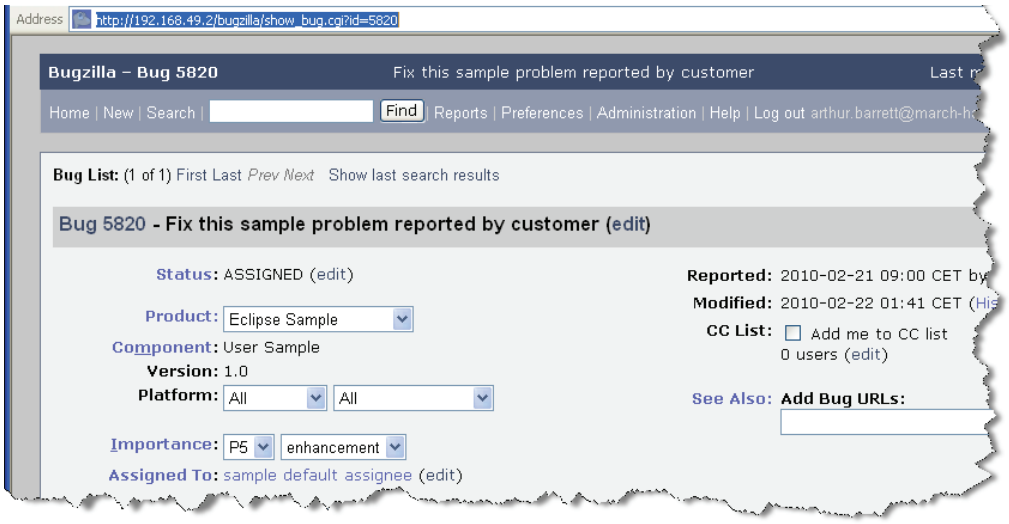
Bug fixing workflow using Eclipse

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

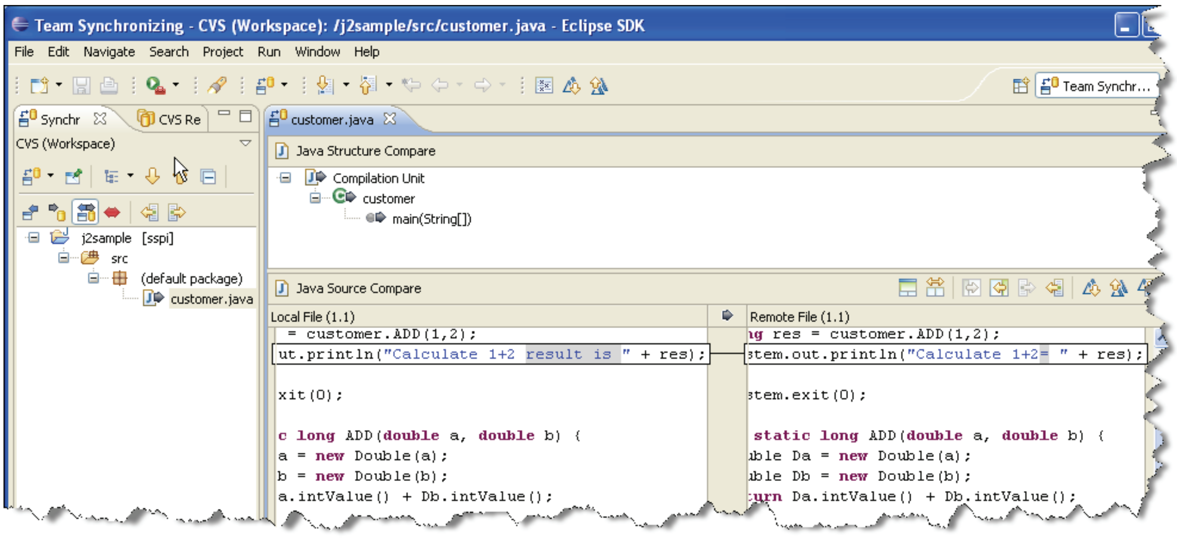
Bug List: (1 of 1) First Last Prev Next Show last search results

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

Status: ASSIGNED (edit)
Product: Eclipse Sample
Component: User Sample
Version: 1.0
Platform: All All
Importance: P5 enhancement
Assigned To: sample default assignee (edit)

Reported: 2010-02-21 09:00 CET by
Modified: 2010-02-22 01:41 CET (His
CC List: ☐ Add me to CC list
0 users (edit)
See Also: Add Bug URLs:

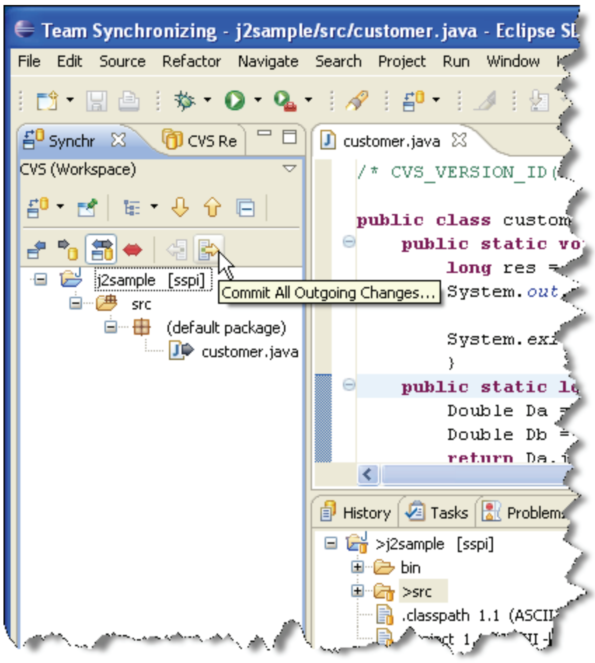
The developer can use the synchronize view to visually compare the changes in the eclipse workspace with the CVS Suite Server repository:



