



Oracle and CVS Suite

WHITEPAPER 2009.4087 March 2011

Legal Notices

There are various product or company names used herein that are the trademarks, service marks, or trade names of their respective owners, and March Hare Software Limited makes no claim of ownership to, nor intends to imply an endorsement of, such products or companies by their usage.

This document and all information contained herein are the property of March Hare Software Limited, and may not be reproduced, disclosed, revealed, or used in any way without prior written consent of March Hare Software Limited.

This document and the information contained herein are subject to confidentiality agreement, violation of which will subject the violator to all remedies and penalties provided by the law.

LIMITED WARRANTY.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, March Hare Software Limited AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES AND CONDITIONS, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, WITH REGARD TO THIS DOCUMENT, AND ANY ADVICE OR RECOMMENDATION CONTAINED IN THIS DOCUMENT.

NO OTHER WARRANTIES.

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL March Hare Software Limited OR ITS SUPPLIERS BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE FOLLOWING DOCUMENTATION INCLUDING ANY RECOMMENDATION OR ADVICE THEREIN, EVEN IF March Hare Software Limited HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, March Hare Software Limited's ENTIRE LIABILITY UNDER ANY PROVISION OF THIS DOCUMENT INCLUDING ANY RECOMMENDATION OR ADVICE THEREIN SHALL BE LIMITED TO THE GREATER OF THE AMOUNT ACTUALLY PAID BY YOU FOR THE DOCUMENT OR £5.00; PROVIDED.

© March Hare Software Limited, 2009

march-hare.com

consultants@march-hare.com

Table of Contents

LEGAL NOTICES	A
TABLE OF CONTENTS	1
OVERVIEW	3
Development integration with Quest SQL Navigator 6.4.....	3
Support for a Dev -> Test -> Production development lifecycle.....	3
Integration with Change Management using a Defect Tracking System, e.g.: Bugzilla	4
INSTALLATION	4
<i>Installed Components</i>	4
<i>Customised installations</i>	4
CONTACTING SALES	5
United Kingdom.....	5
USA & Canada.....	5
FUNDAMENTALS	6
WHAT IS CVSNT AND CM SERVER ALL ABOUT	6
CAN I INSTALL THE SOFTWARE AND READ THIS LATER	6
I DON'T LIKE VERSION CONTROL	6
I PREFER TO USE SOME OTHER TOOL	6
THEORY	7
PROMOTION MODEL	7
Example 1.....	7
Example 2.....	7
WHAT ARE BRANCHES, MAGIC BRANCHES AND VENDOR BRANCHES	8
<i>When are Branches, Magic Branches and Vendor Branches Used</i>	8
Example 1.....	8
Example 2.....	8
Example 3.....	8
Example 4.....	8
Example 5.....	9
<i>What are the benefits to using Branches, Magic Branches and Vendor Branches</i>	9
Making the same changes twice	9
Ensure security.....	9
<i>What is different between a Branch and a Magic Branch</i>	9
MIXED MODEL – BRANCHING AND PROMOTING	10
<i>Mixing up the development models</i>	10
PATCH MANAGEMENT – GETTING FIXES TO CUSTOMERS	10
<i>Service Packs</i>	10
<i>Patches</i>	10
WHAT IS THE REPOSITORY AND THE WORKSPACE	12
<i>Repository Version History</i>	12
<i>Working Copy or Workspace (some people call this the sandbox)</i>	12
<i>Working Copy Folder versus a Working Copy Database</i>	13
<i>Moving Objects from One database to another Database (eg: Dev to Test)</i>	13
CVS SUITE AND ORACLE WORKFLOW	14
WORKFLOW DEFINITION	14
CVS SUITE SERVER REPOSITORY.....	14
<i>Creating the CVS Suite Server Repository</i>	14
<i>Create a Project using CVS Suite Studio</i>	15
Step 1: Right click the server/repository and select New Directory	15
Step 2: Give the the new directory a name	15

DEFECT TRACKING INTEGRATION	16
<i>Activating the CVS Suite Server connection to Bugzilla</i>	16
BUILD AND PROMOTE INTEGRATION	17
<i>Activating the Promotion connection to Oracle</i>	17
DATABASES, SCHEMAS AND USERS.....	17
CREATE A WORKSPACE USING CVS SUITE STUDIO	18
<i>Drag a CVS Suite Module from the server to the My Documents folder</i>	18
PREPARING SQL NAVIGATOR	18
<i>CVS client</i>	18
<i>Install Team Coding Tables</i>	19
<i>Team Coding Administrator</i>	19
SQL NAVIGATOR INTEGRATION	20
<i>Activating the Team Coding connection to CVS Suite</i>	20
<i>Creating Code Control Groups</i>	22
Step 1: Add the Code Control Group.....	23
Step 2: In the New Group window set a Group Name and browse for a project.....	23
Step 3: Browse for a project and the local workspace folder / sandbox.....	24
Step 4: Ensure local path is set to the Workspace Location (Sandbox) created earlier.....	24
Step 5: Complete the New Group dialog.....	25
Step 6: On the Code Control Group dialog Add DB Object Mask.....	25
Step 7: Use the default Mask Properties	25
Step 8: The masks associate Files and DB Objects to CVS Suite Workspaces.....	26
Step 9: The Code Control Group is now created	26
<i>Create a new Database Object</i>	27
<i>Modify a Database Object and track the change in Bugzilla</i>	31
PROMOTING A CHANGE TO TEST OR PROD	33
<i>Drag a CVS Suite Promotion Level from the server to the My Documents folder</i>	33
<i>Choose a project or individual files/objects to promote</i>	34
<i>View promotion history</i>	37
ROLLBACK A CHANGE TO TEST OR PROD.....	38
<i>Promoting a previous release or version</i>	38
<i>Rolling back any set of changes</i>	39
IMPORT EXISTING DATABASE OBJECTS INTO VERSION CONTROL.....	41
<i>Adding missing objects to Version Control</i>	41
Step 1: use the Team Coding menu to start Code Control Groups.....	41
Step 2: Use the Export to VCS option on the Code Control Groups toolbar	41
Step 3: select the objects that you wish to import.....	41
Step 4: Select the export option: Add objects not existing in the repository.....	42
Step 5: Objects already in VCS are skipped and missing objects are added to the VCS	42
Step 6: When the export is complete press the Close button	42
BEGIN WORKING ON RELEASE 2 WHILST MAINTAINING RELEASE 1	43
<i>When to create branches</i>	43
<i>How to begin work on release2 (branching)</i>	43
Step 1: Ensure all current work on Trunk is checked in	43
Step 2: Use CVS Suite Studio to Create the Branch	44
Step 3: Name the branch.....	44
Step 4: Create a new Workspace (Sandbox) for the branch.....	45
Step 5: Create a new Database for the new Branch.....	45
Step 6: Open a New Session in SQL Navigator.....	46
Step 7: Use the Server Side Installation Wizard	46
Step 8: Activate the Team Coding connection to the branch.....	47
Step 9: Create Code Control Groups for the new database/branch	49
Step 10: Use the Import to Database Code Control Groups Option.....	52
Step 11: Import to Database selection.....	52
Step 12: Use the Import Options	53
MOVE PATCH FROM RELEASE 1 TO RELEASE 2.....	54
<i>Release 1 and Release 2 – Branch and Trunk</i>	54
<i>Working with Patching and Merging</i>	54
<i>Applying patch to release2</i>	55

Step 1: Use the CVS->Merge menu.....	55
Step 2: Enter the name of the maintenance branch	56
Step 3: Commit the Merge	56
Step 4: Reload SQL Navigator from VCS.....	57
<i>Viewing the Merge History</i>	58
CVS ARCHITECTURE	59
CLIENT / SERVER ARCHITECTURE	59
MULTI SITE, REPOSITORY REPLICATION AND WAN PERFORMANCE.....	60
Support of CVS Suite Server and your Technical Account Manager.....	60
<i>Multi Site Solutions</i>	60
APPENDIX	61
<i>build_make.bat</i>	61
<i>build_make.sh</i>	63
<i>Makefile</i>	64
makefile.mak (for Microsoft Windows).....	64
makefile.mak (for Linux).....	64

Overview

This document gives a technical oriented overview of how to manage changes to Oracle stored procedures and other objects using CVS Suite 2009, and in particular:

- Development integration with Quest SQL Navigator 6.4
- Support for a Dev -> Test -> Production development cycle
- Support for parallel Development (branching)
- Integration with change management using a defect tracking system such as Bugzilla, Mantis or Atlassian Jira.

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
- 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

Development integration with Quest SQL Navigator 6.4

SQL Navigator software by Quest incorporates some Team Coding features where developers checkin/checkout database objects that they are working on. SQL Navigator supports 3rd party version control systems such as CVS Suite 2009 which can be integrated with Team coding:

- Copies of Database Objects are stored in CVS Suite
- The person making the change using SQL Navigator is prompted for a comment
- CVS Suite Server is automatically updated when a Database Object is modified using SQL Navigator

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
- Promote only individual objects, all objects in a *bug* or *job*, or all changes
- Managers can script actions to occur at specific promotion events, including updating different databases when scripts 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: *All About CVS*.

Installed Components

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

<i>Software Installation</i>		
	<i>Version</i>	<i>Location</i>
CVS Suite Server	2009.4087	D:\program files\cvs suite\cvsnt\
SQL Navigator	6.4	D:\program files\Quest Software\
Oracle	11g	D:\app
GNU Make	3.8	From http://unxutils.sourceforge.net/
Operating System	Windows XP	D:\windows

Customised installations

This whitepaper shows how CVS Suite can be used to manage and control change in an Oracle and SQL Navigator development environment. If you are a very small team (1-8 people) we expect that this may be sufficient for you to get a production system running, however larger teams will usually require that the workflow and processes described here be significantly adjusted to suit their requirements.

Please do not ask for this document to be customised for your requirements, instead our sales team will provide you with a quote for enough licenses for your team plus *On Site Installation and Configuration* and *CM Design and CVSNT Administration Training* – when these on site activities are complete we will provide you with the customised documentation.

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 book, 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 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.

What are Branches, Magic Branches and Vendor Branches

Branches, Magic Branches and Vendor Branches facilitate very powerful ways of organising the planned development of your software projects. Often software development managers indicate that they do not require branches in their solution. Take the time to carefully read this section since once you understand what branches, magic branches and vendor branches are and what they can facilitate you may find that it is very useful to you.

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 organization 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 organization 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 stage “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:

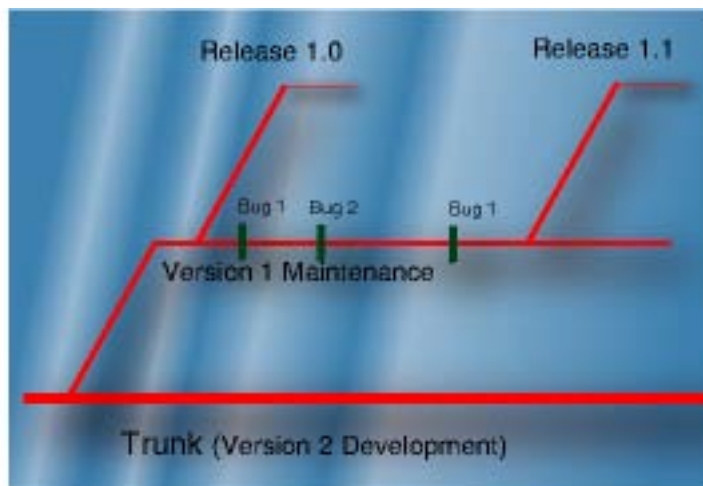
- Version 1
- Fix bug 1
- Fix bug 2
- Release Patch 1
- Fix bug 3
- Release Version 1.1

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

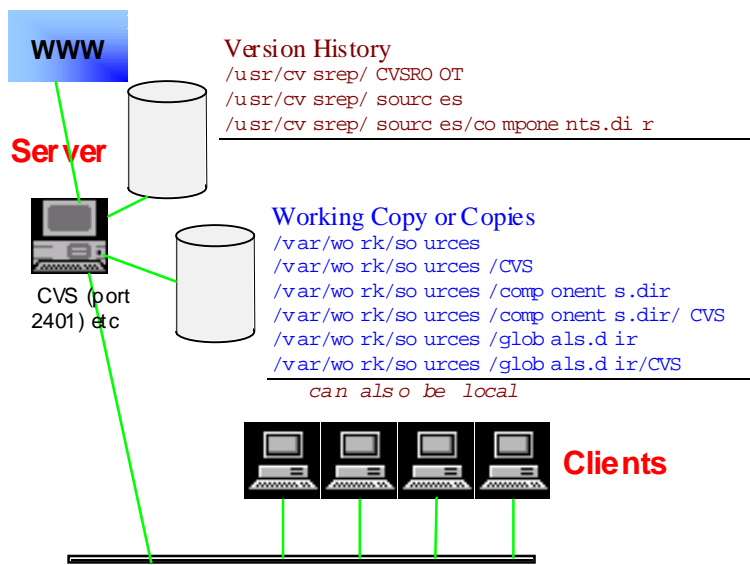
These terms are commonly used in business English but have specific technical meanings for our products and in this whitepaper:

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 stored procedure `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 database instances, eg: CRM_DB, ORDER_INV_DB, STATS_DB etc

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



Working Copy or Workspace (some people call this the *sandbox*)

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 database instance (e.g.: CRM_PROD_DB) or a database schema (e.g.: CRM_PROD_SCH).

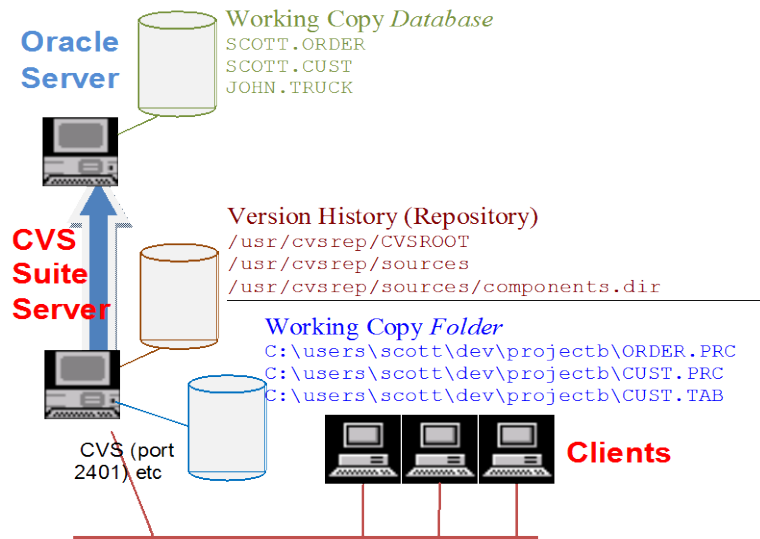
Whilst a working copy is always *linked* back to a repository, making a change in a working copy is controlled by the native permissions of target device (e.g.: a disk or a database instance or database schema). Changes to a working copy may be discarded. Changes to a working copy may be kept (or committed) 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 `custidproc` in dev but not in prod).

Working Copy Folder versus a Working Copy Database

Version Control tools like CVS Suite cannot directly track changes to database objects – instead CVS Suite and SQL Navigator create a file on a disk that is identical to the object in the repository.

You DO NOT EVER need to manually load the files into the database.

The SQL Navigator, Oracle and CVS Suite Software handle the process of synchronising these files to the database. SQL Navigator uses *Code Control Groups* to associate a CVS Suite Repository with a *Working Copy Folder* and a *Working Copy Database*.



Moving Objects from One database to another Database (eg: Dev to Test)

A database administrator may typically use Quest SQL Navigator, or Quest Toad or perhaps a tool from Embarcadero Technologies to move objects from one database to another.

When you are using a change control process this is insufficient because:

- It is possible to introduce human error and perhaps move the wrong version from test to production
- The change is not controlled or tracked

Therefore the Database Administrator or QA Manager manually selects the correct items to move and uses the Promotion procedure to automatically move the objects and also record the change in the change management system.

Using the change control procedure, the Database Administrator or QA Manager can move:

- Individual objects, or
- all objects identified by a change number (bug number), or
- all objects changed since the last move.

See the section above *Theory* and *Promotion model* as well as the sections below *Build and Promote Integration* and *Promoting a change to Test or Prod* for more information on this topic.

CVS Suite and Oracle Workflow

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:

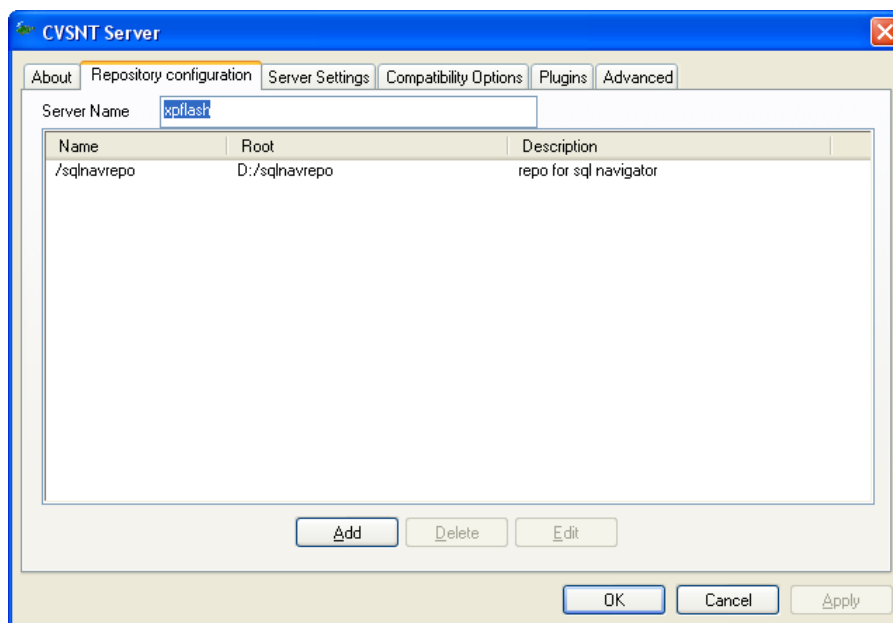
- A Promotion Model of Trunk (Dev) -> TestDB -> ProdDB
- Three databases: Example for Dev, TestDB and ProdDB
- Two privileged users for updating TestDB and ProdDB on promote, these privileged users can choose what individual objects to promote, or promote all objects with a specific bug number, or promote all changes.
- A script to automatically update TestDB or ProdDB on promote
- CVS Suite server integration with Make
- CVS Suite server integration with a single instance of Bugzilla defect tracking
- CVS Suite client SCCI Integration with SQL Navigator

The build scripts are 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:



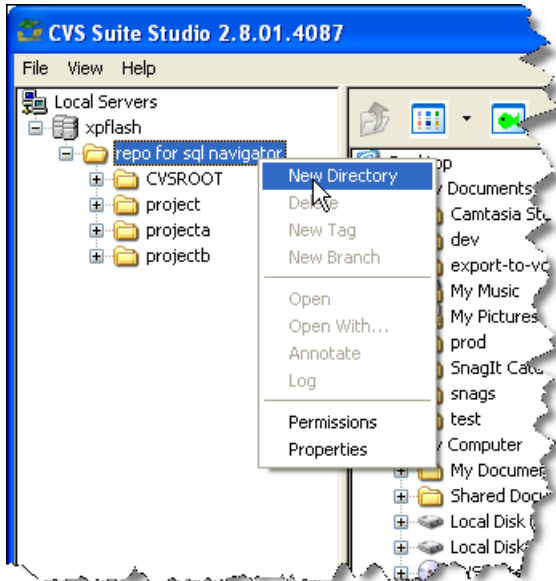
An alternative command line interface is available for Unix, Linux and Mac OS X. Refer to the eBook *All About CVS* for more information for both Windows and Unix.

Create a Project using CVS Suite Studio

Create a CVS Suite Module on the server using CVS Suite Studio.

Step 1: Right click the server/repository and select New Directory

On the left pane find the server and repository you are using and right click and select New Directory:



Step 2: Give the the new directory a name

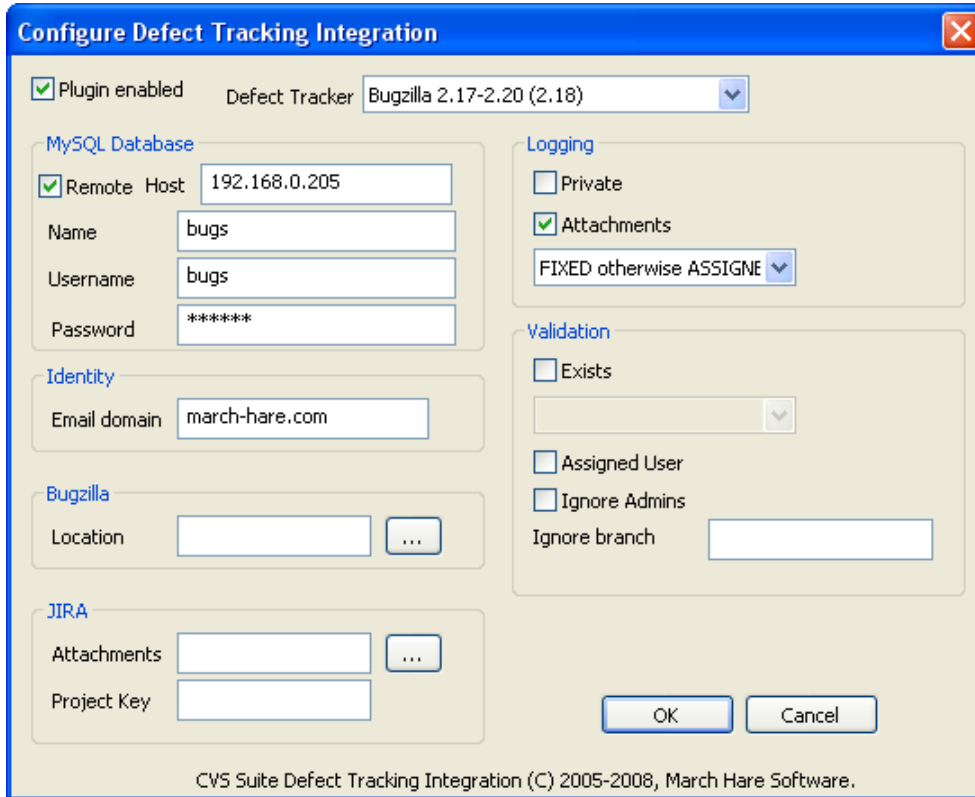
Name the new directory, eg: *projectc*



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 a single instance of Bugzilla:



Configure Defect Tracking Integration

Plugin enabled Defect Tracker: Bugzilla 2.17-2.20 (2.18)

MySQL Database

Remote Host: 192.168.0.205

Name: bugs

Username: bugs

Password: *****

Identity

Email domain: march-hare.com

Bugzilla

Location: [] [...]