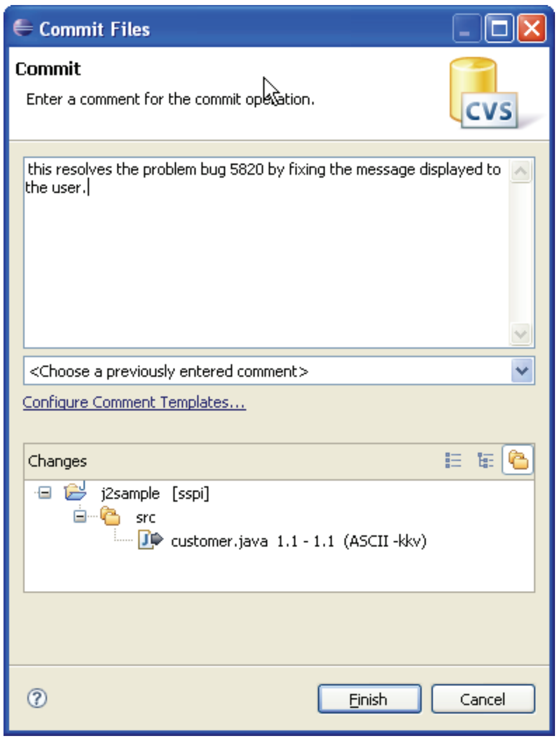
```
 = customer.ADD(1,2);
ut.println("Calculate 1+2 result is " + res);

xit(0);

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

```
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);
turn 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

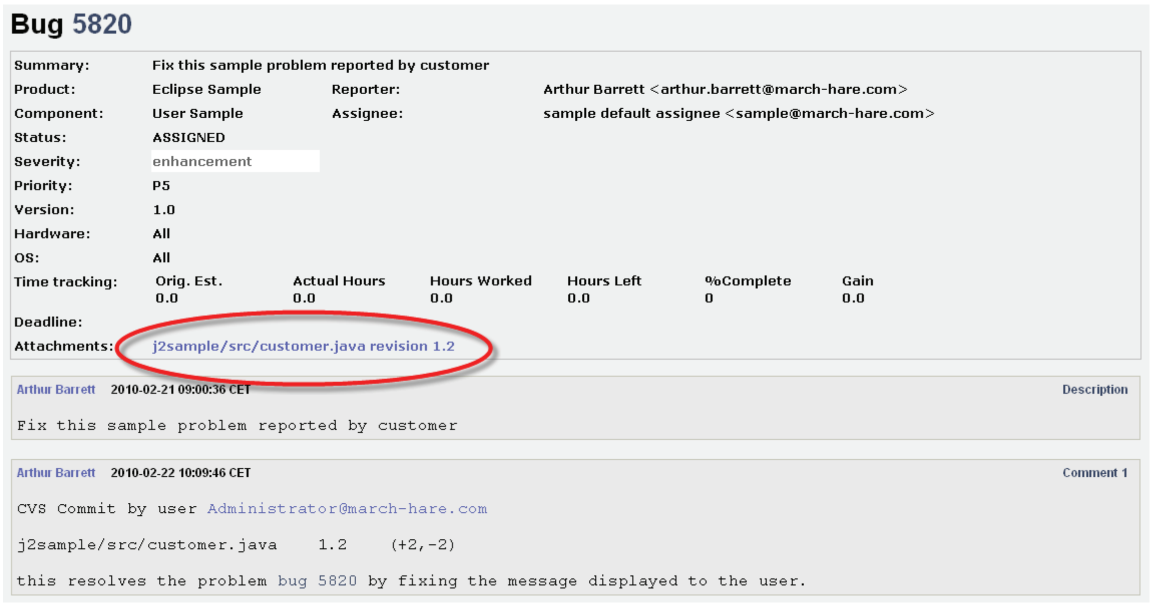
```
/* CVS_VERSION_ID (

public class custom
    public static vo
        long res =
        System.out
        
        System.exi
        }
    public static lo
        Double Da =
        Double Db =
        return Da.
```


Commit Files
Commit
Enter a comment for the commit operation.
CVS
<Choose a previously entered comment>
Configure Comment Templates...
Changes
j2sample [sspi]
src
customer.java 1.1 - 1.1 (ASCII -kkv)
Finish
Cancel

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

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		

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

Deadline:

Attachments: j2sample/src/customer.java revision 1.2

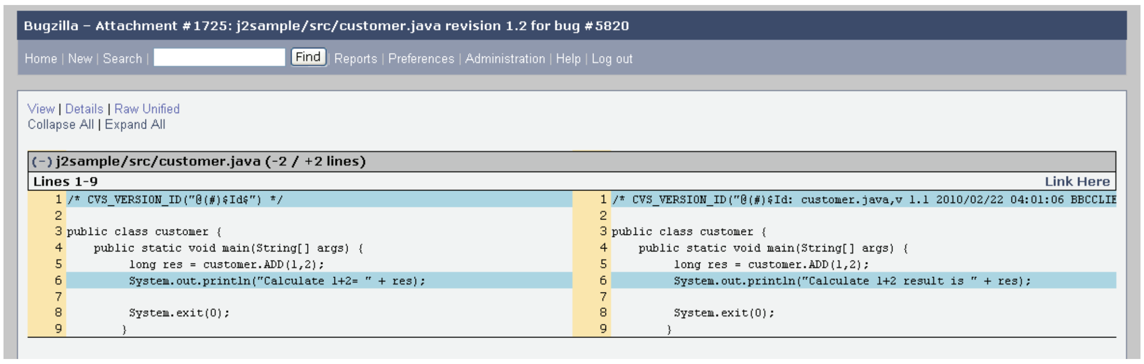
Arthur Barrett 2010-02-21 09:00:36 CET — Description

```
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

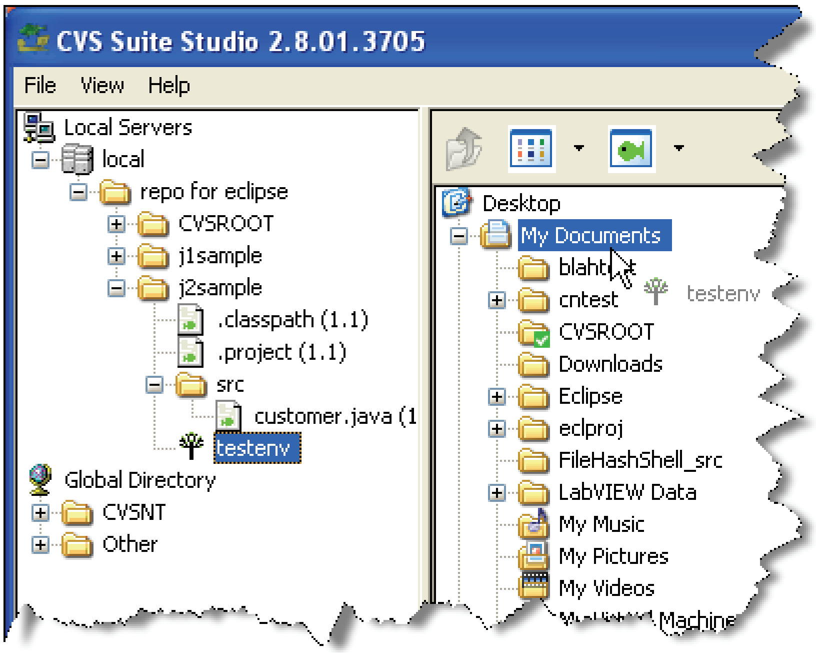
An SQL client (eg: Microsoft Excel) can query the audit database or the Bugzilla database to find all the files changed by a bug or all of the bugs affected by a file.

<u>Promote to test or production by bug number</u>

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:

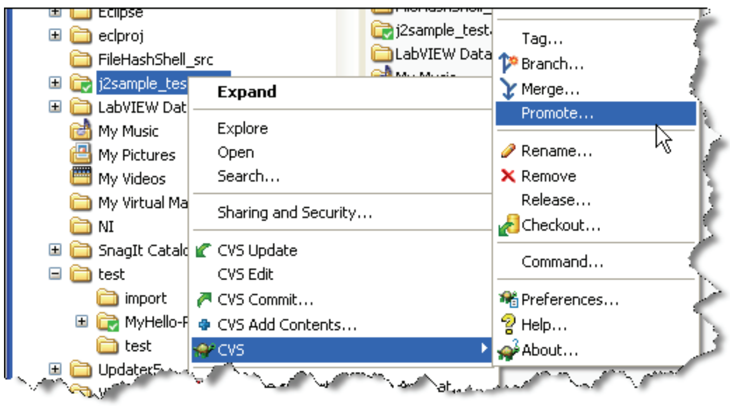
Firstly the QA manager creates a disk area to perform the promote:

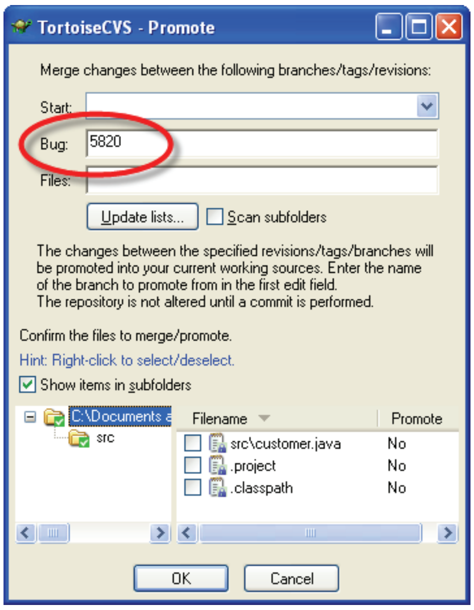
Drag the promotion level to a secure location:



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:

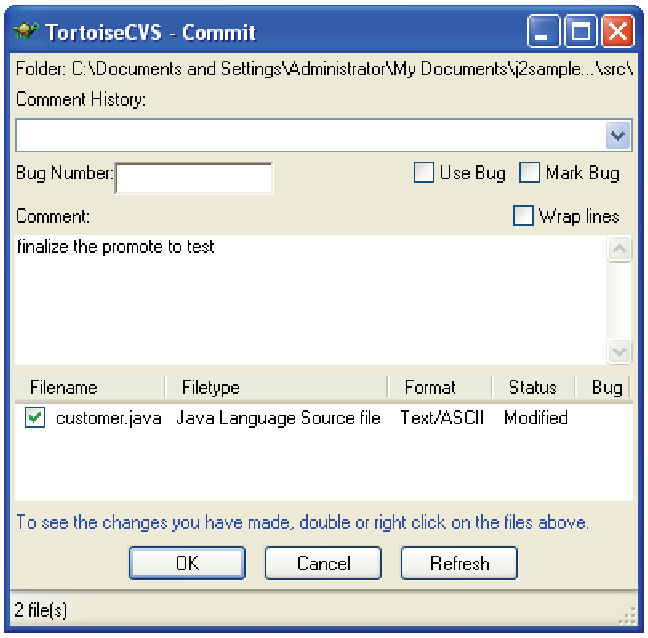
The changes between the specified revisions/tags/branches will be promoted into your current working sources. Enter the name of the branch to promote from in the first edit field.
The repository is not altered until a commit is performed.

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

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

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

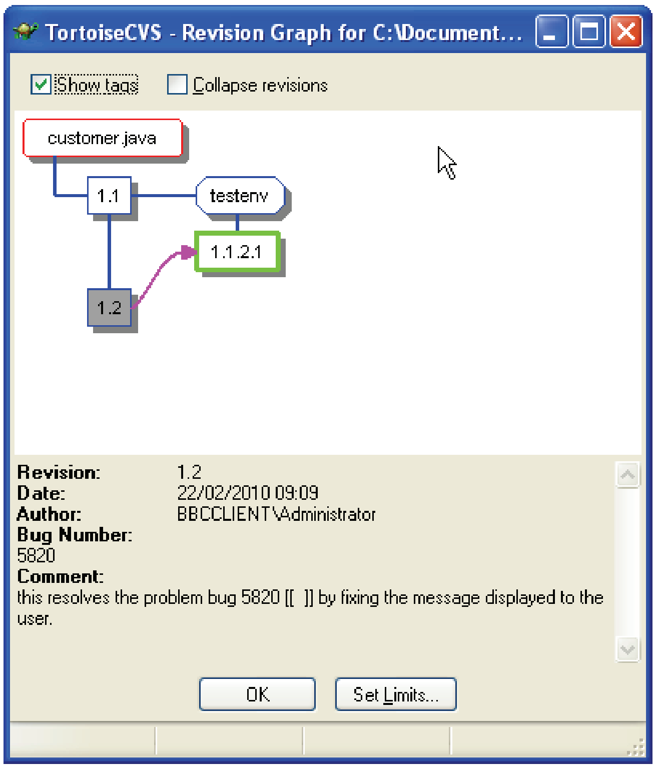
finalize the promote to test

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.

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.

Resolving Problems

Common problems with Eclipse

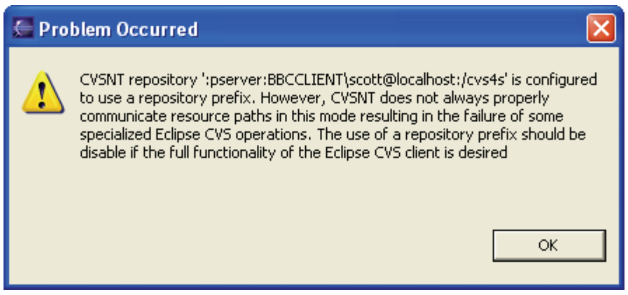
The following problems are frequently reported by people who use Eclipse

– Problem Occurred. CVSNT Repository is configured to use a repository prefix.
– Difficulties viewing tags and branches in Eclipse
– Obtain diagnostic and trace information from Eclipse

Problem Occurred – CVSNT Repository is configured to use a repository prefix

Eclipse may display the following error:

> *CVSNT Repository ':protocol:user@host:/repos' is configured to use a repository prefix. However, CVSNT does not always properly communicate resource paths in this mode resulting in the failure of some specialized Eclipse CVS operations. The use of a repository prefix should be disable if the full functionality of the Eclipse CVS client is desired*

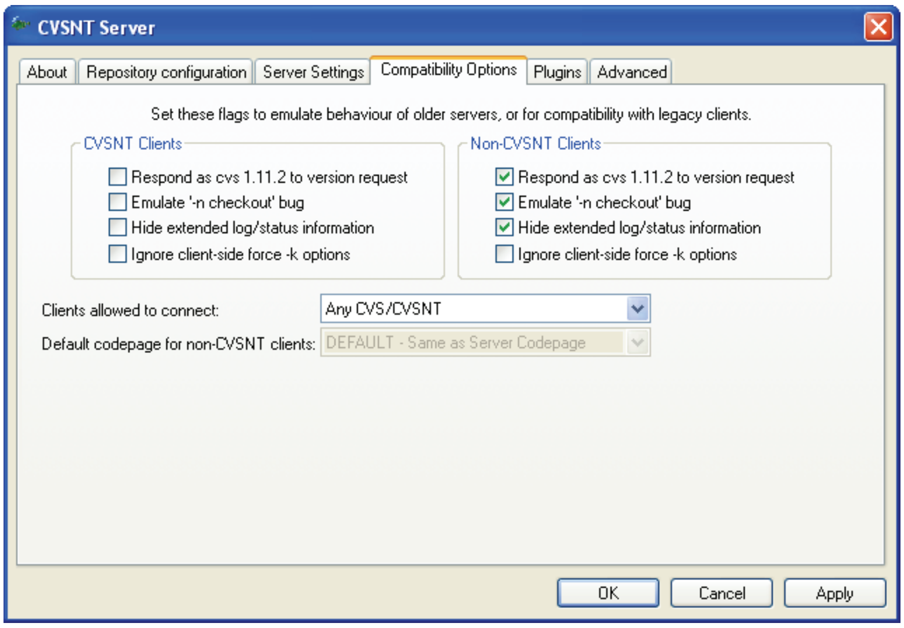

Problem Occurred

CVSNT repository ':pserver:BBCCLIENT\scott@localhost:/cvs4s' is configured to use a repository prefix. However, CVSNT does not always properly communicate resource paths in this mode resulting in the failure of some specialized Eclipse CVS operations. The use of a repository prefix should be disable if the full functionality of the Eclipse CVS client is desired

This message is caused by an internal error in Eclipse, and may be disabled by setting the correct compatibility options on the CVS Suite Server Control Panel Conpatibility Options tab on windows or in the `/etc/cvsnt/PServer` configuration file on unix/linux/mac.

Disabling Eclipse error messages on CVS Suite Server for Windows

To disable the Eclipse error messages using CVS Suite Server for Windows, open the CVSNT Server control panel and open the Compatibility Options tab and enable all the compatibility options for non-CVSNT clients:



Set these flags to emulate behaviour of older servers, or for compatibility with legacy clients.

CVSNT Clients
- [ ] Respond as cvs 1.11.2 to version request
- [ ] Emulate '-n checkout' bug
- [ ] Hide extended log/status information
- [ ] Ignore client-side force -k options

Non-CVSNT Clients
- [x] Respond as cvs 1.11.2 to version request
- [x] Emulate '-n checkout' bug
- [x] Hide extended log/status information
- [ ] Ignore client-side force -k options

Clients allowed to connect: Any CVS/CVSNT

Default codepage for non-CVSNT clients: DEFAULT - Same as Server Codepage

Disabling Eclipse error messages on CVS Suite Server for Linux, Unix and Mac

To disable the Eclipse error messages using CVS Suite Server for Windows, open the CVSNT configuration file `/etc/cvsnt/PServer` amd enable all the compatibility options for non-CVSNT clients:

```
Compat0_OldVersion=1
Compat0_HideStatus=1
Compat0_OldCheckout=1
```

Common problems with Tags and Branches

Many people report problems using Tags and Branches with CVS and Eclipse. The most common causes of this are:

– CVS Server is using *val-tags*
– Repository Paths do not match exactly
– Eclipse `.project` file is versioned and incorrectly tagged
– Eclipse cache is not up to date

Eclipse cannot see tags/branches – server is using val-tags

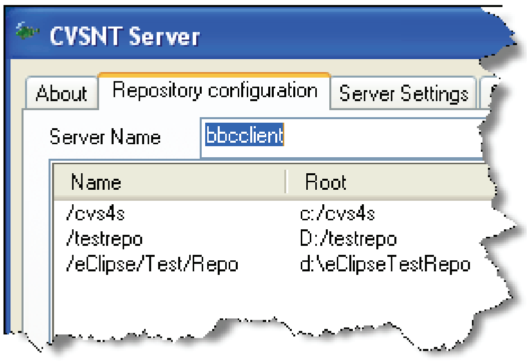
We recommend that the val-tags file not be used, and it is disabled by default with CVS Suite Server 2009. In CVS Suite Server 2009 the use of val-tags can be disabled on windows using the CVS Suite Server Control Panel Advanced tab or in the `/etc/cvsnt/PServer` configuration file on unix/linux/mac. On older servers (before CVS Suite 2008) it cannot be disabled.

If the *val-tags* file exists on the server then all users of CVS and Eclipse should have read and write access to it at the operating system level.

Eclipse cannot see tags/branches – Repository Paths do not match exactly

Since Eclipse 3.0 M9, repository paths must match exactly the case of the repository paths set in the CVSNT control panel. If it is not done this way Eclipse will see the repository and the files, but will not see the branches or versions.

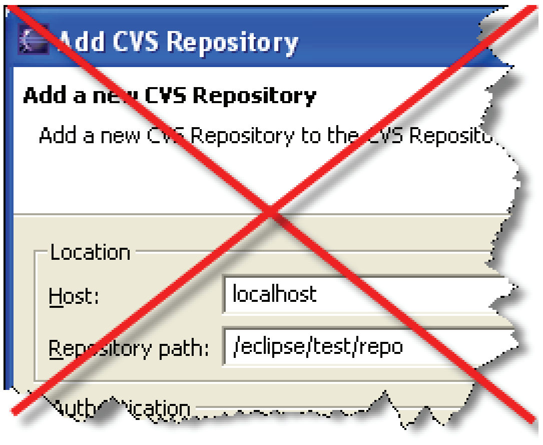
For example, this repository has a mixed case repository prefix:

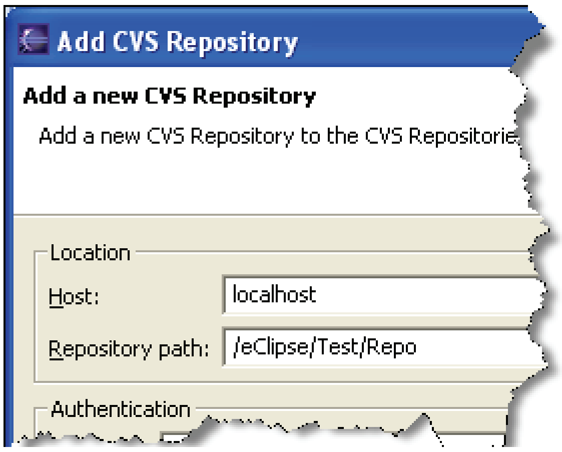


Server Name bbcclient

Name	Root
/cvs4s	c:/cvs4s
/testrepo	D:/testrepo
/eClipse/Test/Repo	d:\eClipseTestRepo

The incorrect and correct definitions in Eclipse are:

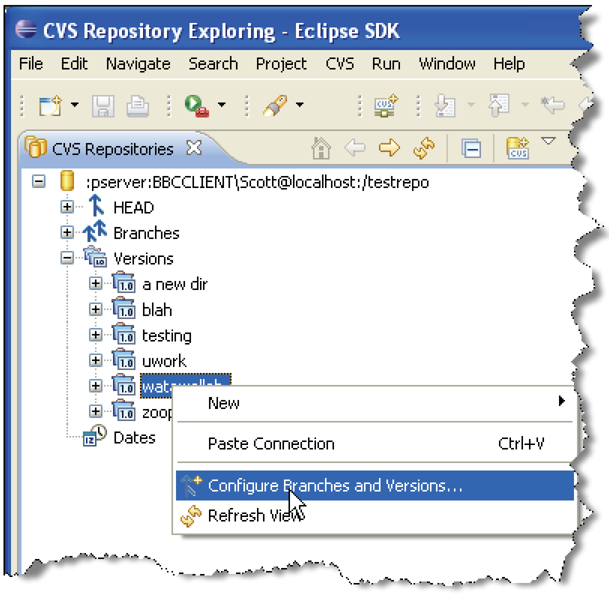

Add CVS Repository
Add a new CVS Repository
Add a new CVS Repository to the CVS Reposito
Location
Host: localhost
Repository path: /eclipse/test/repo
Authentication

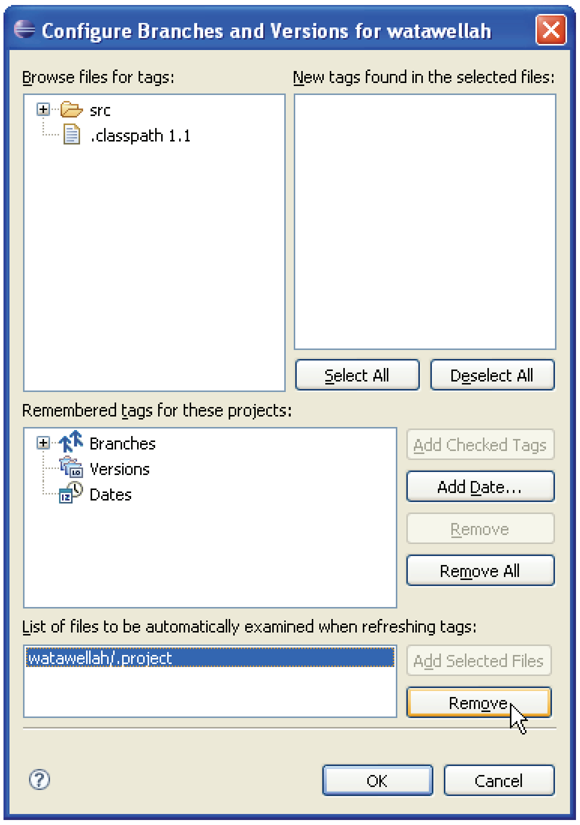

Add CVS Repository
Add a new CVS Repository
Add a new CVS Repository to the CVS Repositorie
Location
Host: localhost
Repository path: /eClipse/Test/Repo
Authentication

Eclipse .project file is versioned and incorrectly tagged

Eclipse will always look for tag in the .project file stored in the root of the project first – only if it fails to find tags on this file will it then search other files. It is good practice to ensure that if the `.project` file is added to the repository that you ensure it is always tagged with all tags, or it is never tagged.

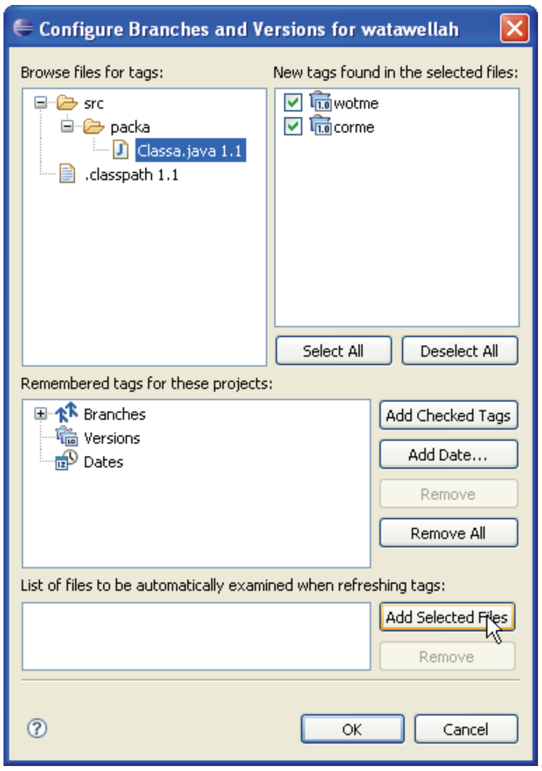
If the `.project` file has already been added to the repository it is possible to configure the Eclipse workspace to search other files in the project for tags. Open the CVS Repositories sidebar using the pulldown menu Window->Perspective->CVS Repository Exploring, then right click the project and select 'Configure branches and versions...':


CVS Repository Exploring - Eclipse SDK
File Edit Navigate Search Project CVS Run Window Help
CVS Repositories
:pserver:BBCCLIENT\Scott@localhost:/testrepo
HEAD
Branches
Versions
a new dir
blah
testing
uwork
watawellah
zoop
Dates
New
Paste Connection Ctrl+V
Configure Branches and Versions...
Refresh View


Configure Branches and Versions for watawellah
Browse files for tags:
src
.classpath 1.1
New tags found in the selected files:
Select All
Deselect All
Remembered tags for these projects:
Branches
Versions
Dates
Add Checked Tags
Add Date...
Remove
Remove All
List of files to be automatically examined when refreshing tags:
watawellah/.project
Add Selected Files
Remove
OK
Cancel

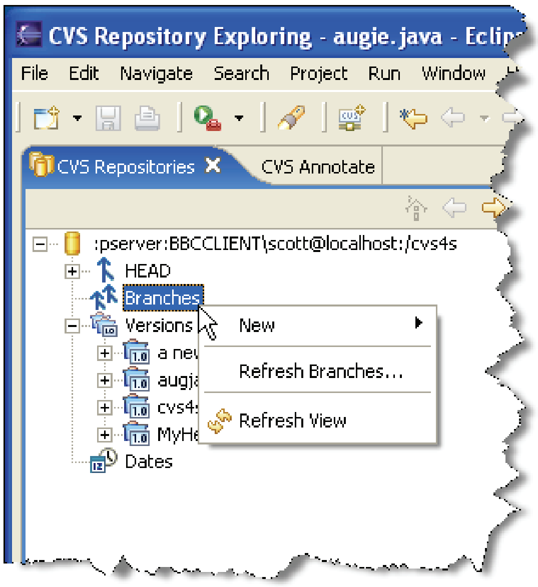
On the 'Configure branches and versions for *project*' dialog, at the very bottom of the screen is the file `project/.project` - select this file and press 'remove'

On the 'configure branches and versions for *project*' dialog, at the top of the screen is 'browse files for tags' - browse to a file which is known to have all the tags *then* press 'Add Selected Files' button (bottom right of dialog) then press OK:

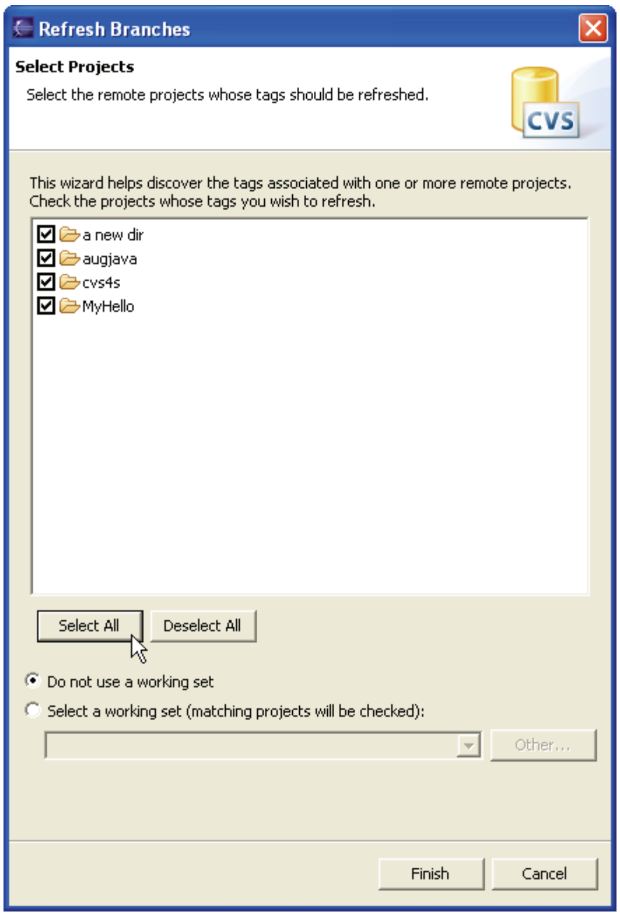


Eclipse cannot see tags/branches – eclipse cache is not up to date

Using Open Perspective from the Window menu open the CVS Repository Exploring perspective then right click branches and choose the option Refresh Branches…:


CVS Repository Exploring - augie.java - Eclip
File Edit Navigate Search Project Run Window
CVS Repositories
CVS Annotate
:pserver:BBCCLIENT\scott@localhost:/cvs4s
HEAD
Branches
Versions
a new
augi
cvs4
MyHe
Dates
New
Refresh Branches...
Refresh View

Choose the option *Select All* plus the option *Do not use a working set* and press OK to refresh the Eclipse tag and branch cache:



Select Projects

Select the remote projects whose tags should be refreshed.

This wizard helps discover the tags associated with one or more remote projects. Check the projects whose tags you wish to refresh.

- ☑ a new dir
- ☑ augjava
- ☑ cvs4s
- ☑ MyHello

[✓ Do not use a working set]
Select a working set (matching projects will be checked):

Diagnostics – What to send to Support

Obtain diagnostic and trace information from Eclipse

If you have a problem with the interoperability try to get a client/server trace from eclipse before reporting bugs as it is difficult for the developers to diagnose problems without this detailed information.

There are tracing facilities in Eclipse that will allow you to see what messages are being communicated between the CVS client and server. This section describes how.

Pre-requisites

You will need Eclipse and a Java Run-time installed on the client. On Windows Eclipse does not usually require a separate Java run time, but to enable debugging you will need this as well.

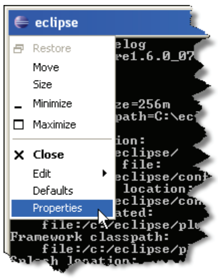
Step 1

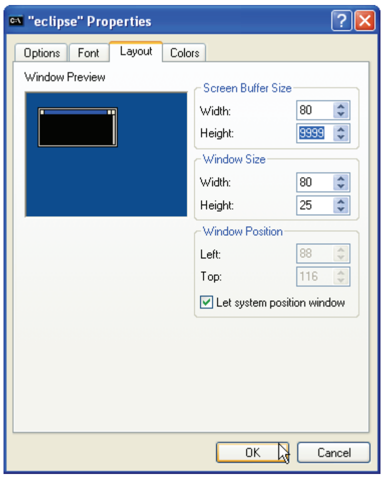
Create a file named `.options` in the directory you start Eclipse from (in most cases this is the directory that contains the executable but it may differ in some cases: for instance, if you use a shortcut in windows and specify a different starting directory) that contains the following 2 lines that enable CVS debugging.