JIRA

Attachments: [] [...]

Project Key: []

Logging

Private

Attachments

FIXED otherwise ASSIGN

Validation

Exists

[]

Assigned User

Ignore Admins

Ignore branch: []

OK Cancel

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

On Unix, Linux and Mac OS X 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.

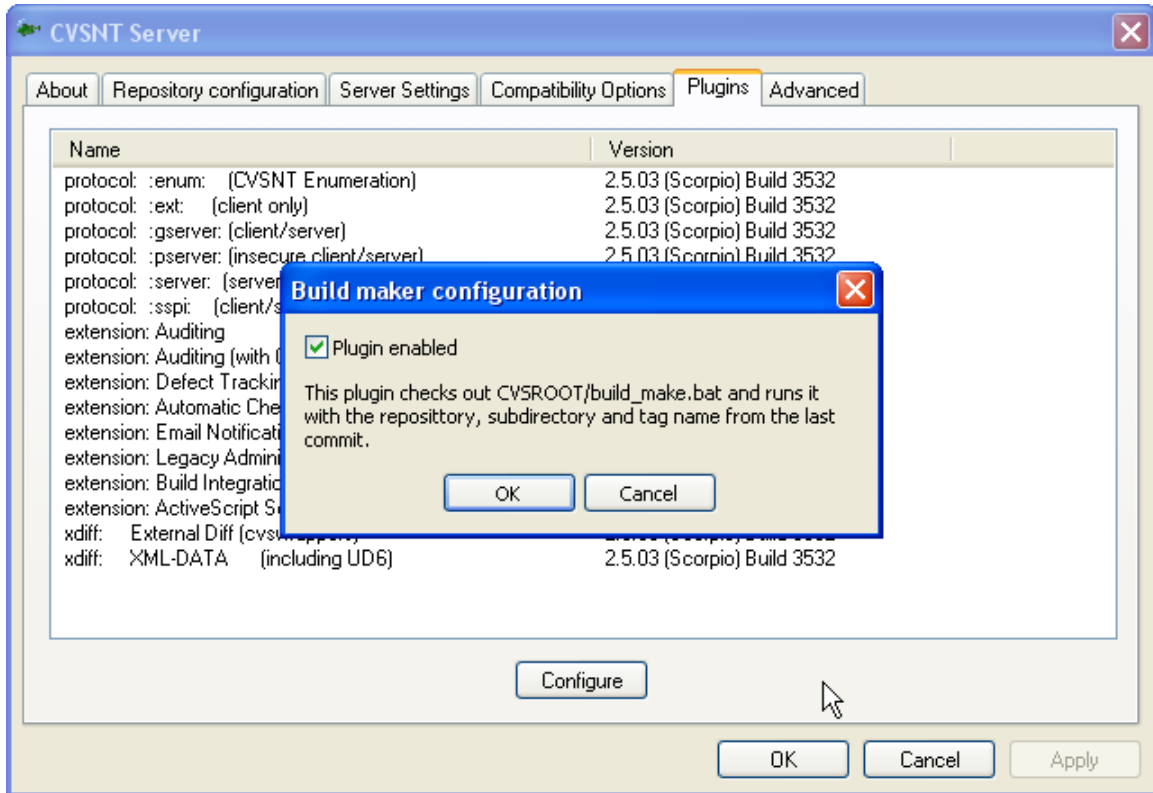
Rather than have separate instances of Bugzilla for different teams, products or projects we recommend that you use a single instance and create separate Products within Bugzilla to which you can assign varying user rights and different component sub-products and versions/releases.

If you require support for multiple Bugzilla instances please contact us for a quote.

Build and Promote Integration

Activating the Promotion connection to Oracle

The following steps activate the Promote connection between CVS Suite Server and Oracle:



Also see the *build_make.bat* configuration file in the appendix.

Databases, Schemas and Users

Each of the organisations that we have helped with change control of Oracle database objects has a unique system to distinguish one project, release and environment from another within the Oracle database.

Example 1: Dev/Test/Prod could be in separate databases, and different applications each have their own schema. When release 2 is done the databases are copied and when release 1 is retired the 'old' database archived and deleted.

Example 2: Dev/Test/Prod could be in a single database in different schemas. Whilst each developer may also have their own schema - all the developers have access to the *dev* schema which they copy their work to when completed.

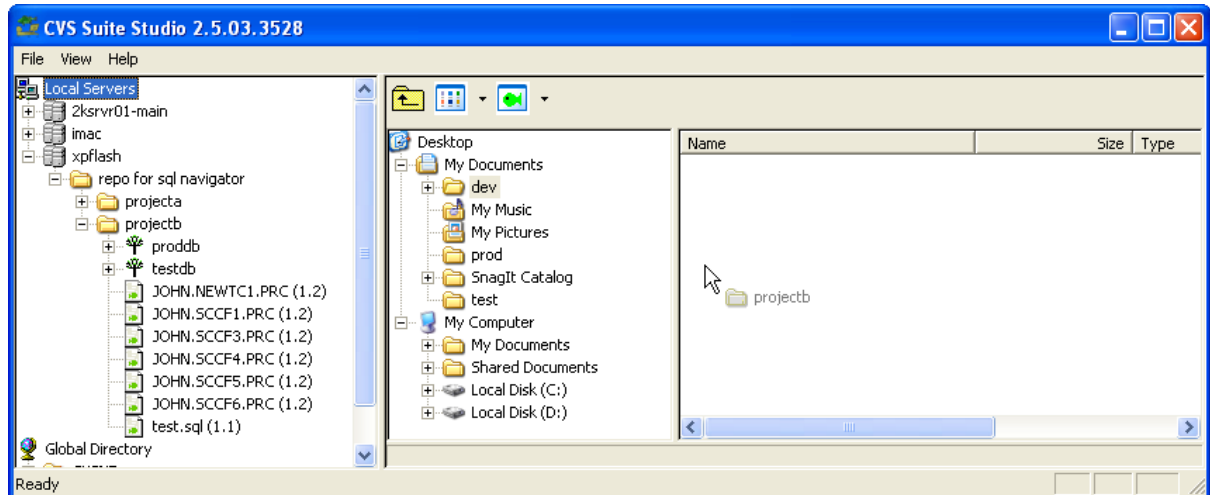
CVS Suite supports many variations of how to use Database, Schemas and Users provided that it is possible to distinguish dev/test/prod and release1 from release2.

This whitepaper uses different databases for each devr1/testr1/prodr1 and devr2/testr2/prodr2. Please do not ask us for tailored examples, instead we will provide a quote for *On Site Installation and Configuration* plus *CM Design and CVSNT Administration Training*.

Create a Workspace using CVS Suite Studio

Drag a CVS Suite Module from the server to the My Documents folder

Dragging the CVS Suite Module you created earlier from the server to a folder on the PC. Doing this creates a workspace (also called a sandbox) that you can then associate with a Code Control Group in SQL Navigator:



Preparing SQL Navigator

This document cannot include all of the steps necessary to configure SQL Navigator; this guide is intended to be used by people already familiar with SQL Navigator including any administrative functions. Please read the book that Quest supply with the product: *SQL Navigator for ORACLE User's Guide* section *Version Control and Team Coding* and in particular the sub-section *Enabling Team Coding in the Oracle Instance*.

CVS client

Quest provide a CVS Client in SQL Navigator. This client appears intended for use only with CVS 1.x servers and March Hare Software do not support this client or support customers who use the SQL Navigator Client to connect to CVS Suite Server. CVS Suite client includes an SCCI interface with SQL Navigator which has been extensively tested and documented by March Hare Software.

Install Team Coding Tables

The SQL Navigator documentation provides details of how to configure the Oracle database to support team coding. In our own testing with SQL Navigator 6.2 and 6.4 we discovered that this can sometimes fail to create the tables, so here is an alternative technique:

```
Command Prompt
Microsoft® Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

D:\Program Files\Quest Software\SQL Navigator for Oracle\Install\>sqlplus
system/manager@orcl

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 8 08:11:39 2011

Copyright (c) 1982, 2007, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> @TCinstall.sql
.
```

Team Coding Administrator

To make changes to the Team Coding screens in SQL navigator you must have the correct The SQL Navigator roles. eg:

```
Command Prompt
Microsoft® Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>sqlplus system/manager@orcl

SQL*Plus: Release 11.1.0.6.0 - Production on Tue Mar 8 08:11:39 2011

Copyright (c) 1982, 2007, Oracle. All rights reserved.

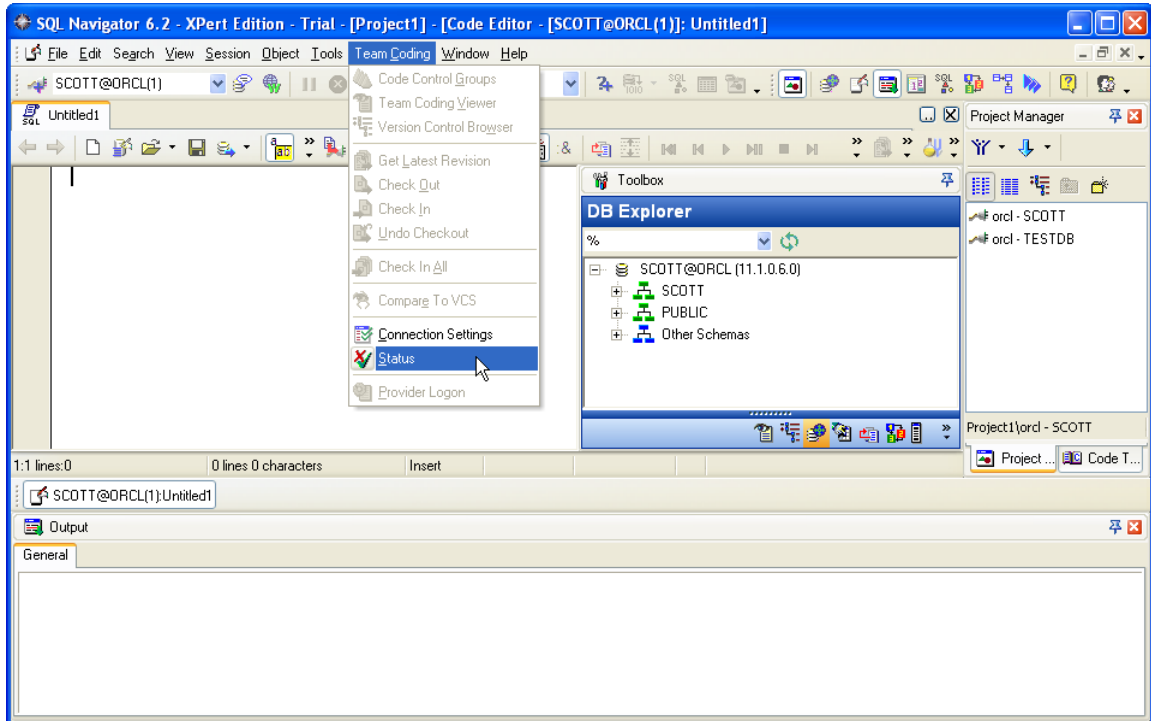
Connected to:
Oracle Database 11g Enterprise Edition Release 11.1.0.6.0 - Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> GRANT SQLNAV_ADMIN TO scott;
SQL> GRANT SQLNAV_MGR TO scott;
SQL> GRANT SQLNAV_LDRTO scott;
```

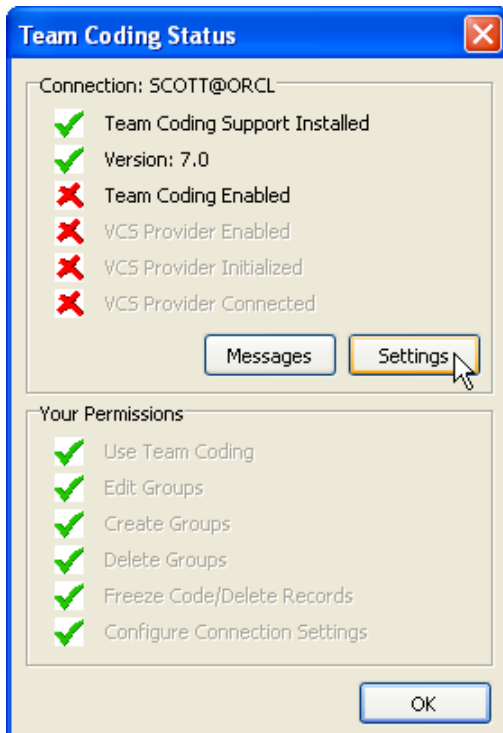
SQL Navigator Integration

Activating the Team Coding connection to CVS Suite

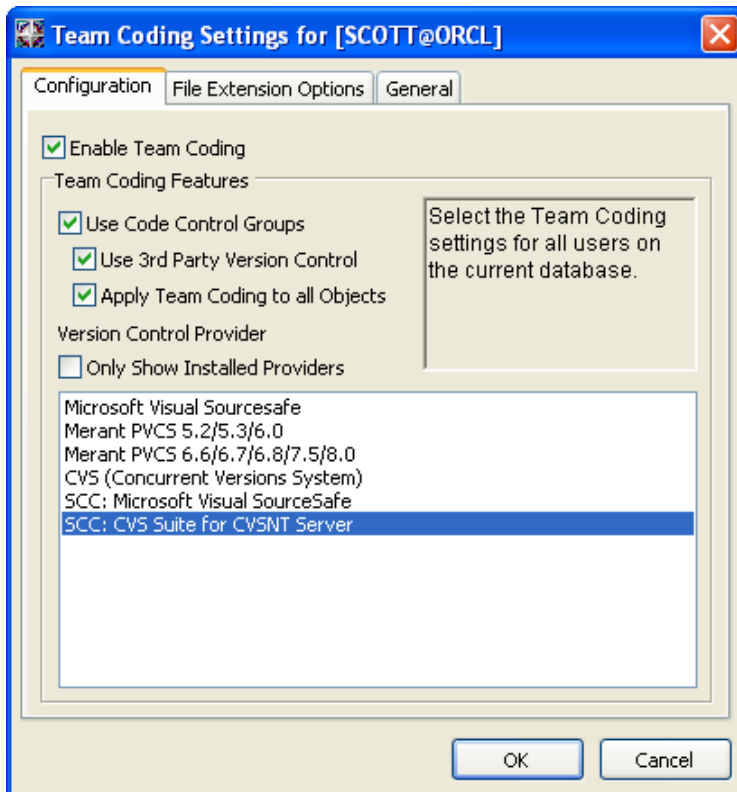
Activate the Team Coding connection between SQL Navigator and CVS Suite:



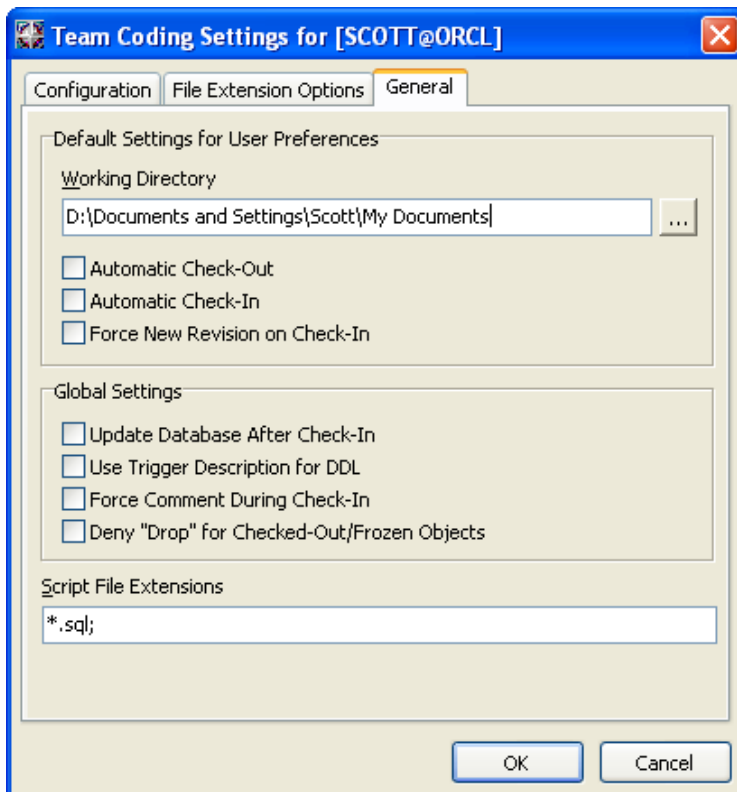
The status window shows that the integration is currently disabled:



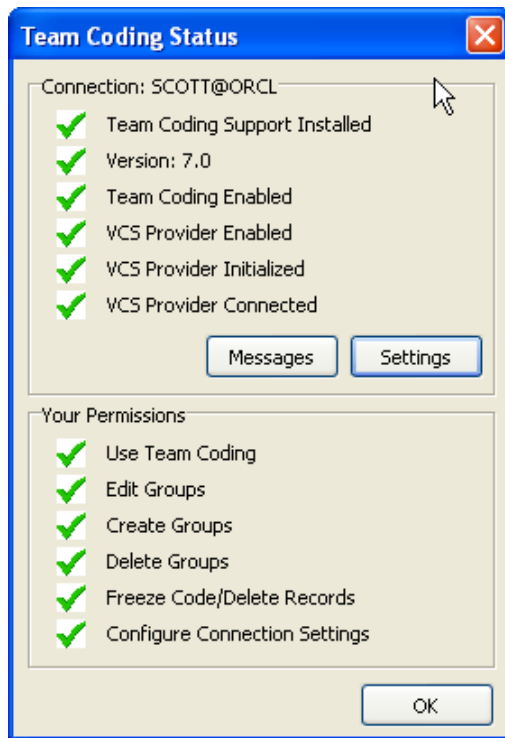
Enable the integration and choose the *CVS Suite for CVSNT Server* SCC provider



In the Team Coding Settings set the Working Directory:

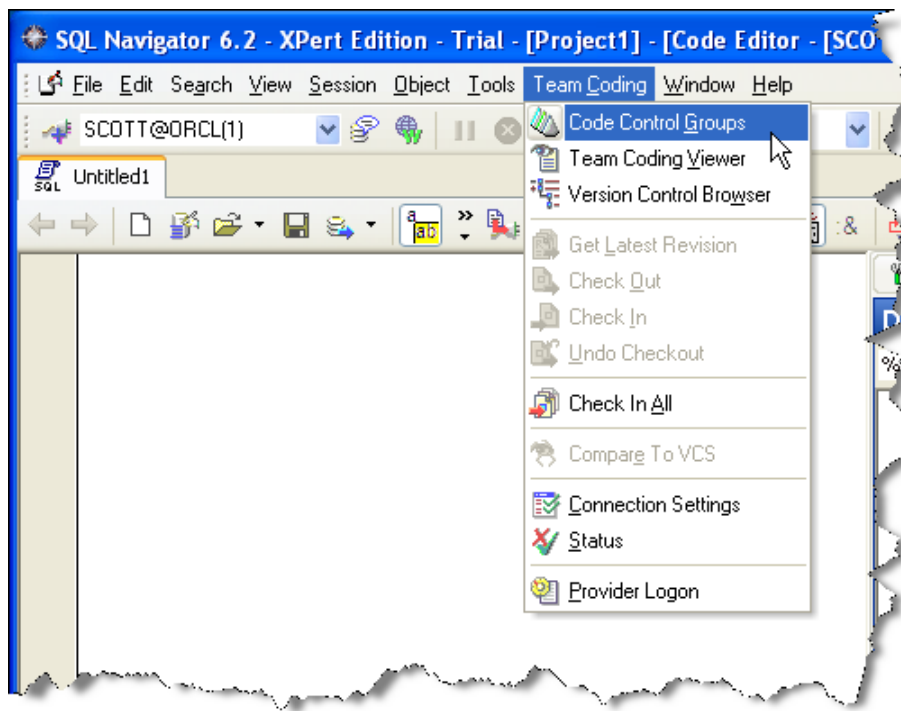


The Team Coding Status is now OK:

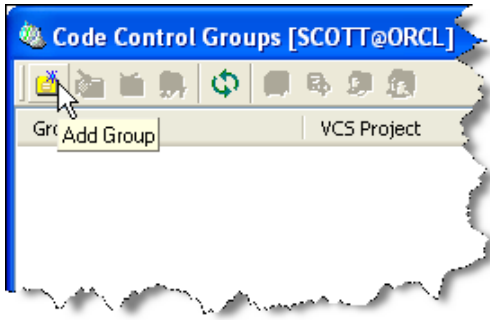


Creating Code Control Groups

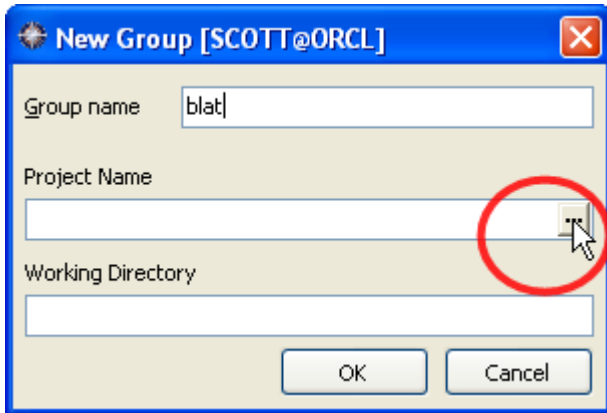
SQL Navigator requires at least one Code Control Group set up to associate your Database Objects with a CVS Suite Module and Folder based Workspace:



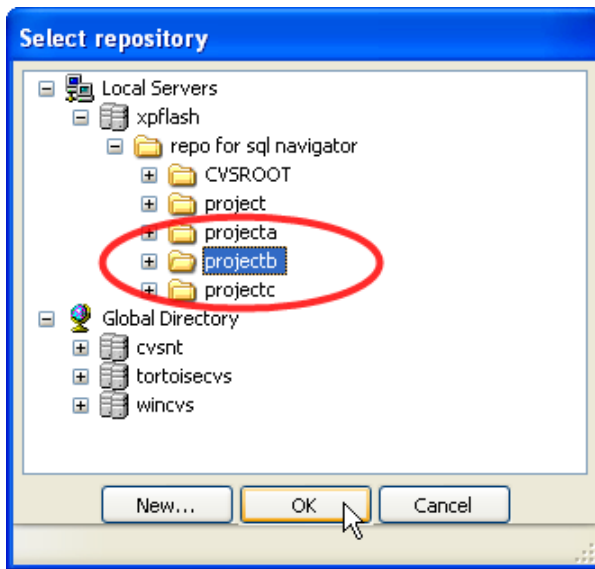
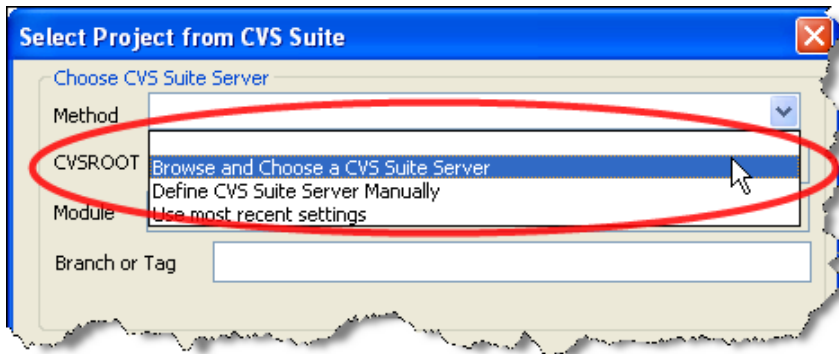
Step 1: Add the Code Control Group



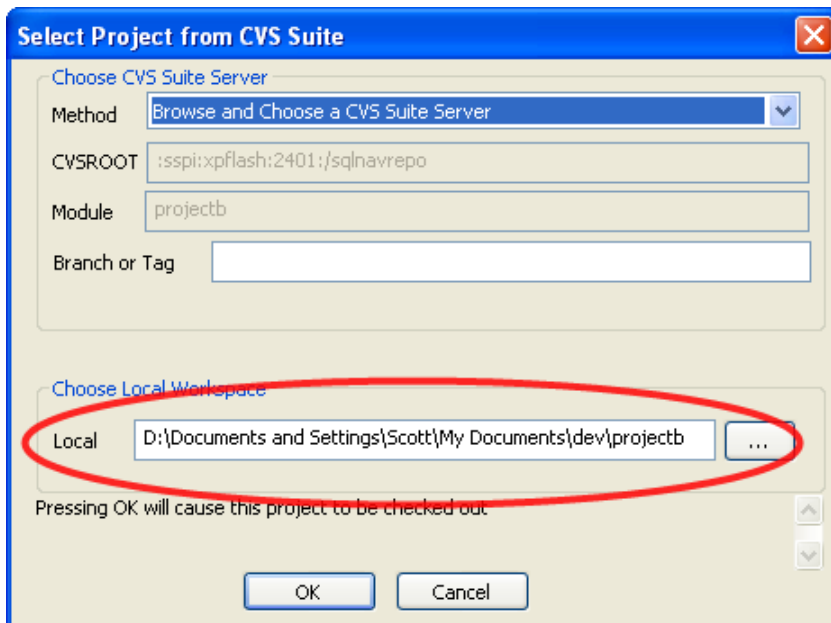
Step 2: In the New Group window set a Group Name and browse for a project



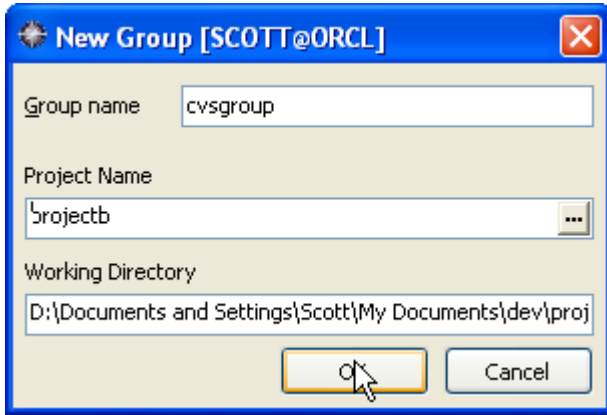
Step 3: Browse for a project and the local workspace folder / sandbox



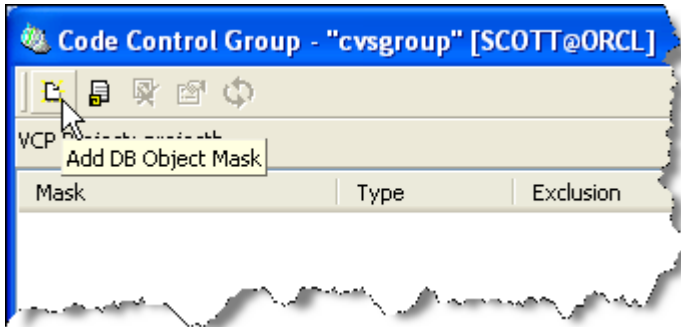
Step 4: Ensure local path is set to the Workspace Location (Sandbox) created earlier



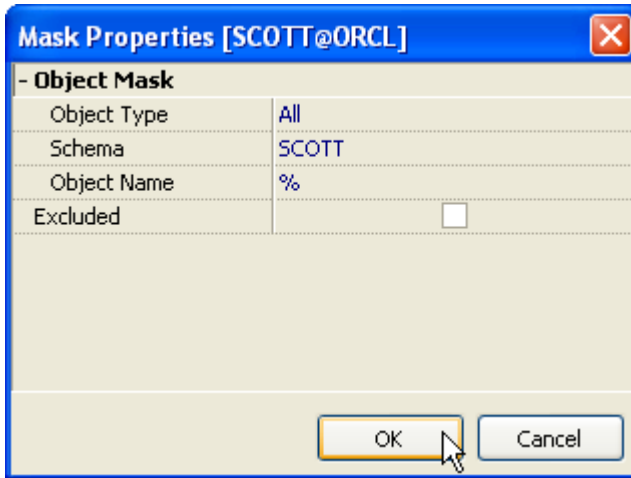
Step 5: Complete the New Group dialog



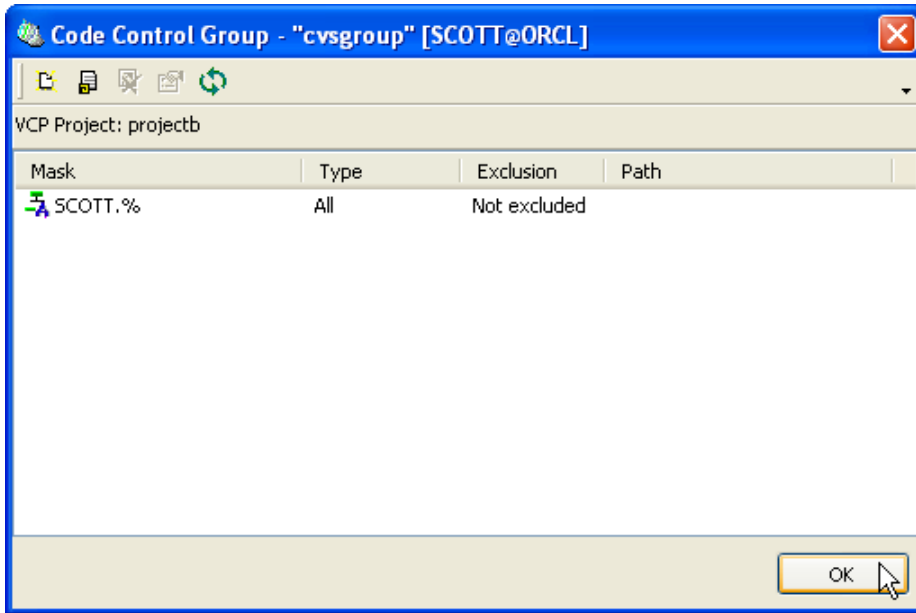
Step 6: On the Code Control Group dialog Add DB Object Mask



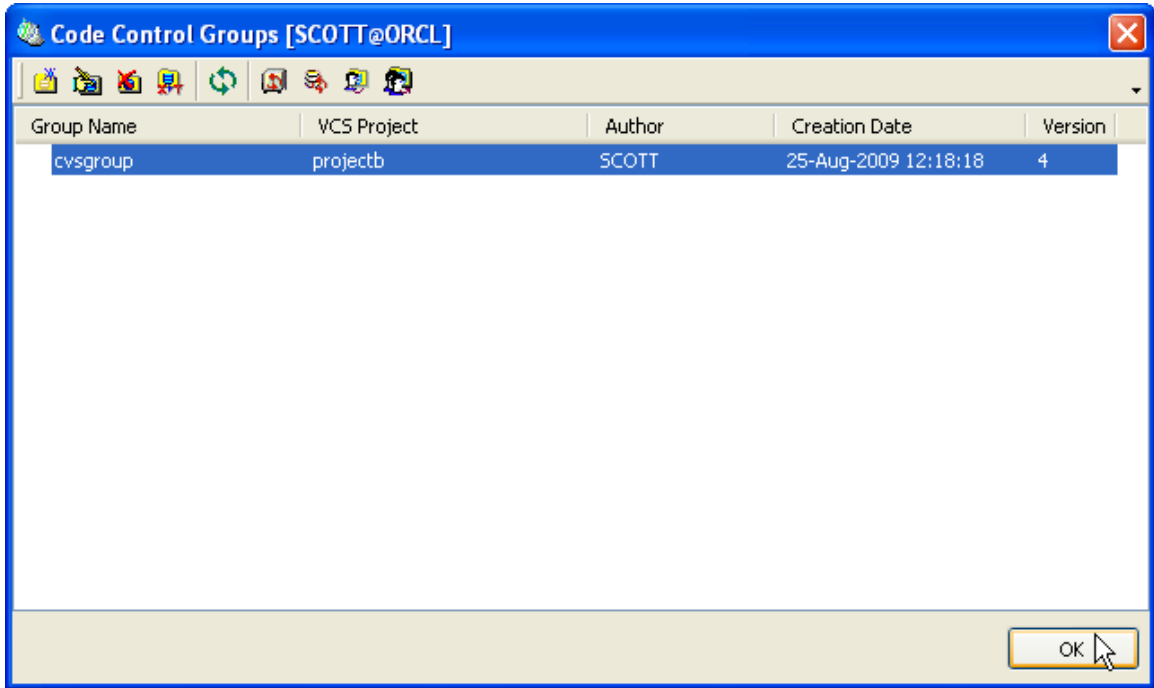
Step 7: Use the default Mask Properties



Step 8: The masks associate Files and DB Objects to CVS Suite Workspaces

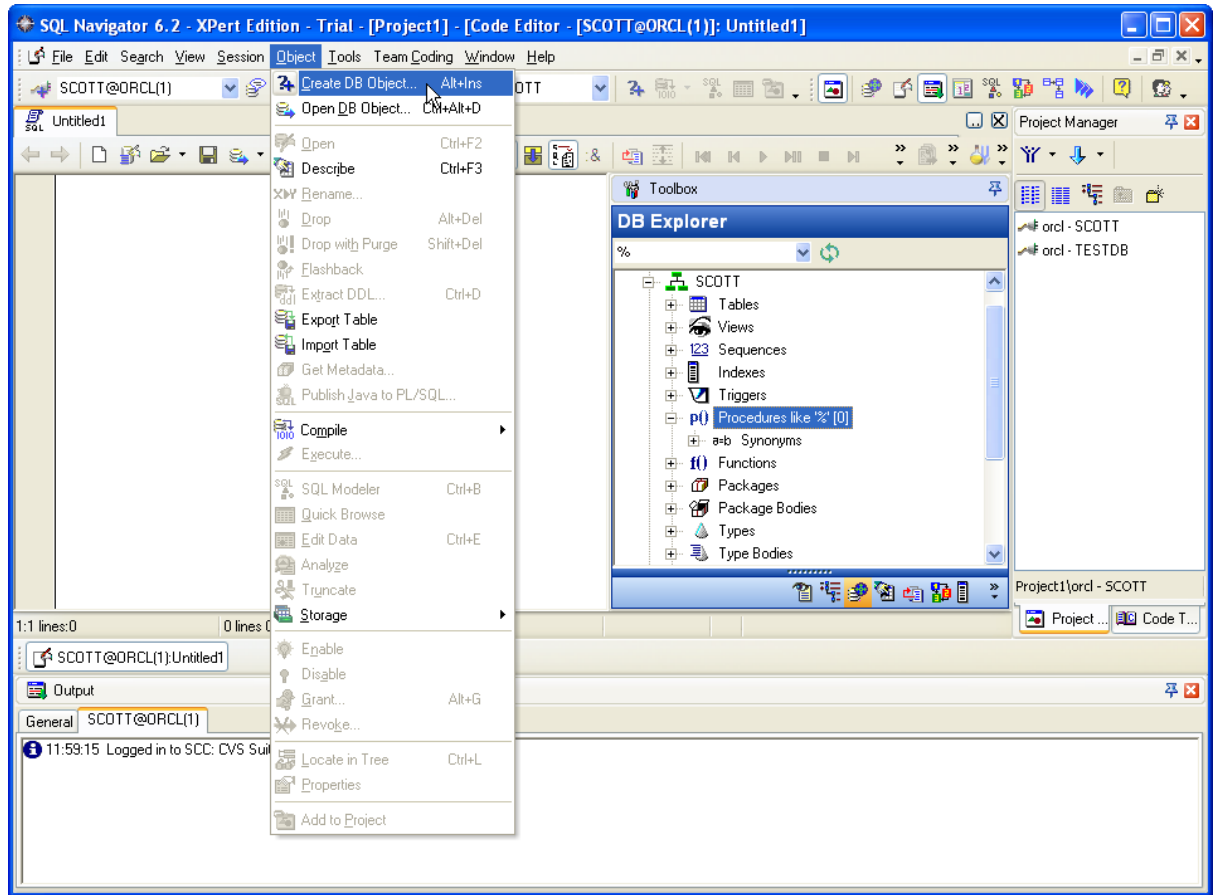


Step 9: The Code Control Group is now created

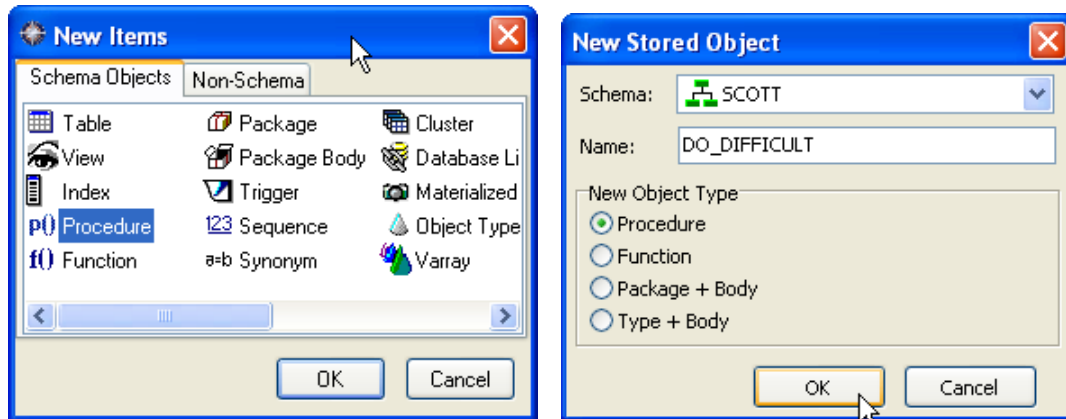


Create a new Database Object

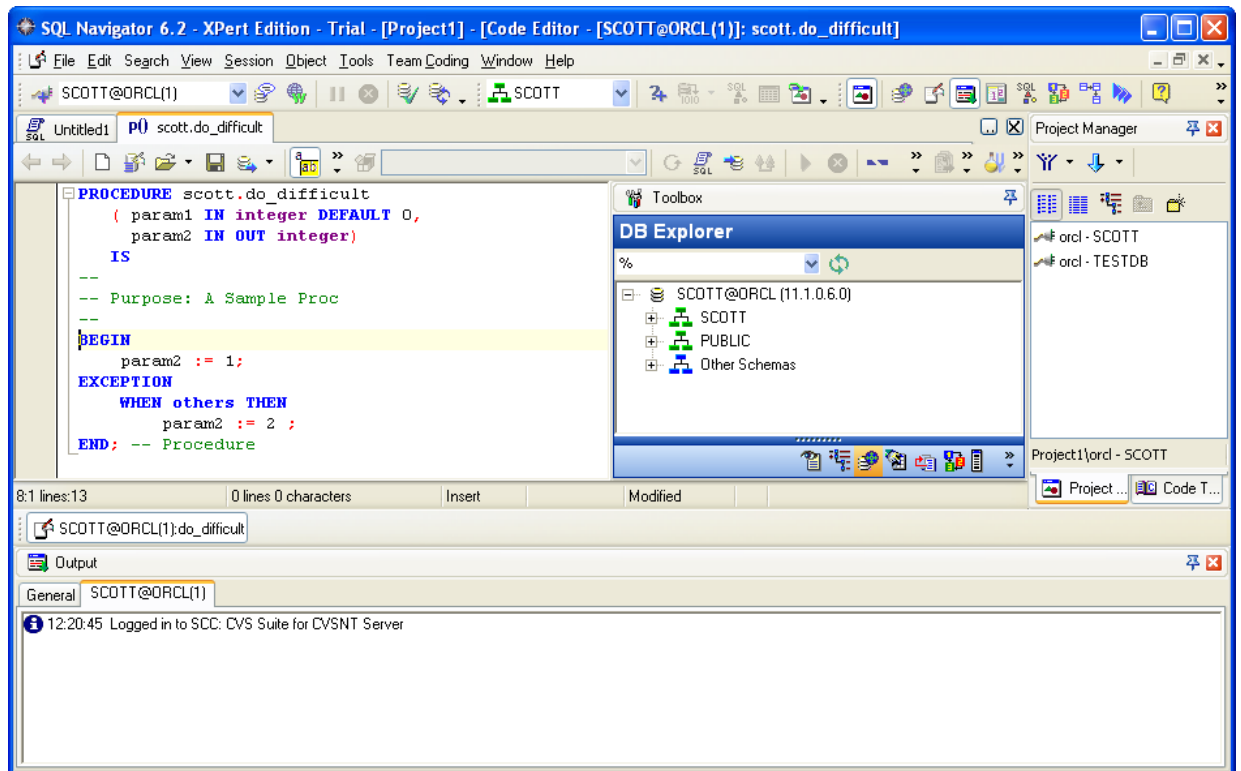
Creating a new Database Object automatically adds it to the CVS Suite server:



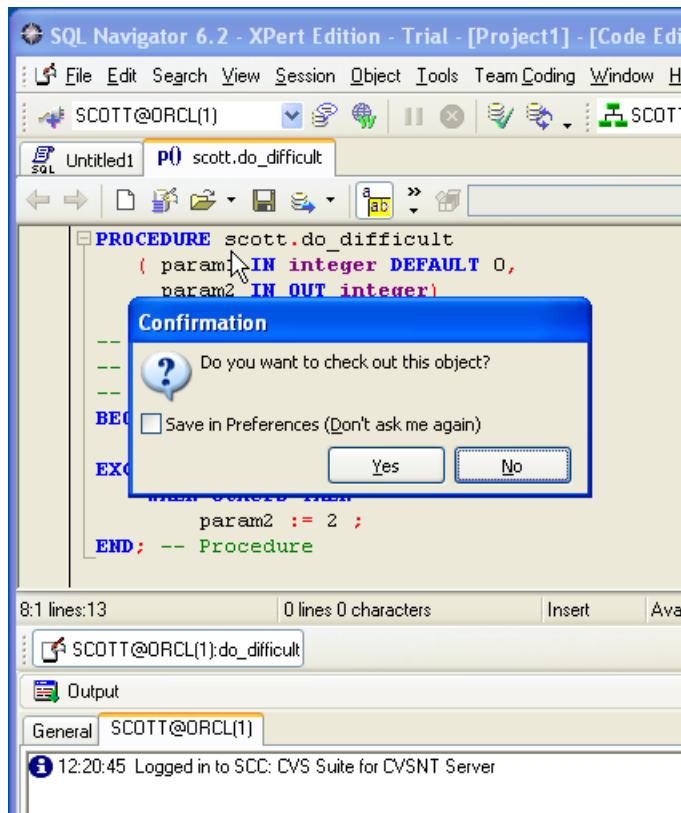
Create a new DB Object, pick an item such as a *Procedure* and name the New Stored Object:



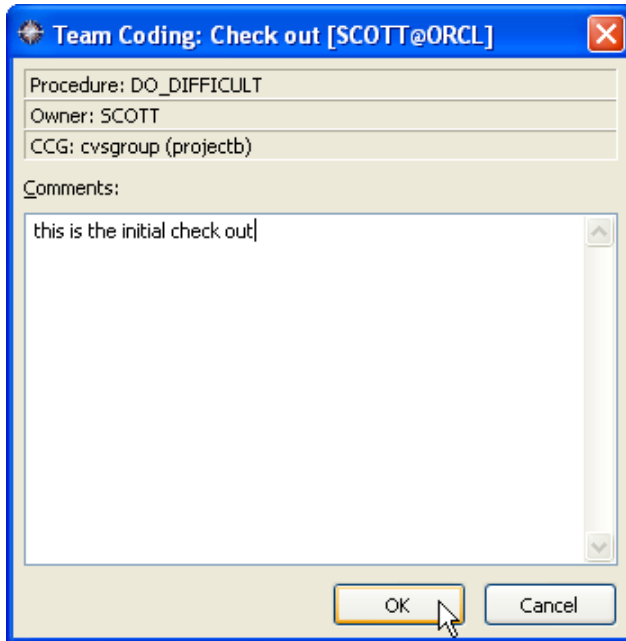
Write the Oracle Stored Procedure using the SQL Navigator tools:



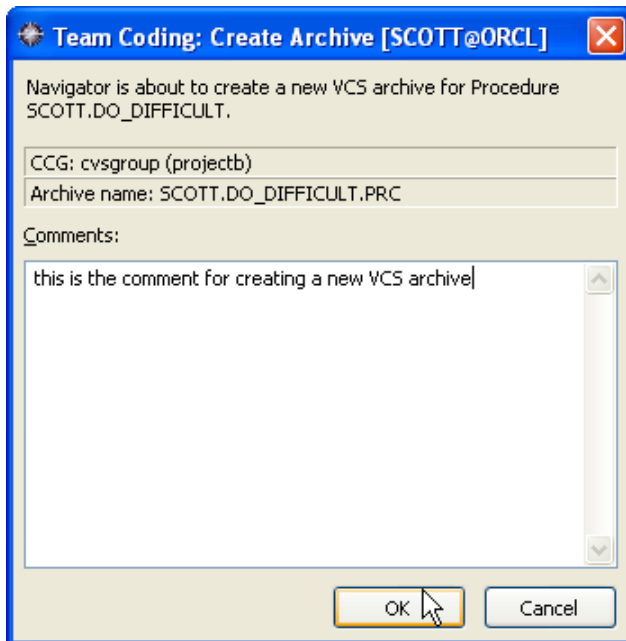
Press Save when complete:



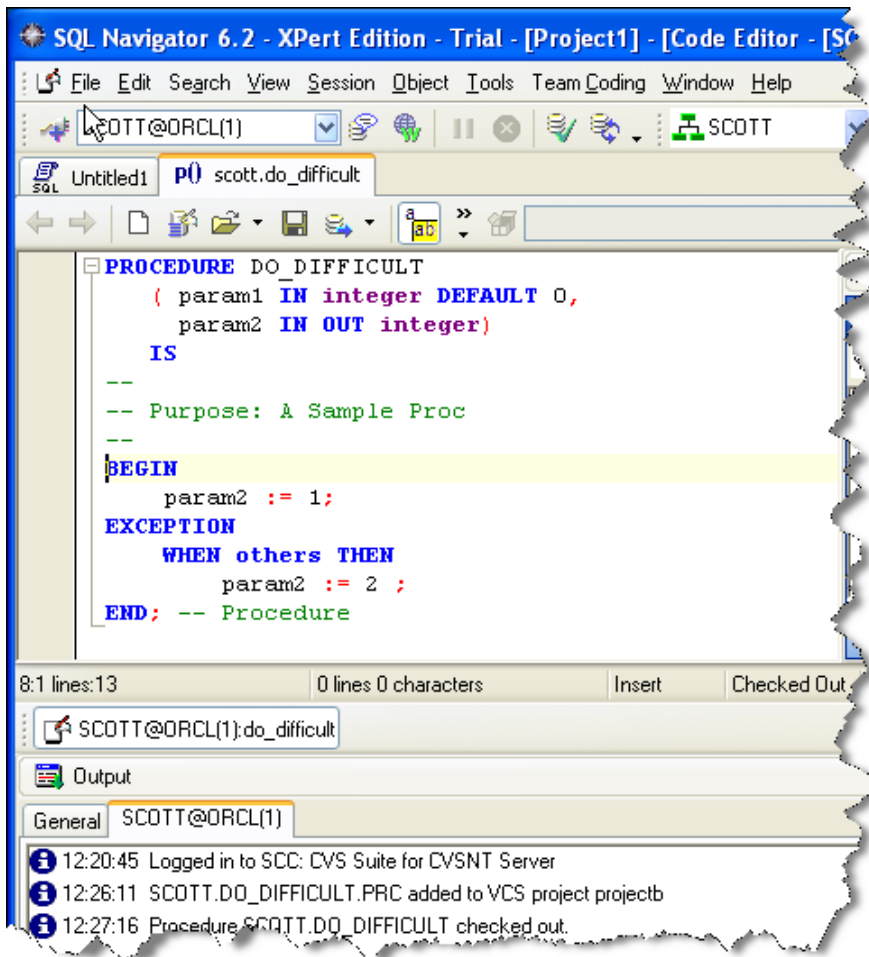
If you have enabled comments on checkout then you can enter a comment here:



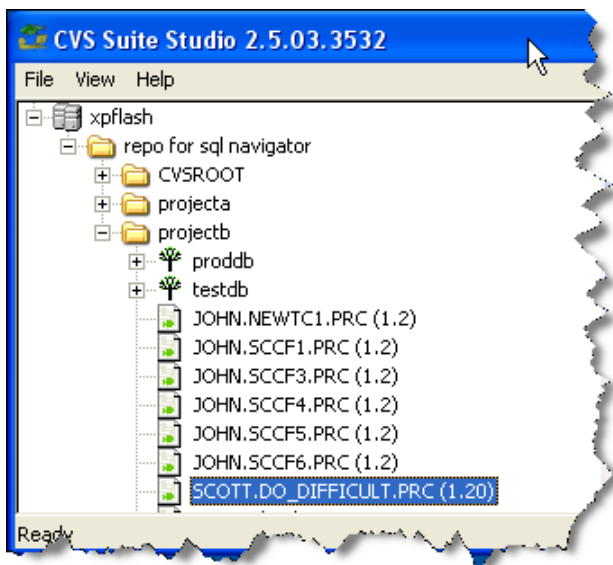
Enabling comments on checkin is recommended, the comment entered here can include keywords to link the change to the defect tracking system, like *Bug 1234*, or *STATUS=ASSIGNED*:



The status window will show when the process is complete:

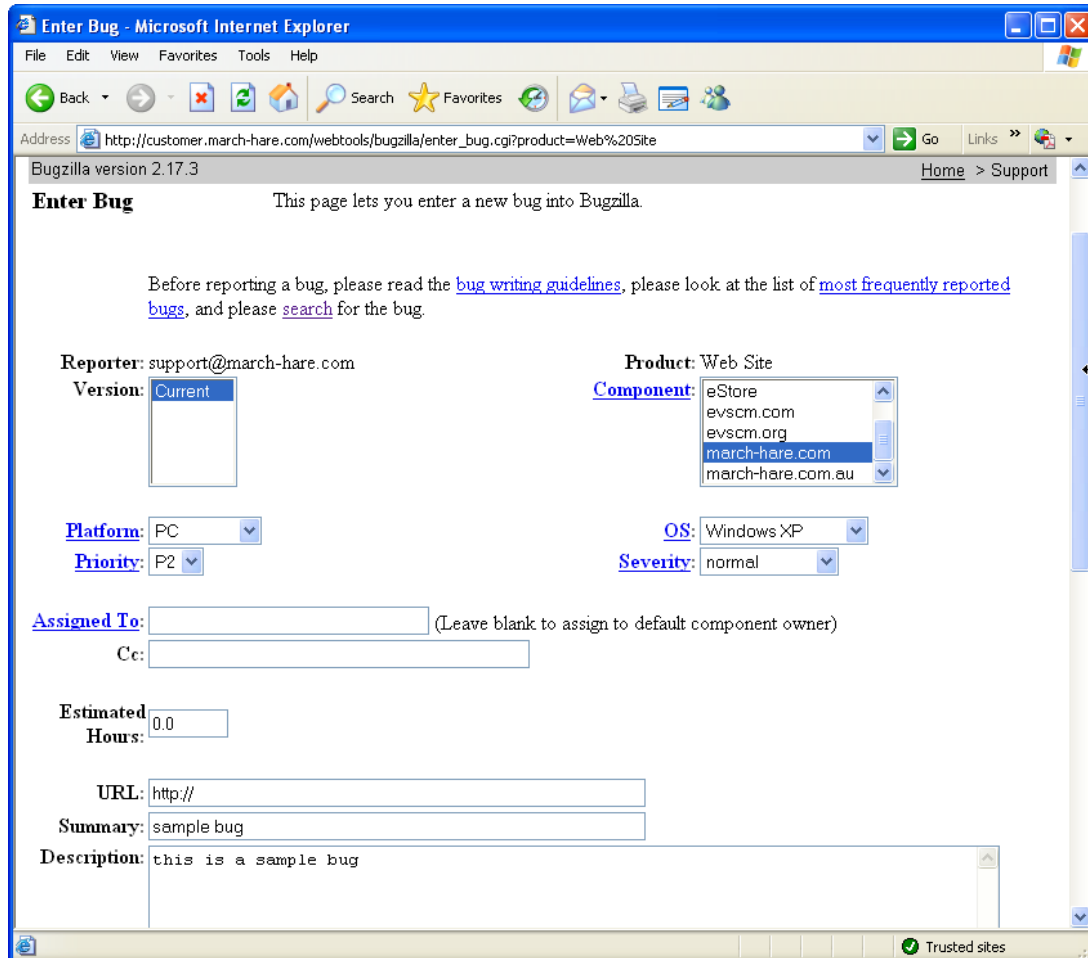


You can also view the copy of the procedure in CVS Suite Server by using the CVS Suite Studio client:

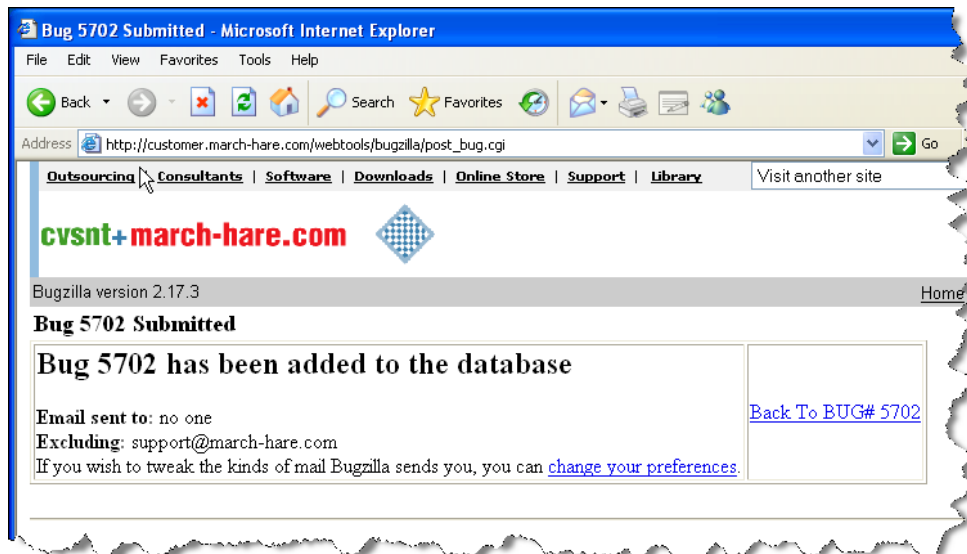


Modify a Database Object and track the change in Bugzilla

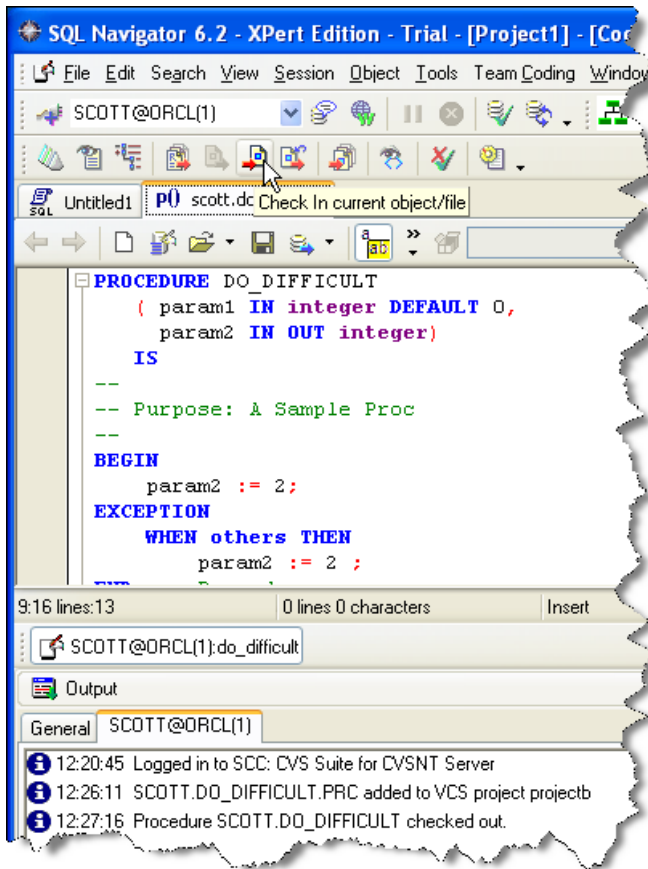
Most changes are prompted by a business event, such as a job/project, customer or employee request, so we begin this change by creating a change request (bug) in Bugzilla:



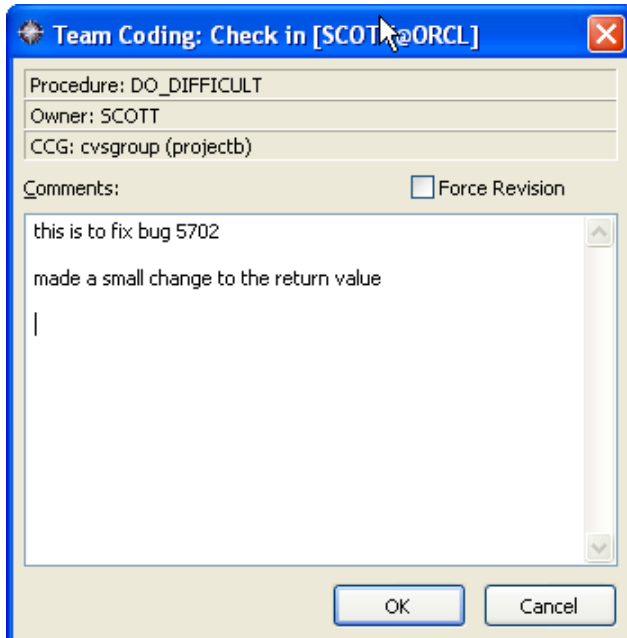
Submit the change request to obtain a bug/job number:



Modifying a Database Object automatically updates the history on CVS Suite server:



Pressing the *Check In current object/file* button on the toolbar begins the update process. The comment entered here can include keywords to link the change to the defect tracking system, like *Bug 1234*, or *STATUS=ASSIGNED*:



When the checkin is complete the Bugzilla database now also contains the comments as well as an attachment showing the changes that were made. From Bugzilla you can search these comments, for the name of the person who performed the commit, for the comment or for the name of the file:

Attachment	Type	Created	Actions
projectb/SCOTT.DO_DIFFICULT.PRC revision 1.2	patch	2009-08-25 21:35 AEST	Edit
Create a New Attachment (proposed patch, testcase, etc.)			View All

----- Additional Comment #1 From [Help Desk](#) 2009-08-25 21:35 AEST ----- Private

CVS Commit by user support@march-hare.com

```
projectb/SCOTT.DO_DIFFICULT.PRC 1.2      (+1,-1)

this is to fix bug 5702

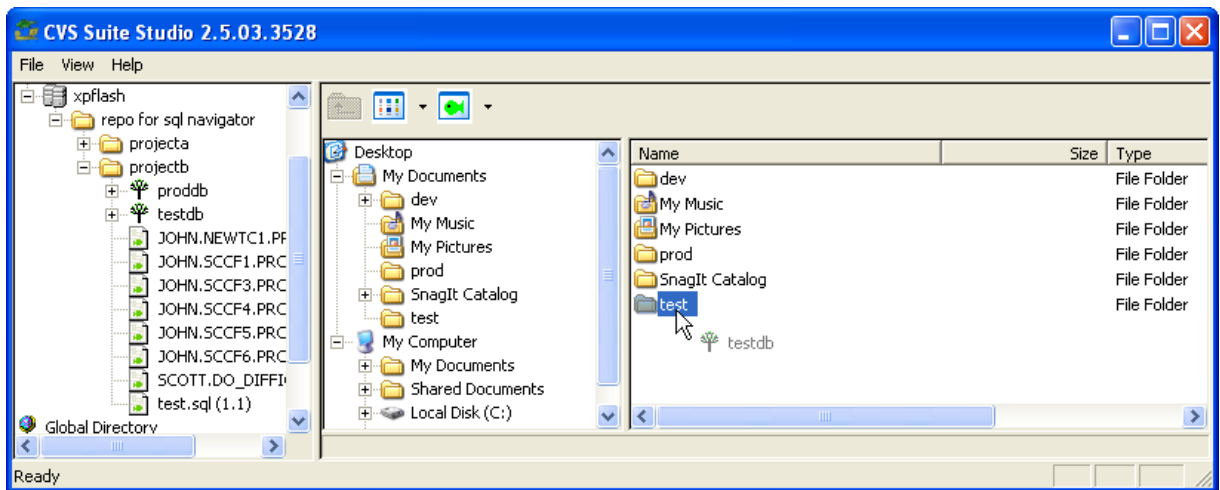
made a small change to the return value
```

Promoting a change to Test or Prod

SQL Navigator is a great tool for helping developers to author changes in a development environment, however releasing changes to a QA (test) or Production environment often has legal and/or legislative requirements on control and manageability. Industry best practice recommends that the process is automated and that ad-hoc changes are not allowed. To automate this process we will use promotion levels.

Drag a CVS Suite Promotion Level from the server to the My Documents folder

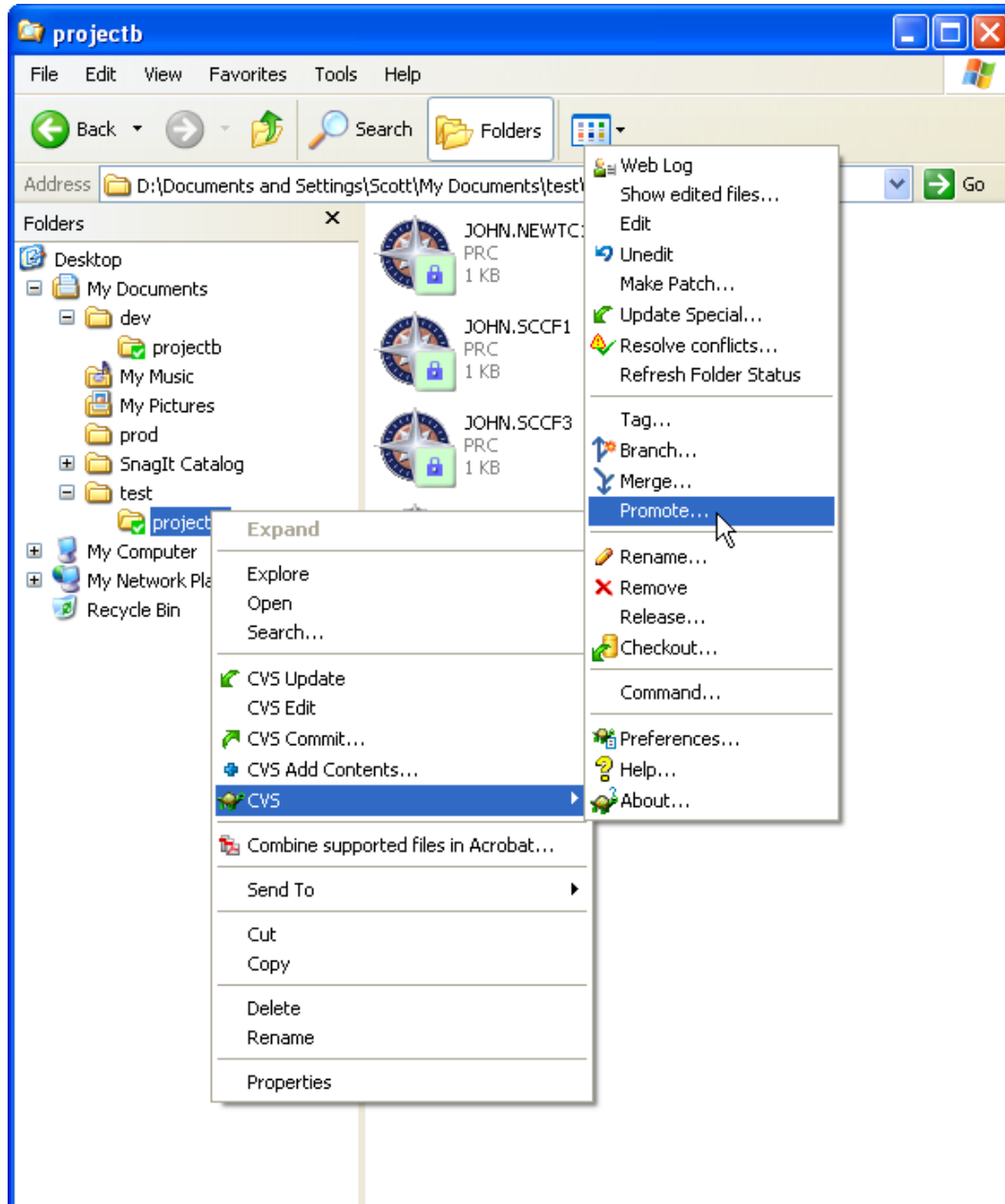
An authorised individual would be able to create a folder to perform the promote by dragging a CVS Suite Promotion level from the server to a folder on their PC:



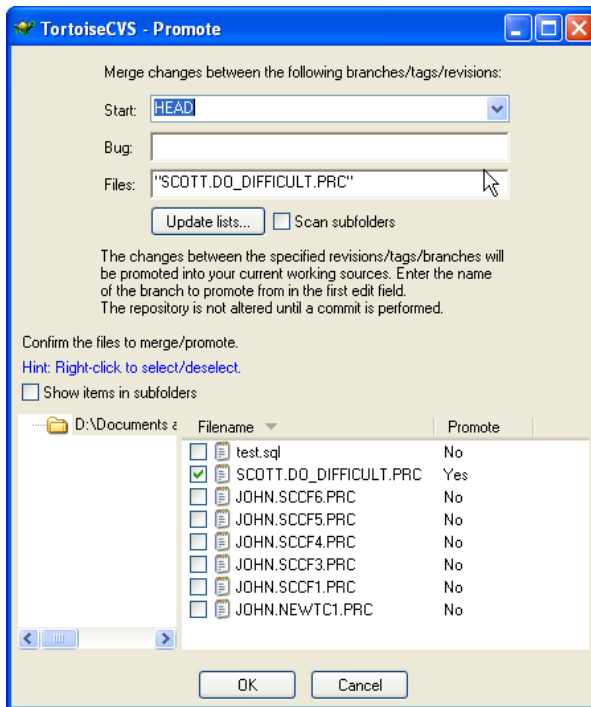
Choose a project or individual files/objects to promote

Right clicking on the folder from any application – including Windows Explorer displays the CVS Suite Explorer context menu. You can view the history of any file or DB Object as well as perform administrative functions such as Promote

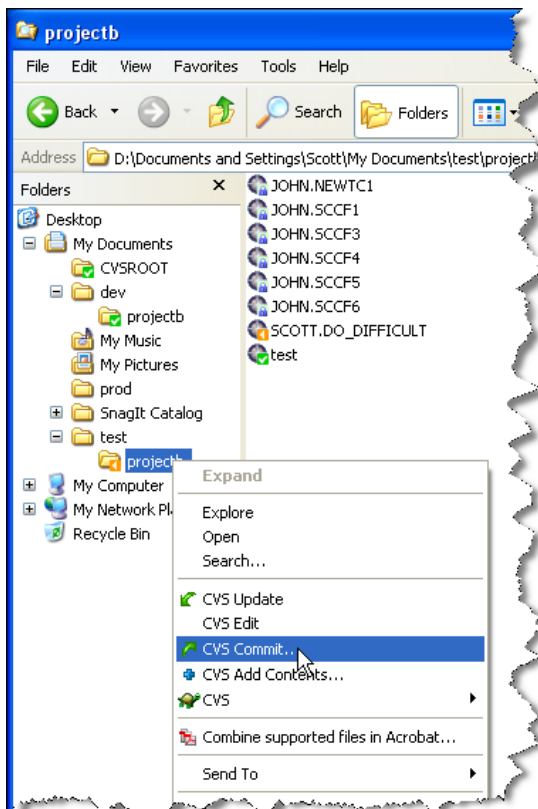
The authorised user can choose a project or an selection of files to promote using the CVS Suite Explorer Context menu as shown in this screen:



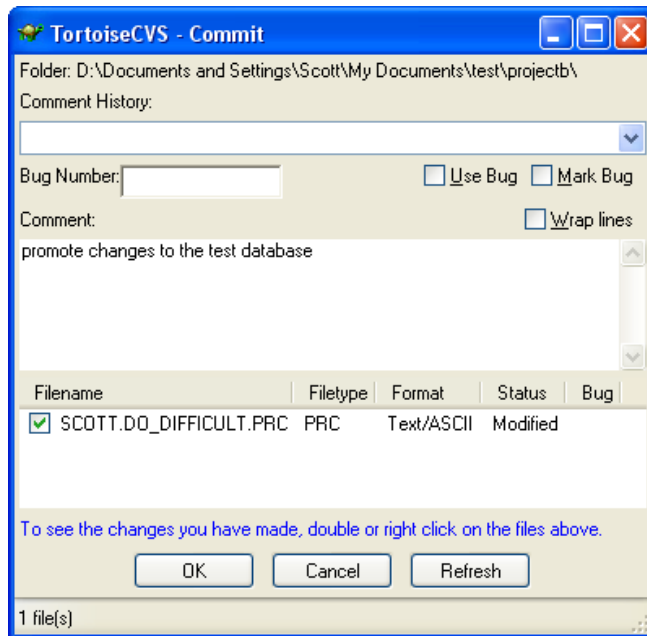
The Promote dialog allows authorised users to select which files or DB objects to promote. Promoting a single item or a large number of items achieved by easily using the graphical promote dialog:



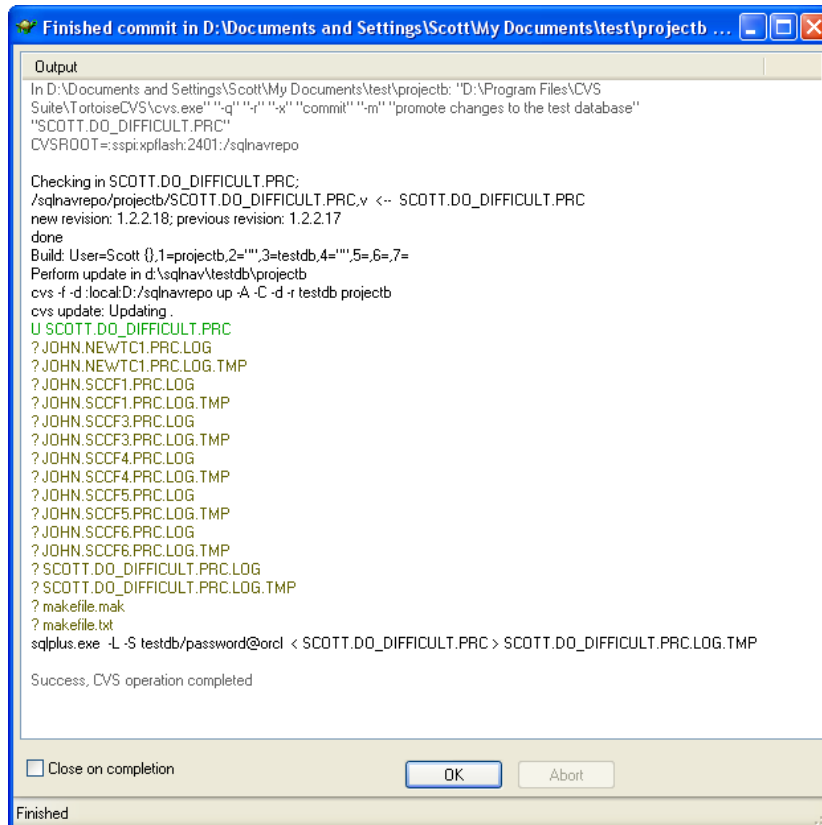
When the promotion is complete the file or DB object is highlighted and ready for commit:



The authorised person is prompted for a comment for the promotion:



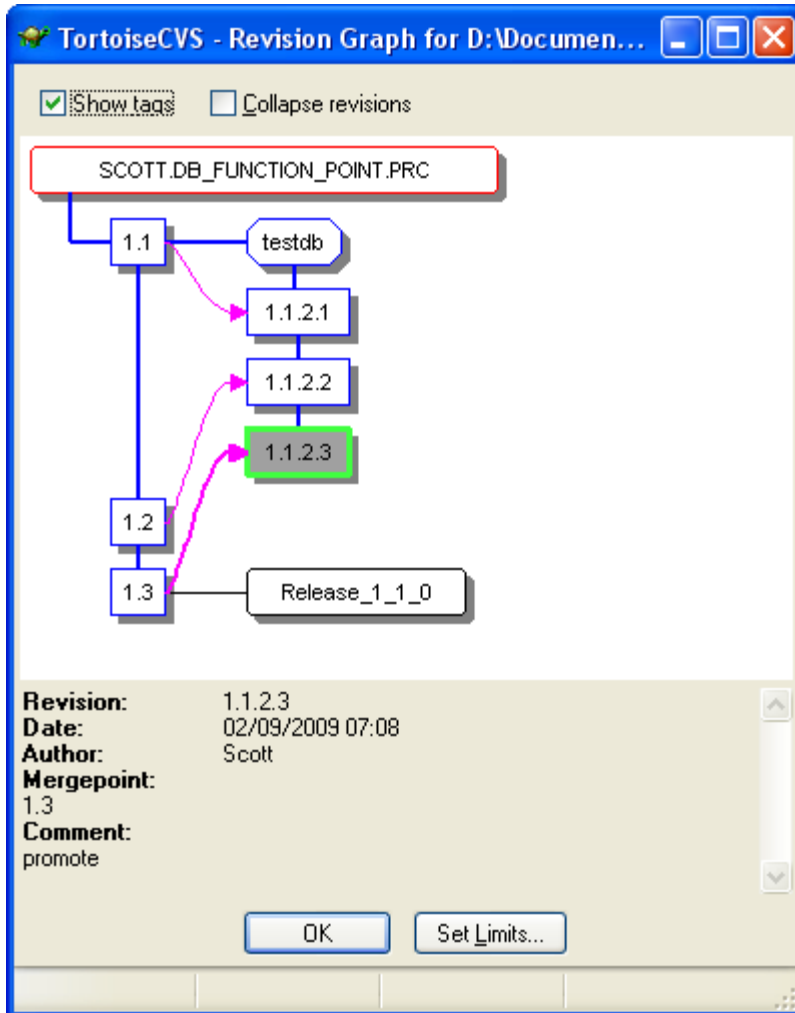
When promote is committed then it is automatically applied to the correct database. From the commit dialog window you can see the *sqlplus* command that has executed the stored procedure script (see below just above the line *Success, CVS operation completed*):



The *build_make.bat* script controls the promotion events (see appendix).

View promotion history

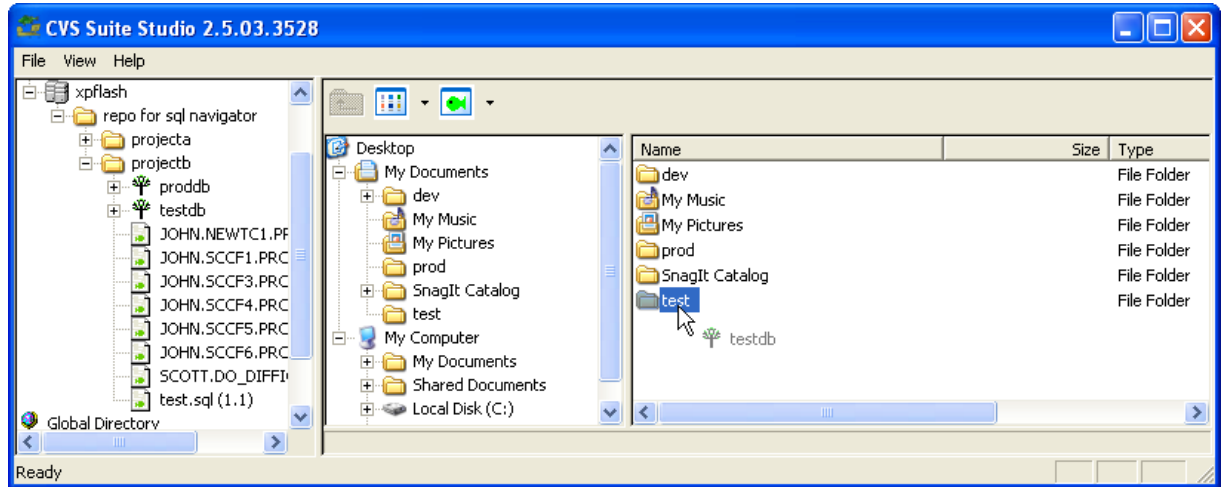
The history of promotion is represented in the revision graph:



Rollback a change to Test or Prod

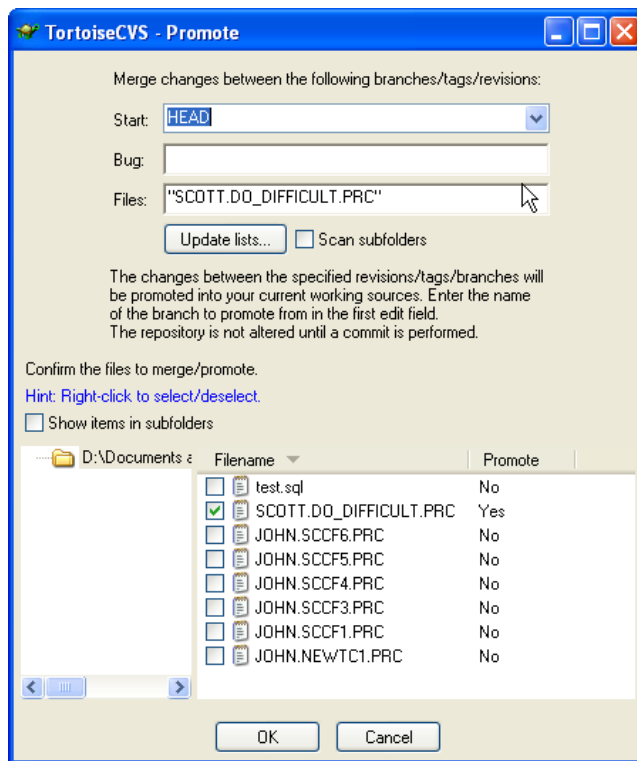
Undoing a change or a series of changes is a variation of a promote – be sure you have already read the previous section on promote.

An authorised individual would be able to create a folder to perform the promote by dragging a CVS Suite Promotion level from the server to a folder on their PC:



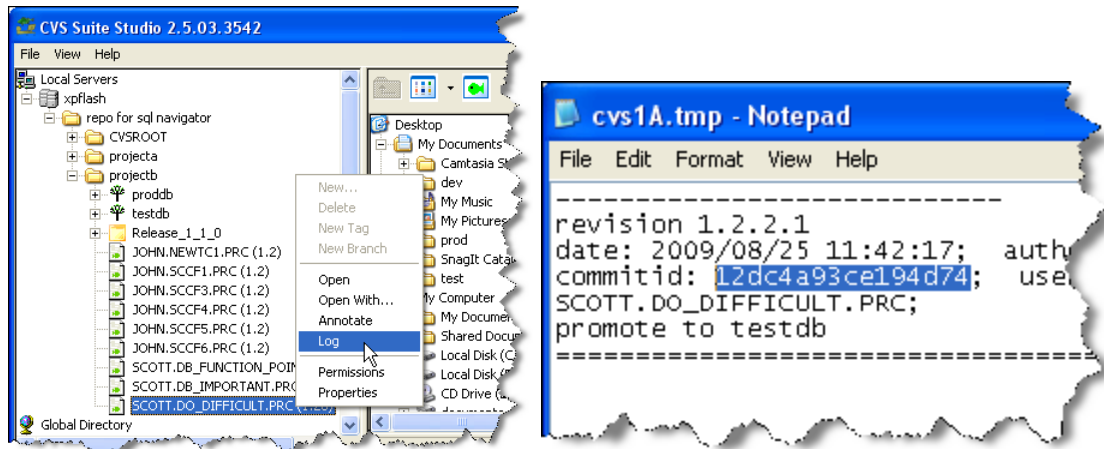
Promoting a previous release or version

The simplest way to roll back a change is to re-promote the previously promoted item. Choose a project or individual files/objects to promote, instead of selecting *HEAD* choose a previous release:

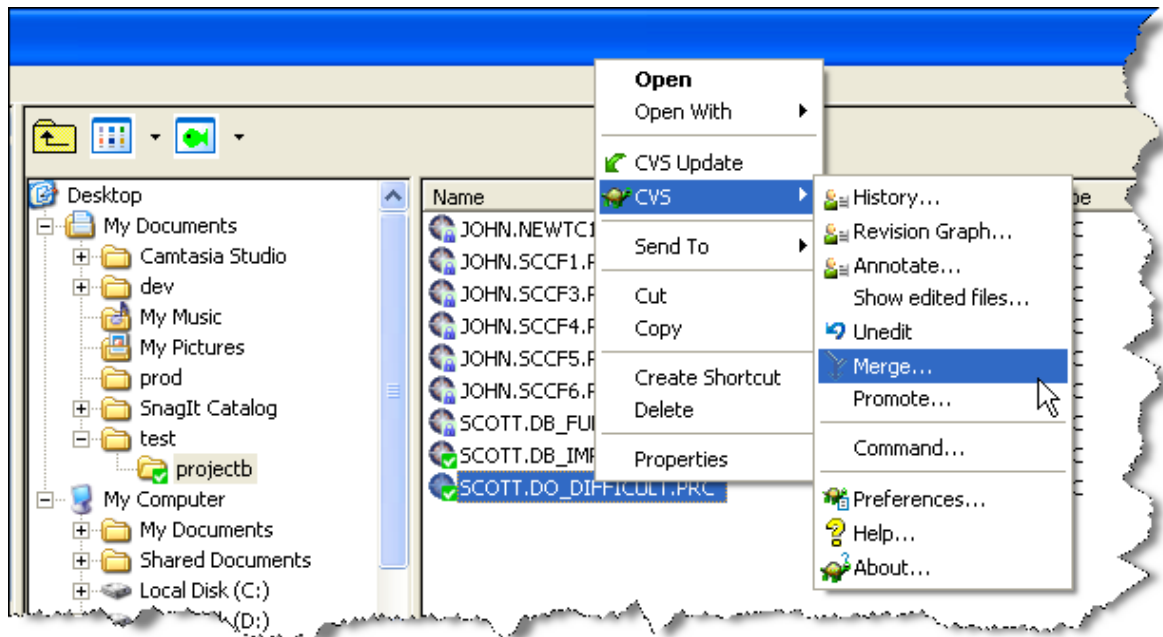


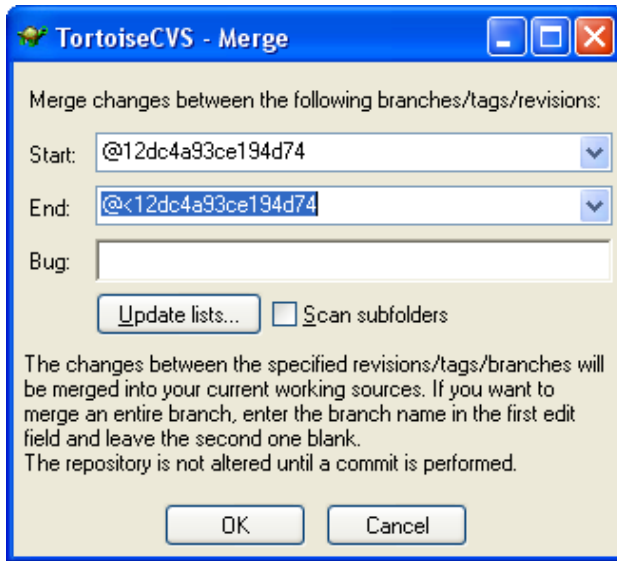
Rolling back any set of changes

Each promote is given a unique commit identifier by the CVS Suite Server.

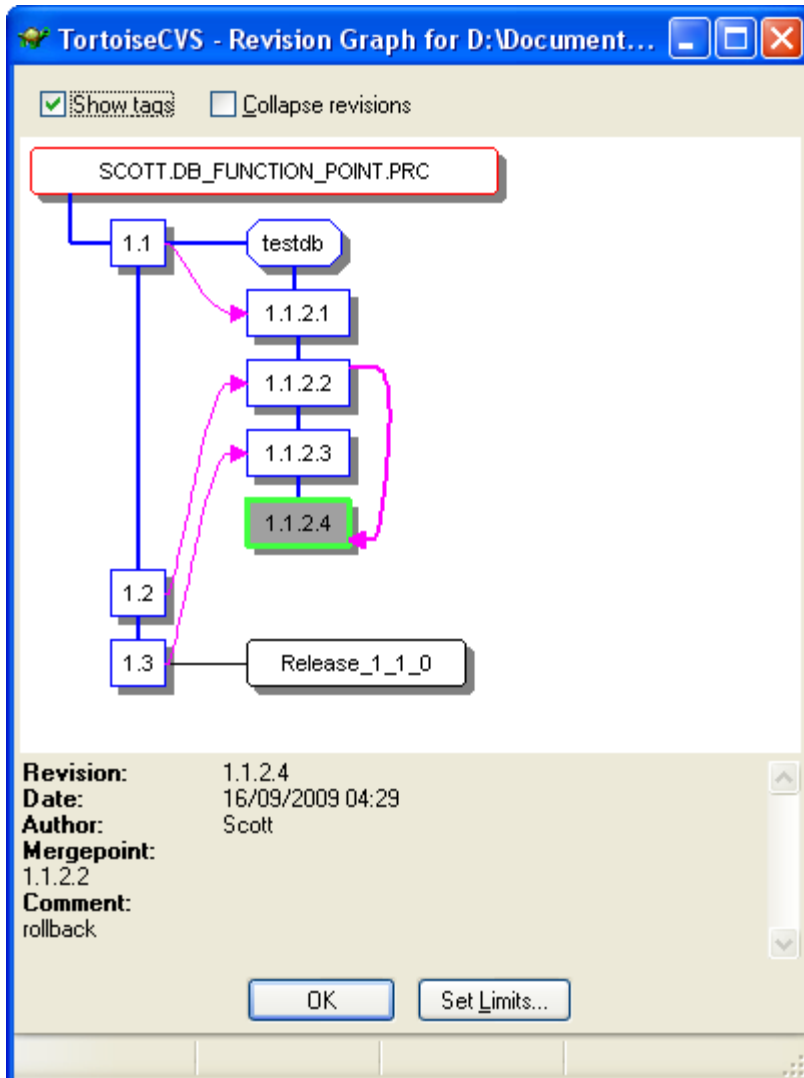


You can use this identifier to undo any set of changes:





The result is a rollback to the previous revision – it rolls back the changeset that was named:



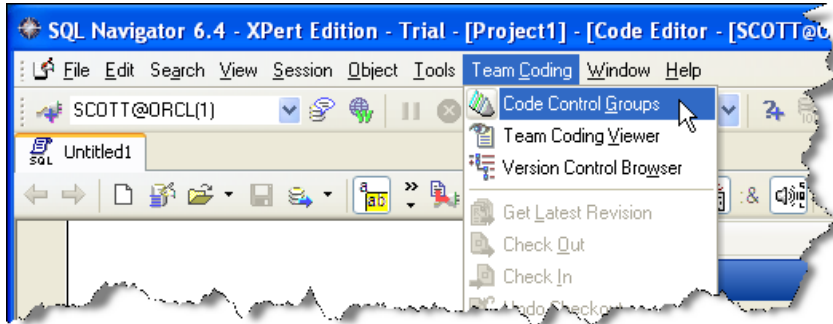
Import existing Database Objects into Version Control

The previous examples have all been based on a NEW object. You may have already been using Oracle and SQL Navigator for months or years and have many objects to bring into version control. This is simple to do via the code control groups screen.

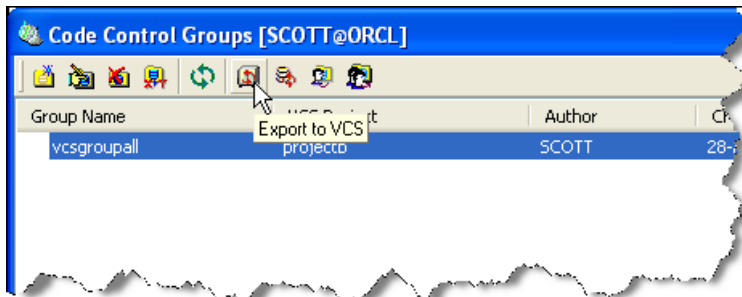
Adding missing objects to Version Control

Use SQL Navigator Control Groups to add missing objects to version control.