```
org.eclipse.team.cvs.core/debug=true
org.eclipse.team.cvs.core/cvsprotocol=true
```

Step 2

Start Eclipse with the following parameters tailored to you local setup by creating a windows shortcut to the eclipse executable. Once the black & white debug window is visible you can also use the 'properties' menu of the debug window to set the Layout Scrren Buffer Size Height and the Edit Option Quick Edit Mode, eg:




"eclipse" Properties

Options | Font | Layout | Colors

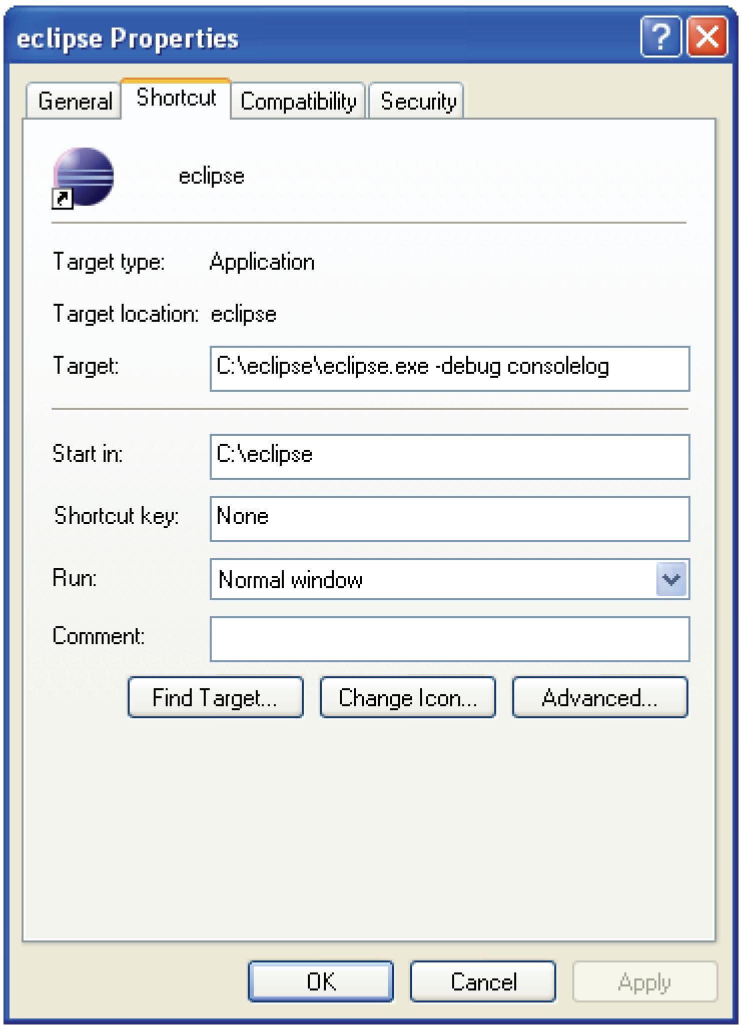
Window Preview

Screen Buffer Size
Width: 80
Height: 9999

Window Size
Width: 80
Height: 25

Window Position
Left: 88
Top: 116
☑ Let system position window

OK | Cancel



eclipse Properties

General | Shortcut | Compatibility | Security

eclipse

Target type: Application

Target location: eclipse

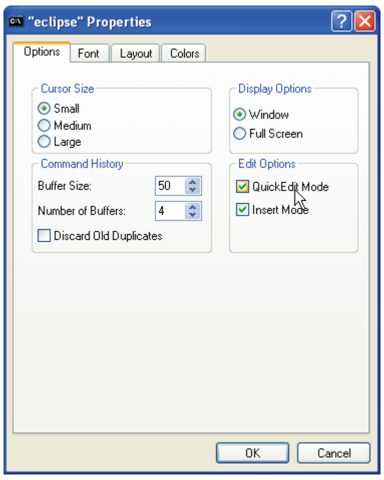
Target: C:\eclipse\eclipse.exe -debug consolelog

Start in: C:\eclipse

Shortcut key: None

Run: Normal window

Comment:



"eclipse" Properties

Options | Font | Layout | Colors

Cursor Size: [✓ Small] Medium Large

Display Options: [✓ Window] Full Screen

Command History

Buffer Size: 50

Number of Buffers: 4

☐ Discard Old Duplicates

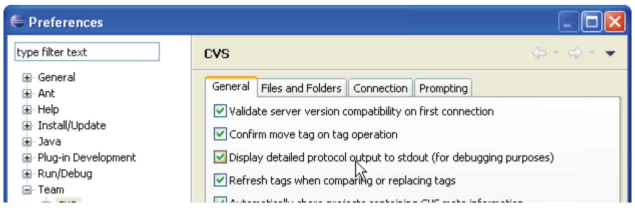
Edit Options

✓ QuickEdit Mode

✓ Insert Mode

Step 3

Inside Eclipse, in the preferences set the checkbox:
Team->CVS->Display detailed protocol output to stdout.



✓ Validate server version compatibility on first connection

✓ Confirm move tag on tag operation

✓ Display detailed protocol output to stdout (for debugging purposes)

✓ Refresh tags when comparing or replacing tags

Step 4

Inside Eclipse, create your repo location and expand it in the repositories view (for example). The CVS command traffic in the debug console should contain an invocation of the update command that looks something like (this is output from dev.eclipse.org):

```
CMD> cvs -n update -d "."
...
update
E cvs server: Updating .
E cvs server: New directory `CVSROOT' -- ignored
E cvs server: New directory `jdt-core-home' -- ignored
E cvs server: New directory `jdt-debug-home' -- ignored
...
```

Create a text file containing all of the output from the debug console by using cut and paste from the debug console window to a notepad window.

Send Debugging information to March Hare Software

Find the ".metadata" directory in your workspace directory and zip it along with the output of the debug console to your March Hare Software Technical Account manager for further assistance.

The .metadata directory contains your workspace preferences, and in particular these files are vital to the support team reproducing your environment:

```
.metadata\.plugins\org.eclipse.core.runtime\.settings

   org.eclipse.team.core.prefs
   org.eclipse.team.cvs.core.prefs
   org.eclipse.team.cvs.ui.prefs
   org.eclipse.team.ui.prefs
```

Appendix

build_make.bat (windows) - incomplete

```
@echo off
@echo Build: User=%USER% {%Username%},1=%1,2=%2,3=%3,4=%4,5=%5,6=%6,7=%7
IF %1~==projectb~ GOTO runjava_projectb
IF %1~==projecta~ GOTO runjava_projecta
@echo "This sample build only makes projecta and projectb"
exit /b 9
:runjava_projectb
IF %3~==testenv~ GOTO manageant
IF %3~==prodenv~ GOTO manageant
@echo "This sample build only makes projectb on promotion levels testenv or prodenv"
exit /b 8
:runjava_projecta
IF %3~==testenv~ GOTO manageant
IF %3~==prodenv~ GOTO manageant
@echo "This sample build only makes projecta on promotion levels testenv or prodenv"
exit /b 8

:manageant
d:
@if exist d:\antbld goto antnavok
mkdir d:\antbld
chdir d:\antbld
@if exist d:\antbld goto antnavok
@echo "Could not create directory for Eclipse ANT projects"
exit /b 7

:antnavok
@if exist d:\antbld\%3 goto antlvlok
mkdir d:\antbld\%3
chdir d:\antbld\%3
@if exist d:\antbld\%3 goto antlvlok
@echo "Could not create directory for promotion level %3 "
exit /b 7

:antlblok
@if exist d:\antbld\%3\%1 goto antprojupd
cvs -f -d :local:%REAL_CVSROOT% co -r %3 %1
@if NOT ERRORLEVEL 0 goto antprojcoerr
@if exist d:\antbld\%3\%1\CVS goto antprojok
:antprojcoerr
@echo "Could not checkout %1 promotion level %3 "
exit /b 6

:antprojupd
@if not exist d:\sqlnav\%3\%1\CVS goto antprojerr
@if not exist d:\sqlnav\%3\%1\CVS\Root goto antprojerr
@if not exist d:\sqlnav\%3\%1\CVS\Entries goto antprojerr
cd d:\antbld\%3\%1
@echo Perform update in d:\antbld\%3\%1
@echo cvs -f -d :local:%REAL_CVSROOT% up -A -C -d -r %3 %1
cvs -f up -A -C -d -r %3
@if ERRORLEVEL 0 goto antprojok
:antprojerr
@echo "Could not update %1 promotion level %3 "
exit /b 5

:antprojok
cd d:\antbld\%3\%1
@if exist d:\antbld\%3\%1\build.xml goto antmakeok
@echo "<?xml version="1.0"?>" > build.xml
@echo "<!—sample ant build file -->" >> build.xml
@echo+ >> build.xml
@if exist d:\antbld\%3\%1\build.xml goto antmakeok
@echo "Could not find or create a build.xml for %1 promotion level %3"
exit /b 4


:antmakeok
ant -f build.xml
exit /b %errorlevel%
```

build_make.sh (unix) - incomplete

```
build_make.bat
@echo off
@echo Build: User=%USER% {%Username%},1=%1,2=%2,3=%3,4=%4,5=%5,6=%6,7=%7
```

build.xml (windows generated) -- incomplete

```
<?xml version="1.0"?>
<!—sample ant build file -->
```

makefile.mak (linux) -- incomplete

```
<?xml version="1.0"?>
<!—sample ant build file -->
```