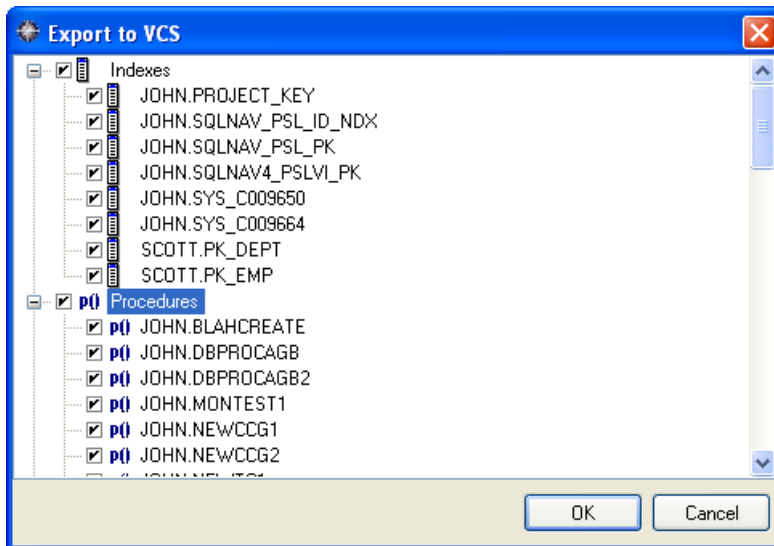
Step 1: use the Team Coding menu to start Code Control Groups



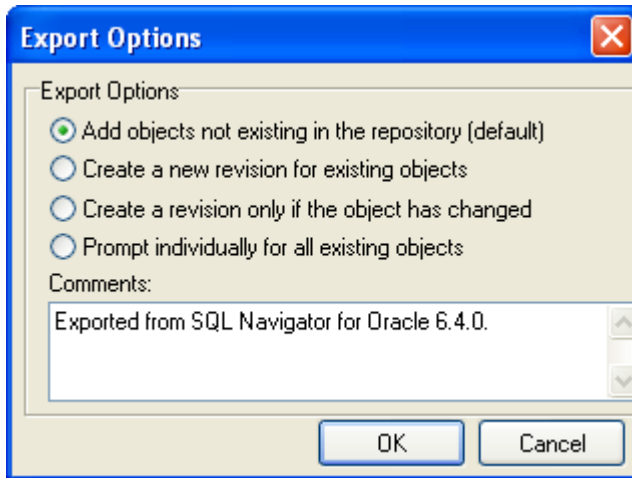
Step 2: Use the Export to VCS option on the Code Control Groups toolbar



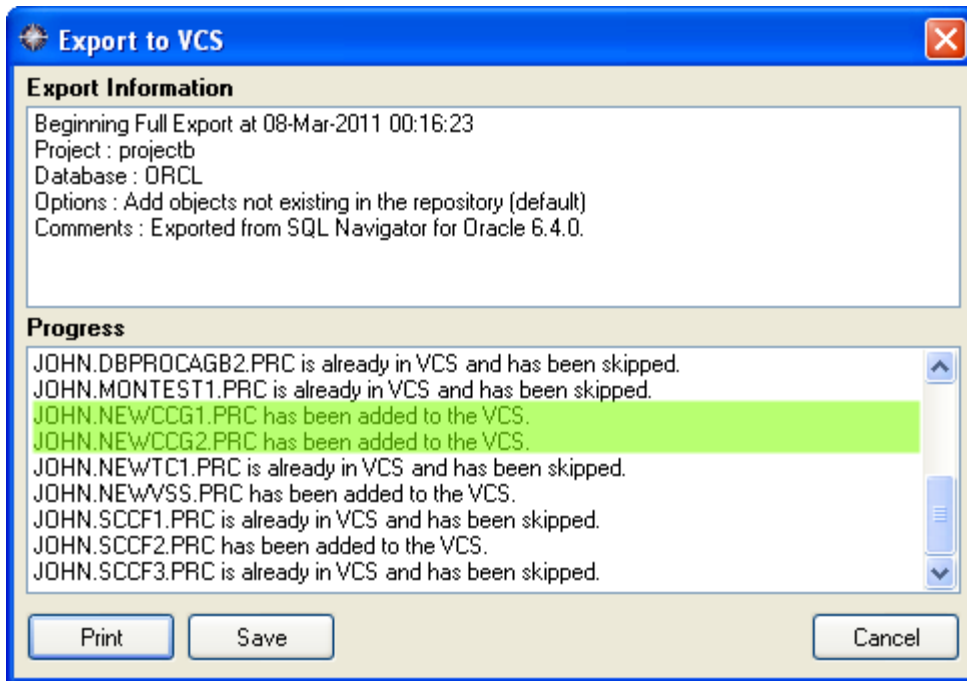
Step 3: select the objects that you wish to import



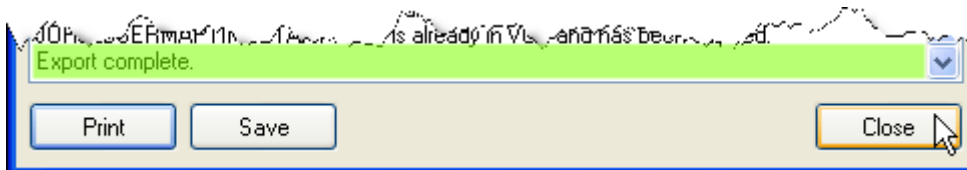
Step 4: Select the export option: Add objects not existing in the repository



Step 5: Objects already in VCS are skipped and missing objects are added to the VCS



Step 6: When the export is complete press the Close button



Begin working on Release 2 whilst maintaining Release 1

You can continue working on release 1 whilst starting work on release 2 by creating a branch. Creating a branch is an administrative task and is not performed by regular staff for minor changes.

Once the administrator has created the branch and separated working directories and databases for development of *release1* and *release2* then patches can be easily merged between them, ie: if a customer reports a bug in *release1* that must be fixed urgently, then once it is fixed in *release1* and tested and promoted to production then the identical patch can be merged into the stream for *release2*.

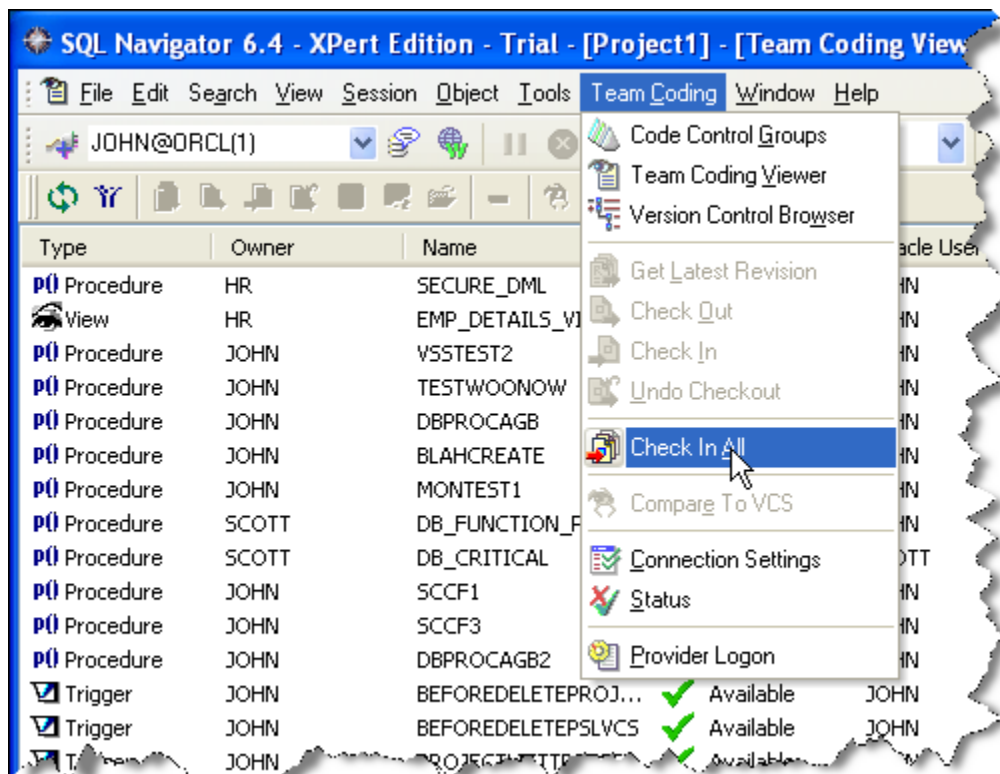
When to create branches

Typically the administrative tasks for creating the branch are done well before *release1* is released to a production environment. This is because as *release1* gets closer and closer to production some resources are no longer required and these are diverted to begin work on release2.

How to begin work on release2 (branching)

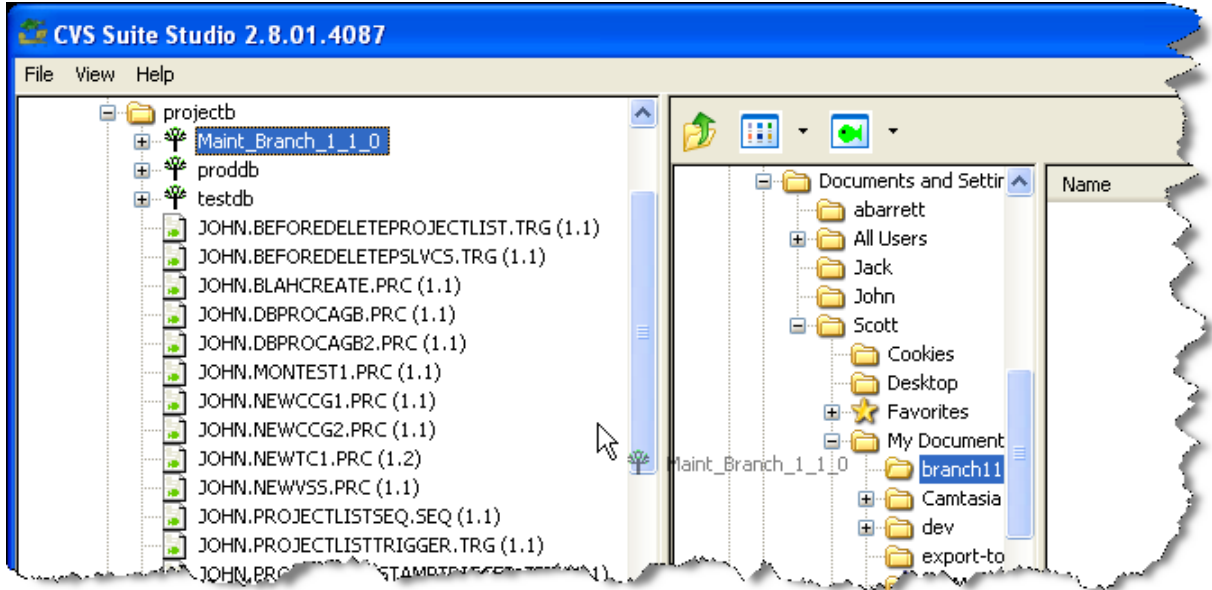
Step 1: Ensure all current work on Trunk is checked in

Before creating a branch ensure that all work on the *Trunk* is completed and checked in:



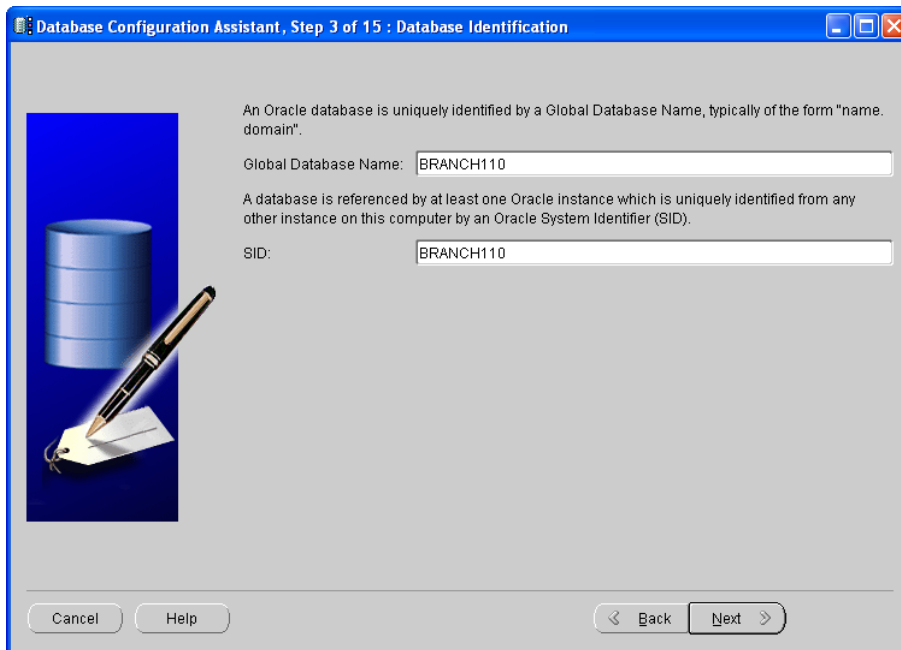
Step 4: Create a new Workspace (Sandbox) for the branch

Create a directory for the new workspace (sandbox) and checkout the code into the created directory by dragging the branch name from the left pane in CVS Suite Studio to the Right pane:



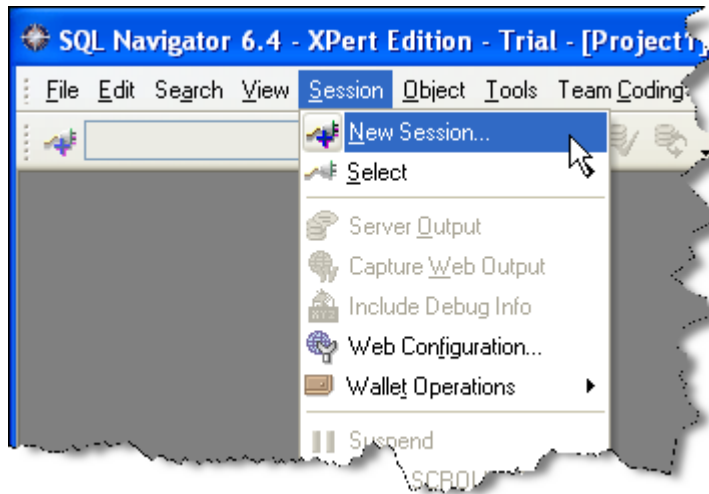
Step 5: Create a new Database for the new Branch

Each branch must be in a unique location – it does not need to be a separate database, however that is the simplest option and the one we will demonstrate here. See the section *What is the Repository and the Workspace* above and particularly the headings *Working Copy Folder versus a Working Copy Database* and *Moving Objects from One database to another Database (eg: Dev to Test)* for more information about this. Create a new database for the new branch:



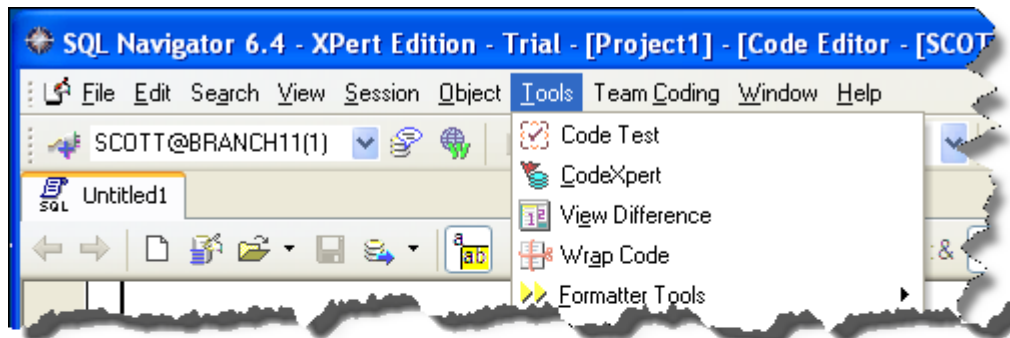
Step 6: Open a New Session in SQL Navigator

Open a new Session in SQL Navigator for the new database:



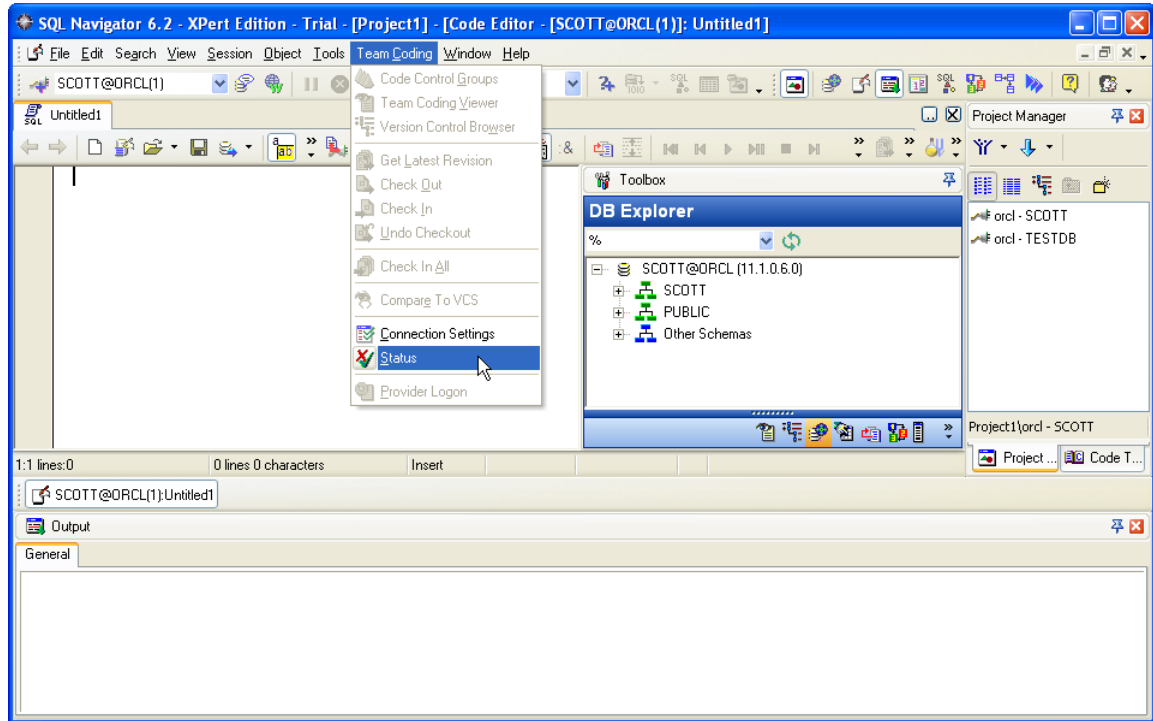
Step 7: Use the Server Side Installation Wizard

This document cannot include all of the steps necessary to configure SQL Navigator for use with a database; this guide is intended to be used by people already familiar with SQL Navigator including any administrative functions. Please read the book that Quest supply with the product: SQL Navigator for ORACLE User's Guide section Version Control and Team Coding and in particular the sub-section Enabling Team Coding in the Oracle Instance.:

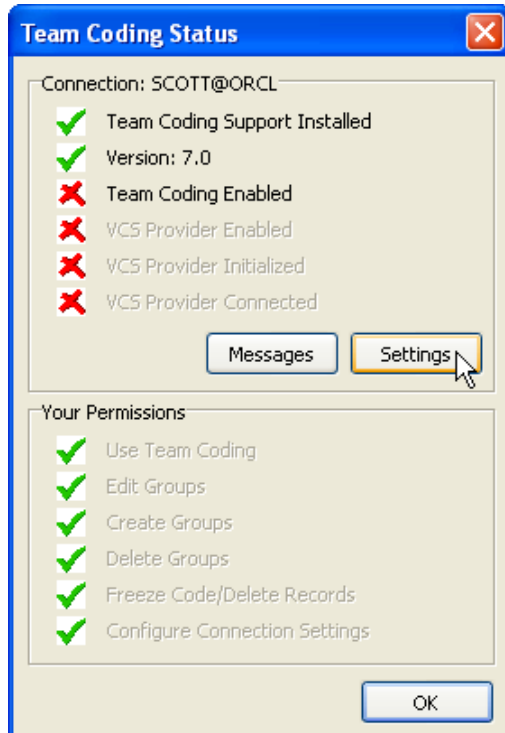


Step 8: Activate the Team Coding connection to the branch

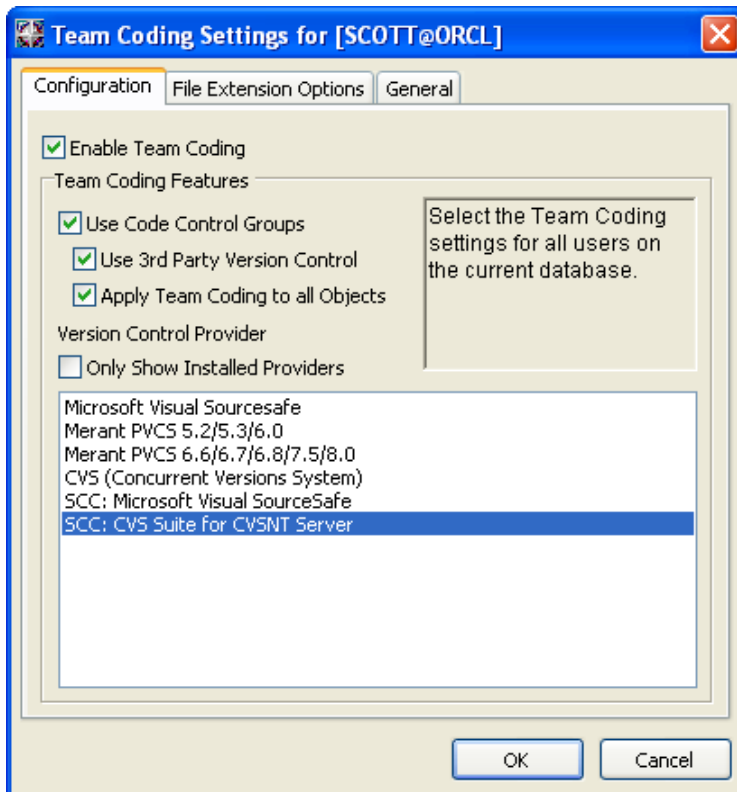
Activate the Team Coding connection between SQL Navigator and CVS Suite for the new database and branch:



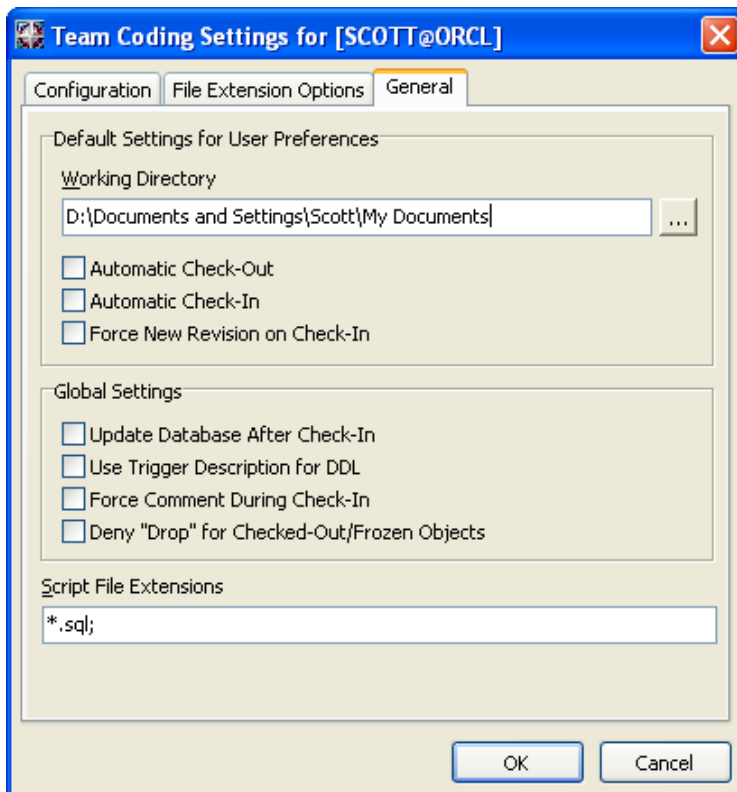
The status window shows that the integration is currently disabled:



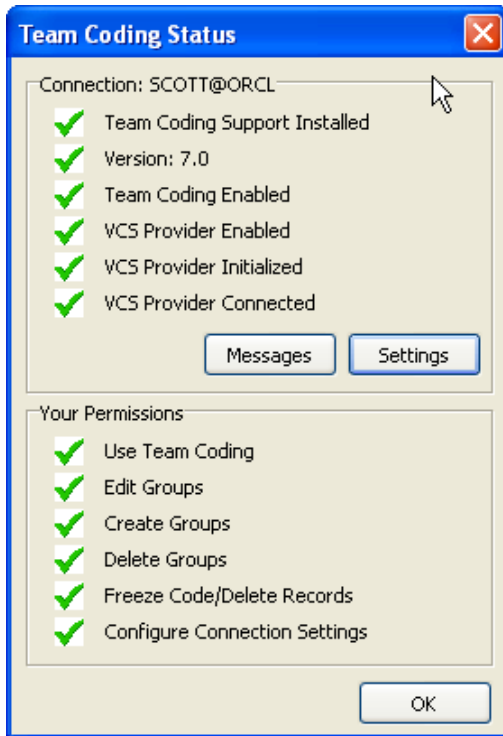
Enable the integration and choose the *CVS Suite for CVSNT Server* SCC provider



In the Team Coding Settings set the Working Directory:

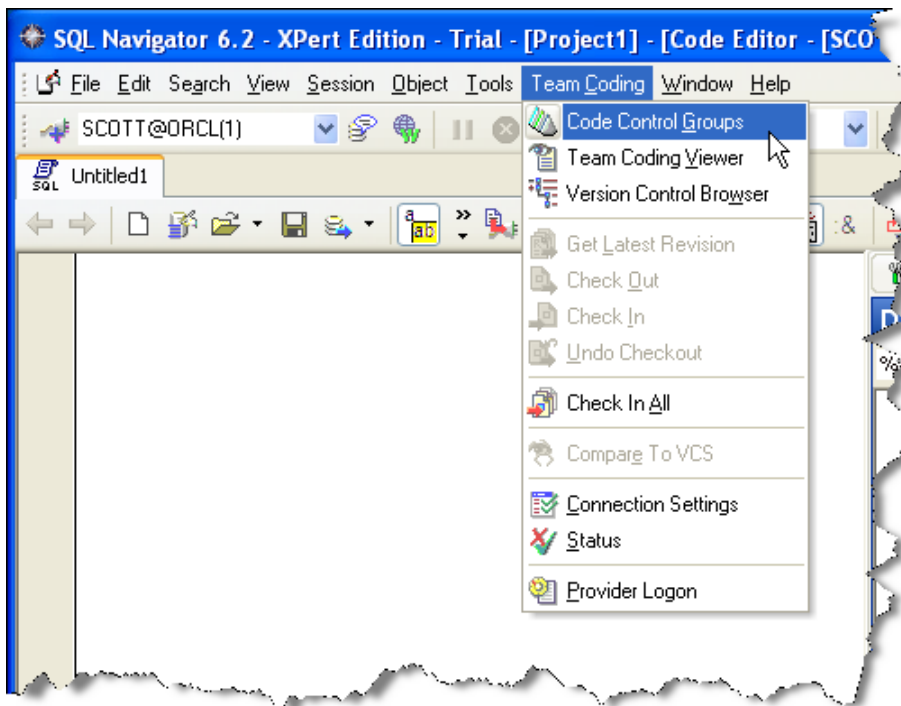


The Team Coding Status is now OK:

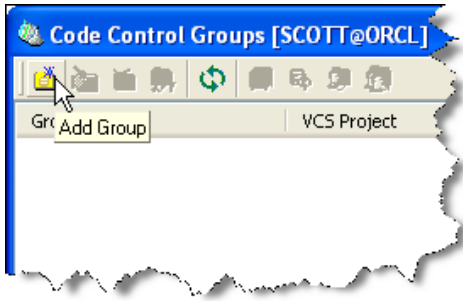


Step 9: Create Code Control Groups for the new database/branch

SQL Navigator requires at least one Code Control Group set up to associate your Database Objects on the branch with a CVS Suite Branch Module and Folder based Workspace:



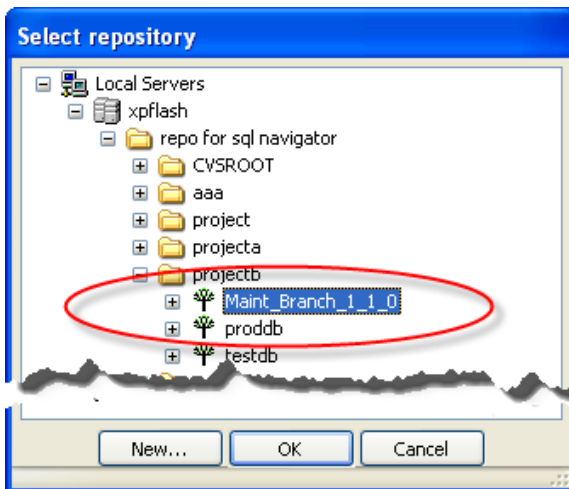
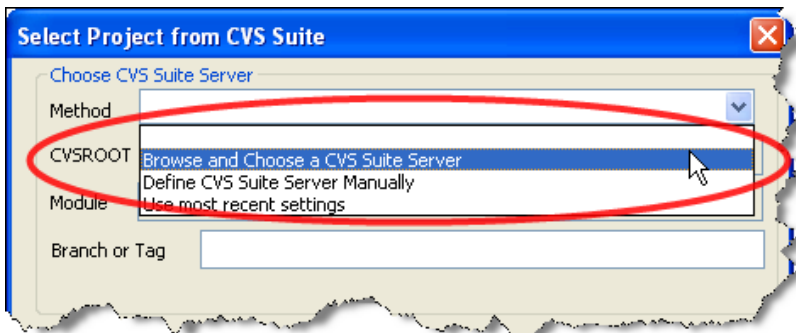
Step 9b: Add the Code Control Group



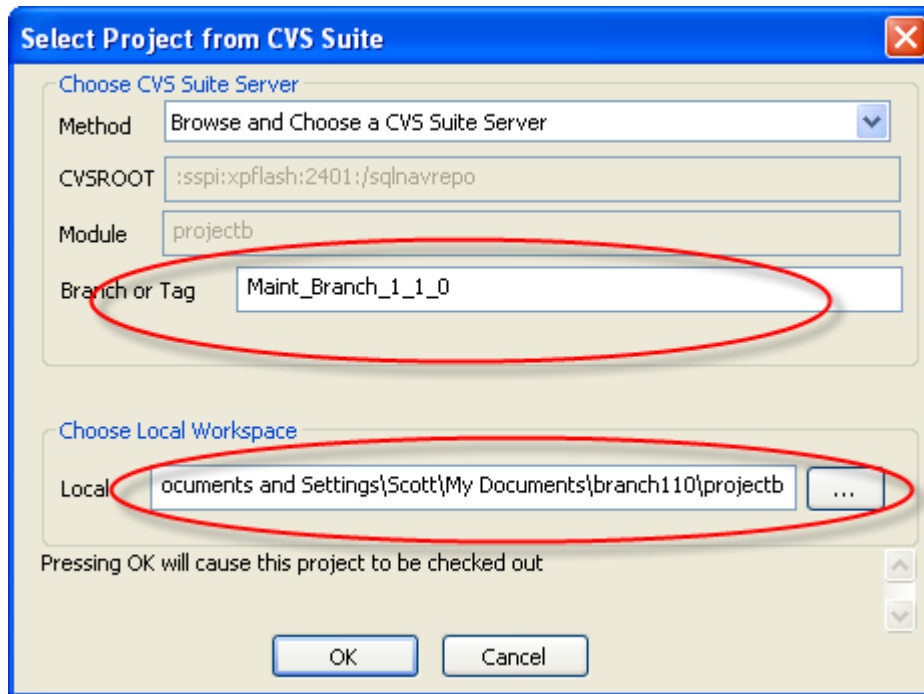
Step 9c: In the New Group window set a Group Name and browse for a project



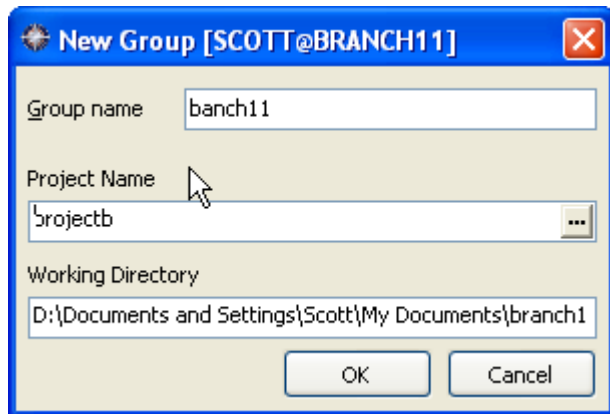
Step 9d: Browse for a project and the local workspace folder / sandbox



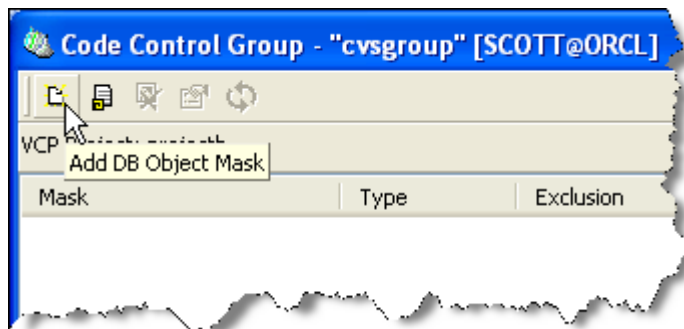
Step 9e: Ensure local path is set to the Workspace Location (Sandbox) created earlier



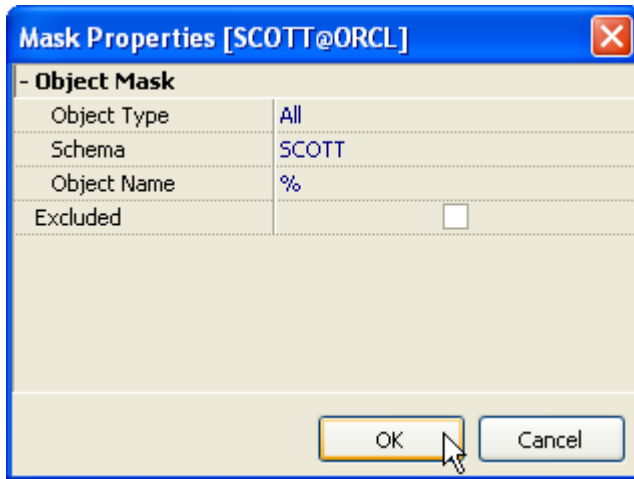
Step 9f: Complete the New Group dialog



Step 9g: On the Code Control Group dialog Add DB Object Mask

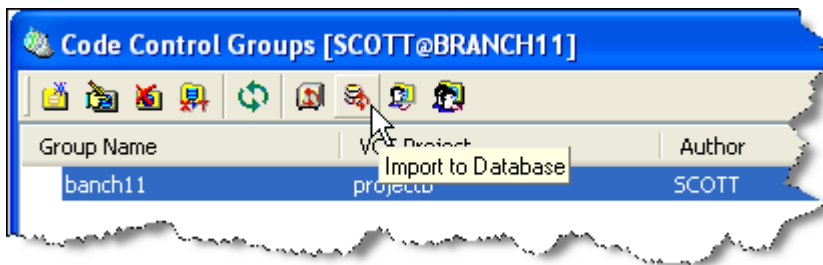


Step 9h: Use the default Mask Properties



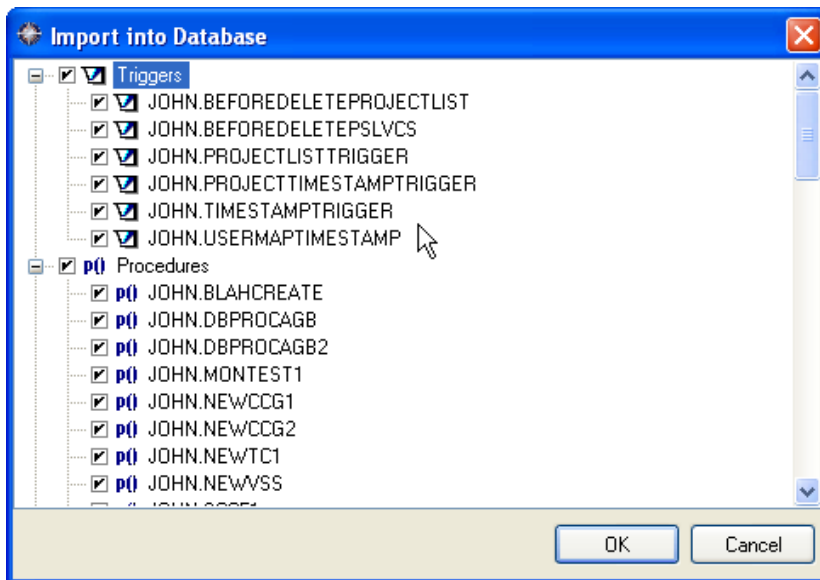
Step 10: Use the Import to Database Code Control Groups Option

On the *Code Control Groups* dialog use the *Import To Database* toolbar button to begin the process of populating the new Branch database with the branch code:



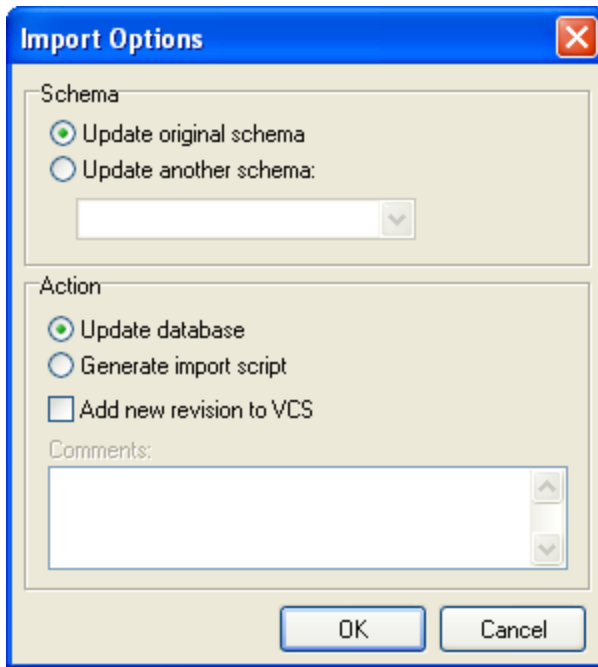
Step 11: Import to Database selection

Use the *Import to Database* dialog to select which objects to import from the branch code:

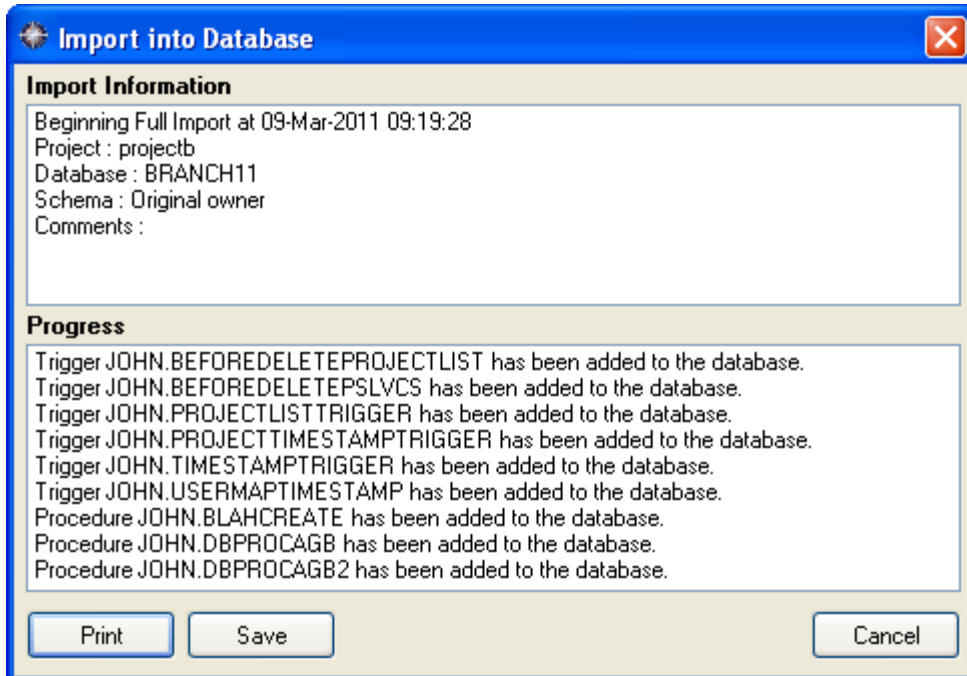


Step 12: Use the Import Options

On the *Import Options* dialog use the default selection to import the code from the branch to the branch database:



The import progress is displayed in the dialog:



When complete press the close button.

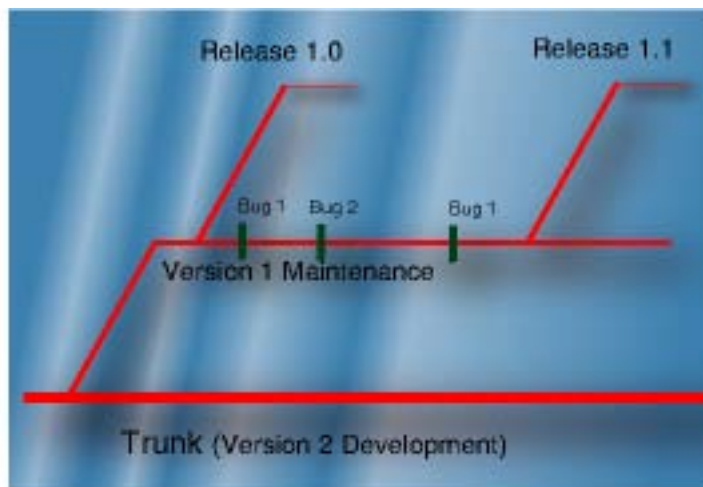
Move patch from release 1 to release 2

When the administrator has created branches and separate working directories and databases for development of *release1* and *release2* then patches can be easily merged between them, ie: if a customer reports a bug in *release1* that must be fixed urgently, then once it is fixed in *release1* and tested and promoted to production then the identical patch can be merged into the stream for *release2*.

Each branch must be in a unique location – it does not need to be a separate database; however that is the simplest option and the one we will demonstrate here. See the section *What is the Repository and the Workspace* above and particularly the headings *Working Copy Folder versus a Working Copy Database* and *Moving Objects from One database to another Database* (eg: *Dev to Test*) for more information about this

Release 1 and Release 2 – Branch and Trunk

See the sections *What are Branches*, *Magic Branches and Vendor Branches*, and *Patch management – getting fixes to customers* above for more information on how to organise branches. In this example Release 1 is maintained on the branch and the Trunk has the development for the next major release (Release 2).



Working with Patching and Merging

You can merge any or all of the following from a branch to the trunk:

- Specific changes (atomic commits or user defined change sets),
- One or more files (all changes)
- Specific changes to one or more files

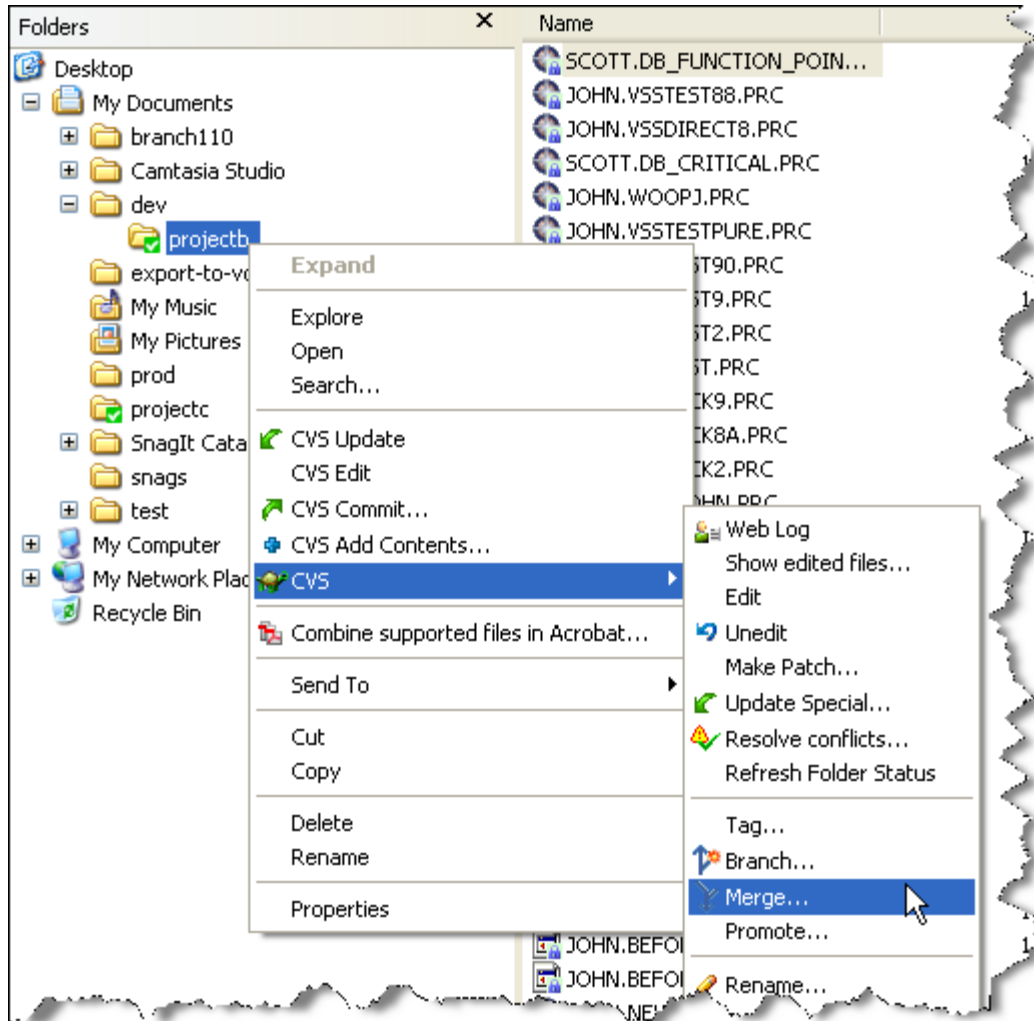
Each change committed to the CVS Suite Server Repository automatically receives a change set id (atomic commit id) and you can optionally assign a user defined change set (bug id) to one or more atomic change sets. To facilitate merging changes from Release 1 back to the main release 2 it is good to specific a user defined change set with each change (bug id).

Applying patch to release2

To apply a patch (eg: a bug fix) from the version 1 maintenance branch to the trunk release 2 development you use the release 2 workspace.

Step 1: Use the CVS->Merge menu

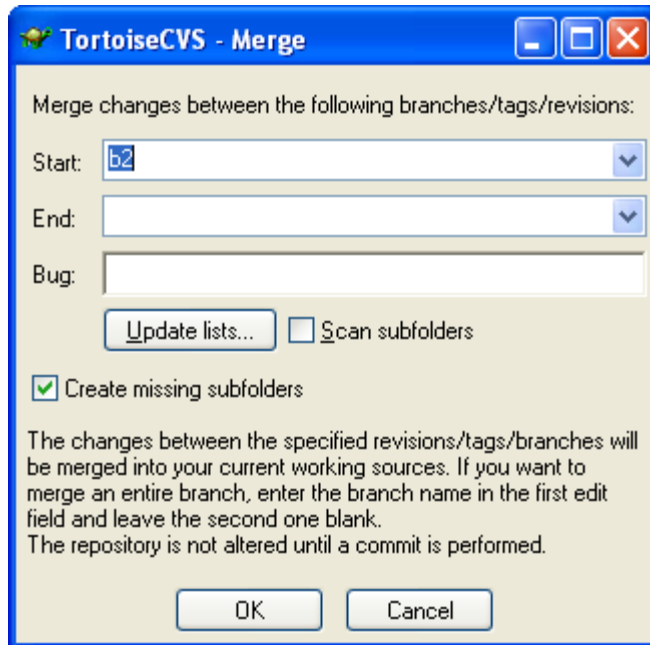
Use the right click menu *CVS ->Merge* in Windows Explorer with the release 2 workspace (sandbox):



Step 2: Enter the name of the maintenance branch

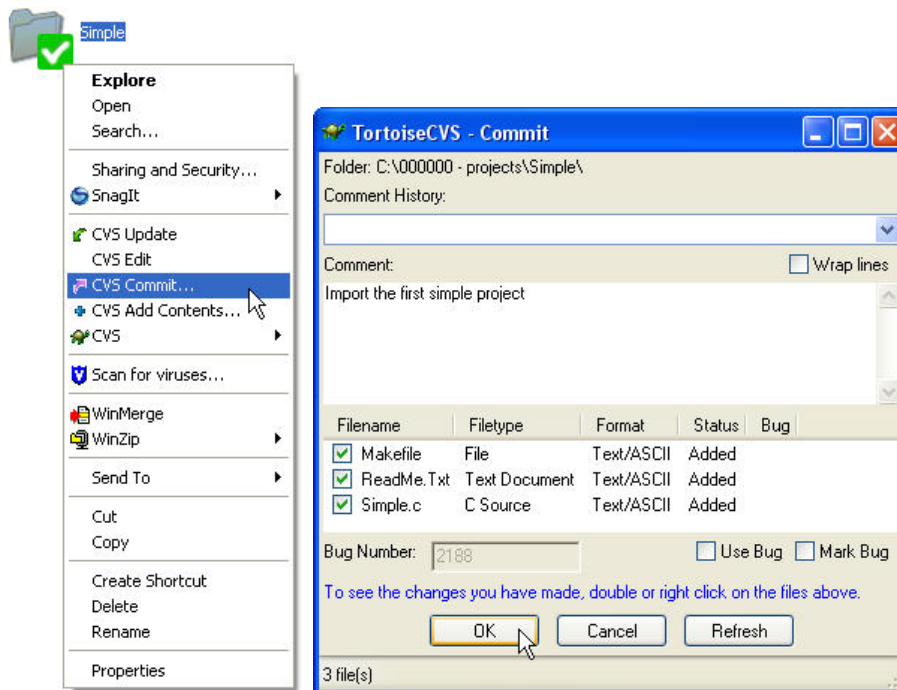
Enter the name of the branch you are merging from in the Merge dialog, eg: *branch1-1-0*.

To merge a specific change set enter the change set number (bug id) in the field *Bug*:



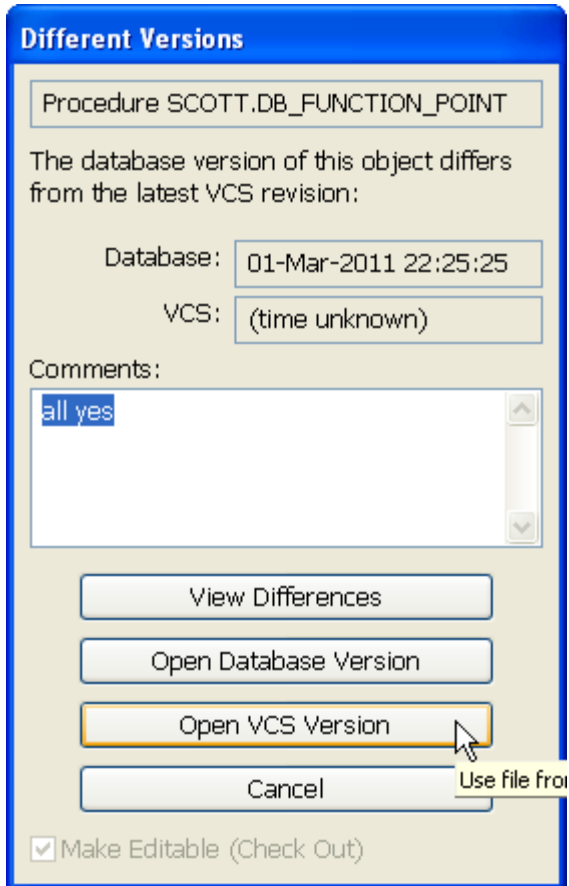
Step 3: Commit the Merge

Use the *CVS Commit* right click menu in Windows explorer to commit the change to the server repository:



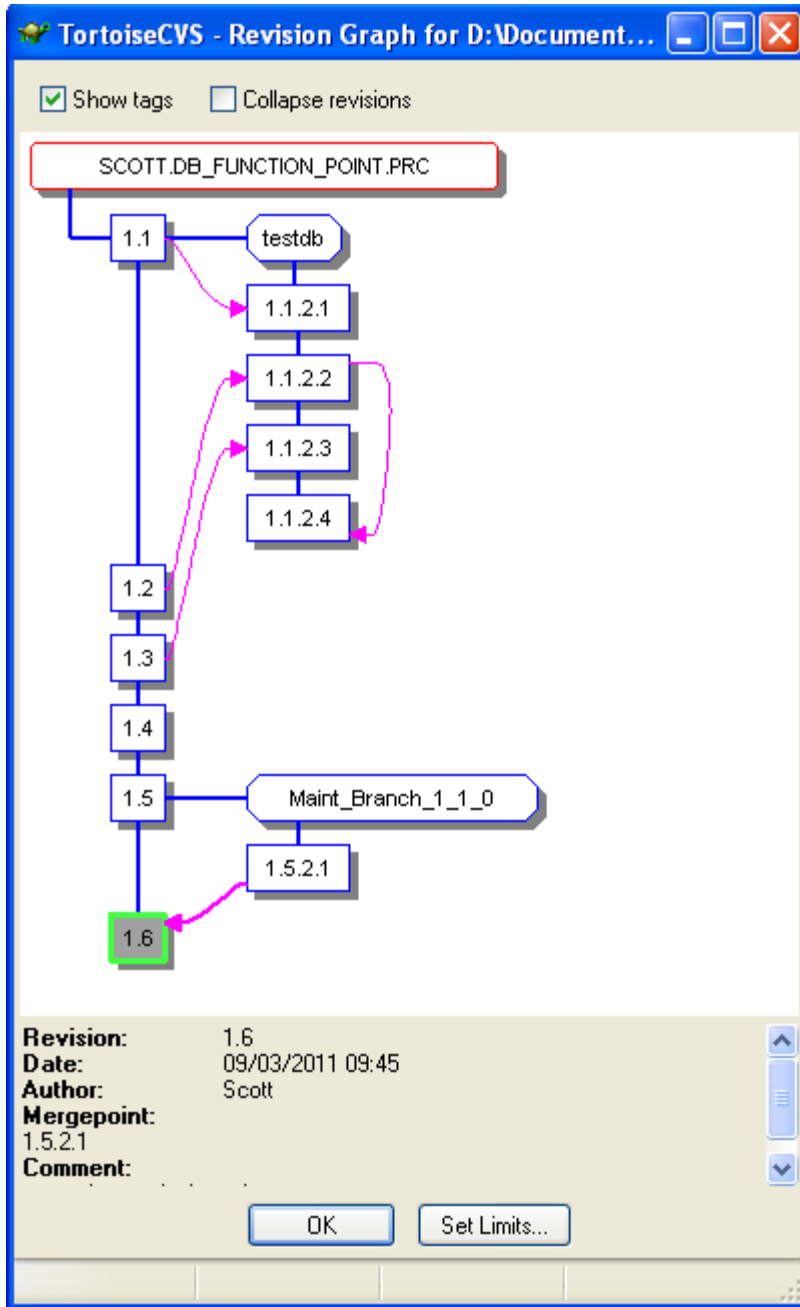
Step 4: Reload SQL Navigator from VCS

Use SQL Navigator to reload the database from the VCS. You can do this in bulk (see *How to begin work on release2 (branching)*) and in particular *Step 10: Use the Import to Database Code Control Groups Option* through to *Step 12*) or you can *Check Out* a single object in SQL Navigator and you will be automatically prompted:



Viewing the Merge History

You can use the *CVS->Revision Graph* right click menu in Windows Explorer to view the merge history, eg:

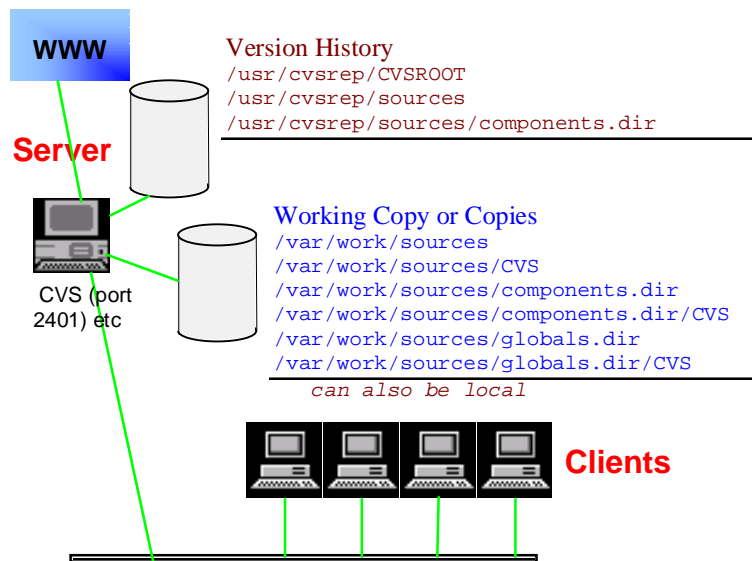


CVS Architecture

CVS Suite Server and Client are designed to work together and in conjunction with Configuration Management processes can improve developer productivity. For a more detailed explanation refer to the documentation that accompanies the software: *All About CVS*.

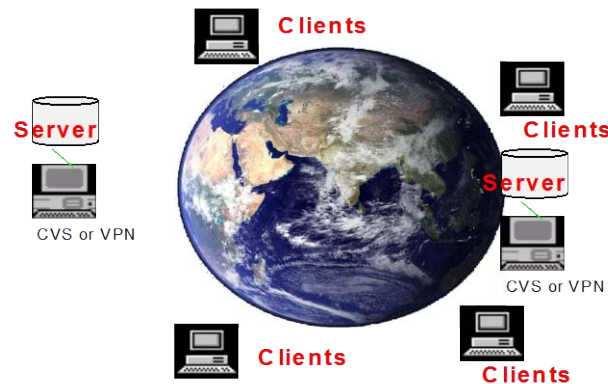
Client / Server Architecture

When using CVS to version control your documents and other objects you always will use the client program (with or without a GUI) with files in a *sandbox*. The CVS Server will always operate on the repository files.



Multi Site, Repository Replication and WAN Performance

CVS Suite is designed to work well when users are distributed worldwide:



Support of CVS Suite Server and your Technical Account Manager

Ensuring the reliability, resilience and integrity of your repository are important when implementing a SCM versioning server. The people who wrote the server control software are the people who can ensure that. If you are unhappy with the performance or reliability of an existing CVS or CVSNT server then the first step is to ensure that you have support from the people who wrote it, and speak to your Technical Account Manager regarding the specific problems you are experiencing – March Hare Software write CVSNT and can assist with resolving problems through CVS Professional Support Services and our QA tested software solution CVS Suite.

Multi Site Solutions

March Hare Software can assist with the following Multi Site Solutions – they are all natively supported in the CVSNT Server which is the basis of CVS Suite – no 3rd party software is required:

- WAN Optimised Workspaces and Procedures
- Multiple Repositories
- Large Dual Site Development - Repository Replication Cache
- Read Only Mirrors - Repository Replication Disaster Recovery or Ownership
- Multi Site Parallel Development
- CVS Server Clustering

Contact March Hare Software for more information.

Appendix

build_make.bat

```

@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 runsql_projectb
IF %1-==projecta~ GOTO runsql_projecta
@echo "This sample makefile only makes projecta and projectb on testdb or proddb"
exit /b 9
:runsql_projectb
IF %3-==testdb~ GOTO managesql
IF %3-==proddb~ GOTO managesql
@echo "This sample makefile only makes projectb on promotion levels testdb or proddb"
exit /b 8
:runsql_projecta
IF %3-==testdb~ GOTO managesql
IF %3-==proddb~ GOTO managesql
@echo "This sample makefile only makes projecta on promotion levels testdb or proddb"
exit /b 8

:managesql
d:
@if exist d:\sqlnav goto sqlnavok
mkdir d:\sqlnav
chdir d:\sqlnav
@if exist d:\sqlnav goto sqlnavok
@echo "Could not create directory for SQL Navigator projects"
exit /b 7

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

:sqldbok
@if exist d:\sqlnav\%3\%1 goto sqlprojupd
cvs -f -d :local:%REAL_CVSROOT% co -r %3 %1
@if NOT ERRORLEVEL 0 goto sqlprojcoerr
@if exist d:\sqlnav\%3\%1\CVS goto sqlprojok
:sqlprojcoerr
@echo "Could not checkout %1 promotion level %3 "
exit /b 6

:sqlprojupd
@if not exist d:\sqlnav\%3\%1\CVS goto sqlprojerr
@if not exist d:\sqlnav\%3\%1\CVS\Root goto sqlprojerr
@if not exist d:\sqlnav\%3\%1\CVS\Entries goto sqlprojerr
cd d:\sqlnav\%3\%1
@echo Perform update in d:\sqlnav\%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 sqlprojok
:sqlprojerr
@echo "Could not update %1 promotion level %3 "
exit /b 5

```

```
:sqlprojok
cd d:\sqlnav\%3\%1
@if exist d:\sqlnav\%3\%1\makefile.mak goto sqlmakeok
@echo # This makefile requires a make utility, eg: GNU Make from: >> makefile.mak
@echo # http://unxutils.sourceforge.net/ >> makefile.mak
@echo # >> makefile.mak
@echo # Note: this uses the DOS find command not the grep command >> makefile.mak
@echo # to minimise dependance on 3rd party tools >> makefile.mak
@echo+ >> makefile.mak
@echo SQLPLUS=sqlplus.exe >> makefile.mak
@echo SQLFLAGS=-L -S %3/password@orcl >> makefile.mak
@echo+ >> makefile.mak
@echo SOURCES=$(wildcard ^*.PRC) >> makefile.mak
@echo TARGETS=$(patsubst %%.PRC,%%.PRC.LOG,$(wildcard *.PRC)) >> makefile.mak
@echo+ >> makefile.mak
@echo+ >> makefile.mak
@echo .PHONY : all clean >> makefile.mak
@echo+ >> makefile.mak
@echo all: $(TARGETS) >> makefile.mak
@echo+ >> makefile.mak
@echo clean: >> makefile.mak
@echo @-erase /f /q *.TMP >> makefile.mak
@echo @-erase /f /q *.LOG >> makefile.mak
@echo+ >> makefile.mak
@echo %%.PRC.LOG: %%.PRC makefile.mak >> makefile.mak
@echo $(SQLPLUS) $(SQLFLAGS) ^< $^< ^> $@.TMP >> makefile.mak
@echo @find "Procedure created." $@.TMP ^> $@.RESULT >> makefile.mak
@echo @-erase $@ 2^> nul >> makefile.mak
@echo @rename $@.RESULT $@ >> makefile.mak
@echo+ >> makefile.mak
@if exist d:\sqlnav\%3\%1\makefile.mak goto sqlmakeok
@echo "Could not find or create a makefile for %1 promotion level %3"
exit /b 4

:sqlmakeok
make -f makefile.mak
exit /b %errorlevel%
```

build make.sh

```

#!/bin/bash
#shopt -s igncr; #
REAL_CVSROOT=/i01_02/elc_cvsrepo

echo "Build: User=$USER ($Username),1=$1,2=$2,3=$3,4=$4,5=$5,6=$6,7=$7)"
if [ "x$1" = "xproja" ]; then
    # do stuff for project a
    echo "do stuff for project a"
fi

if [ "x$1" = "xprojb" ]; then
    # do stuff for project b
    if [ "x$3" = "xdev" ]; then
        # do stuff for project b / dev promotion level
        echo " do stuff for project b / dev promotion level"
    fi
    if [ "x$3" = "xstage" ]; then
        # do stuff for project b / stage promotion level
        if [ ! -d /home/cvsadmin/sqlnav ]; then
            mkdir /home/cvsadmin/sqlnav
        fi
        cd /home/cvsadmin/sqlnav
        if [ ! -d /home/cvsadmin/sqlnav/$3 ]; then
            mkdir /home/cvsadmin/sqlnav/$3
        fi
        cd /home/cvsadmin/sqlnav/$3
        if [ -d /home/cvsadmin/sqlnav/$3/$1 ]; then
            if [ ! -d /home/cvsadmin/sqlnav/$3/$1/CVS ]; then
                echo "/home/cvsadmin/sqlnav/$3/$1/CVS doesn't exist!"
                exit -1
            fi
            if [ ! -f /home/cvsadmin/sqlnav/$3/$1/CVS/Root ]; then
                echo "/home/cvsadmin/sqlnav/$3/$1/CVS/Root doesn't exist!"
                exit -1
            fi
            if [ ! -f /home/cvsadmin/sqlnav/$3/$1/CVS/Entries ]; then
                echo "/home/cvsadmin/sqlnav/$3/$1/CVS/Entries doesn't exist!"
                exit -1
            fi
            cd /home/cvsadmin/sqlnav/$3/$1
            echo "Perform update in /home/cvsadmin/sqlnav/$3/$1"
            echo "cvs -f -d :local:$REAL_CVSROOT up -A -C -d -r $3 $1"
            cvs -f up -A -C -d -r $3
            # really needs some error detection right here!
        else
            # never done a promote before
            cvs -f -d :local:$REAL_CVSROOT co -r $3 $1
            # needs some error checking here...
        fi

        # at this point we now have the promoted code on the server available to act on
        # the following starts a process which will "run" the modified/promoted code
        # against the correct database
        if [ ! -f /home/cvsadmin/sqlnav/$3/$1/Makefile.mak ]; then
            echo "Makefile missing!"
            exit -1
        else
            echo "Do the build now"
            cd /home/cvsadmin/sqlnav/$3/$1/
            export ORACLE_BASE=/i01_01/app/oracle
            export ORACLE_HOME=/i01_01/app/oracle/product/11.2.0/db01
            export PATHX=/usr/kerberos/bin:/usr/local/bin:/bin:/usr/bin:/sbin:/usr/X11R6/bin
            export PATH=$ORACLE_HOME/bin:$ORACLE_HOME/OPatch:$PATHX:$PATH
            make -f Makefile.mak
        fi
    fi
fi

```

Makefile

The *makefile* inserts the objects into the database from the file system, only changed objects are inserted. The samples below must be modified for each branch and each promotion level to have the correct username/password and database name hardcoded.

makefile.mak (for Microsoft Windows)

```
# This makefile requires a make utility, eg: GNU Make from:
#   http://unxutils.sourceforge.net/
#
# Note: this uses the DOS find command not the grep command
#       to minimise dependance on 3rd party tools

SQLPLUS=sqlplus.exe
SQLFLAGS=-L -S testdb/password@orcl

SOURCES=$(wildcard *.PRC)
TARGETS=$(patsubst %.PRC,%.PRC.LOG,$(wildcard *.PRC))

.PHONY : all clean

all: $(TARGETS)

clean:
    @-erase /f /q *.TMP
    @-erase /f /q *.LOG

%.PRC.LOG: %.PRC makefile.mak
    $(SQLPLUS) $(SQLFLAGS) < $< > $@.TMP
    @find "Procedure created." $@.TMP > $@.RESULT
    @-erase $@ 2> nul
    @rename $@.RESULT $@
```

makefile.mak (for Linux)

```
# This makefile required GNU Make

SQLPLUS=/i01_01/app/oracle/product/11.2.0/db01/bin/sqlplus
SQLFLAGS=-L -S elelis/winter10@dna2

SOURCES=$(wildcard *.PRC)
TARGETS=$(patsubst %.PRC,%.PRC.LOG,$(wildcard *.PRC))

.PHONY : all clean

all: $(TARGETS)

clean:
    @-erase *.TMP
    @-erase *.LOG

%.PRC.LOG: %.PRC Makefile.mak
    $(SQLPLUS) $(SQLFLAGS) < $< > $@.TMP
    @grep "Procedure created." $@.TMP > $@.RESULT
    @rm -f $@ 2> /dev/null
    @mv $@.RESULT $@
```