CM+ DEMO TUTORIAL

CM+ Demo Overview

The CM+ demo tutorial lets users become familiar with the capabilities of CM+ as it is configured out-of-the-box. Users wishing to evaluate CM+ customized to their own environment should consult their CM+ Administration team..

The tutorial is role-based, focusing on three typical development roles: project manager, developer and configuration manager. Executing the entire tutorial will give the evaluator a feel for the role interaction and the seamless integration between applications.

Although many of the sections of this guide involve interactive operations, some sections are provided for information purposes only.

The tutorial is centered around a sample program, which helps its users calculate their bonuses. The BonusCalculator is a Windows executable which can be found in the “bin” directory of your STS installation - feel free to try it out. After selecting minimum and maximum bonus parameters using the Options menu, you calculate a bonus by clicking the Calculate button. Click the OK button to receive your bonus. In Release 2 of BonusCalculator, the project manager has identified some required enhancements and has been advised of an emergency problem that prevents users from receiving their bonuses.

CM+ Libraries and Products

A CM+ Library is the entire CM+ Repository along with the set of external customization files which help to give a project it's specific user interface. Each library contains all project data that can be stored in soft form. This includes documentation, source code, test cases, requirements, problem reports, planning data, and so forth. The out of the box configuration of CM+ is meant to hold most Application Lifecycle Management (ALM) data. It may be extended to meet the additional requirements of any product development team.

The CM+ Repository is part of the STS Engine which provides not only a Hybrid Database capability (i.e. relational data, revisioned data, hierarchical data, large objects, etc.), but also a Process Engine, Command and Macro Language, and a GUI Generation capability.

Each CM+ Library can manage one or more Products. Typically, independent development teams have their own CM+ libraries, while products which are administered by a single team or which have some level of data sharing or common planning are grouped into a single library. Views of the library contents can be easily partitioned by product, so that any particular user can see only the products assigned to that user. By default users can see all products but typically work with a single product at a time.

A Product is a container for all of the data associated with a product's application life cycle. When navigating data within the CM+ Library, it is normal to establish a Product context so that the data shown in queries, reports, source trees, to-do lists etc. reflect only the product (and its sub-products) in context.

Each product has a "root node" which denotes the root directory for all product source code. Source code is arranged hierarchically below this root node and typically will be grouped according to the various target deliverables to be generated from the code. There may or may not be specific conventions used for grouping source code within a given product team.

Starting the Demo Server

CM+ is a Client/Server application. The smart client permits a full suite of query capabilities without a running server. However, in order to process changes to a library's data, such as adding a problem report, the CM+ Library Server needs to be running for the project library.

In normal operation, the CM+ Library Server runs as a service or daemon on a server platform. In the tutorial, you will start and stop the server manually. In this way, Neuma will not be adding to the “services” or “Initialization Sequence” running on your Windows or Unix system.

As well, in normal operation, the user will not have write access to the repository and its files. This is accomplished through the use of the CM+ Transaction Server process which collects transactions and sends them to the CM+ Library Server. However, to keep things simple for now, the user has write access to the demo repository.

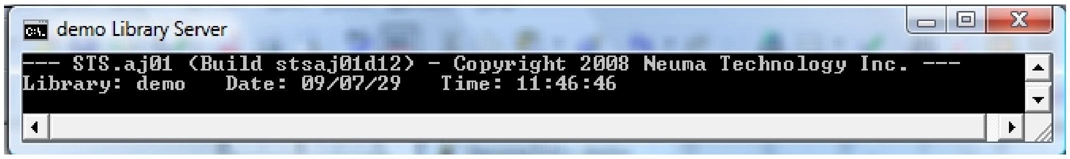
To start up the demo library:

1. **Windows:** Use the **Start** menu to Select **Programs/CM+/CM+ Demo Administrator**
   Unix: sts demo -s -u stsmgr (from command shell)

   PLEASE NOTE: The **"stsmgr"** (Administrator) **Password** should normally be blank.

   A CM+ client session will start with user 'stsmgr' logged in. This user can perform all administrative operations from the standard CM+ client.

2. Under the **Administrator** Menu select **Start Server**. A library server process will start, noting the library name and date and time. On Windows, the process will start in a command window. Minimize, but do not close, this window. (This window would not appear if the server were started as a service.)



```
--- STS.aj01 (Build stsaj01d12) - Copyright 2008 Neuma Technology Inc. ---
Library: demo    Date: 09/07/29    Time: 11:46:46
```

3. You may minimize the stsmgr`s CM+ administrator client session (or close it down) once the server process has started. To stop the library server, first return to the CM+ Administrator (stsmgr) client session, then select **Stop Server** from the **Administrator** menu.

PROJECT MANAGEMENT ROLE

Project Manager Overview

To demonstrate some of the project management capabilities, log into CM+ as the Project Manager (smith).

On Windows, select **CM+ Demo Project Manager** from the **Start/Programs/CM+** menu, or log on as user smith (e.g. run **C:\Neuma\sts\bin\sts demo -g -u smith**).

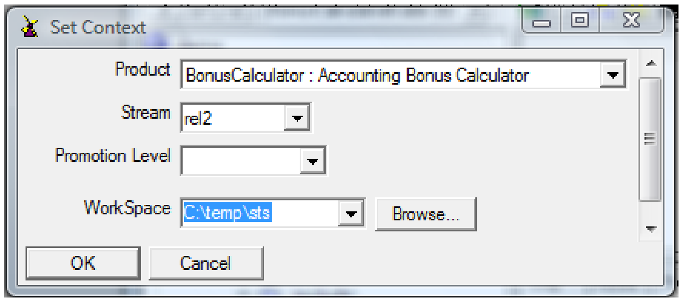
On Unix, specify the "-u smith" after the library name when entering the library: **sts demo -g -u smith** (assumes an sts alias or the CM+ 'bin' directory in your PATH).

Once logged in as user "smith", your menus will include project management functions not available to other roles.

Setting a Development Context

CM+ maps your product road map (i.e. release history and plans) onto tracked **products** and development **streams** (which culminate in one or more releases). Setting a development context enables you to control your view of the repository data, and ensures you see the correct revisions of files and directories. It also helps to ensure that the correct information is captured, by default, on forms, that your reporting and query operations are normally specific to your context, and that pick lists and other information lists have the most suitable information in them.

This demo tutorial assumes we`re working with the **BonusCalculator** product in the **rel2** stream. Set your development context by selecting **Set Context** from the **Context** menu.



Product: BonusCalculator : Accounting Bonus Calculator
Stream: rel2
Promotion Level:
WorkSpace: C:\temp\sts

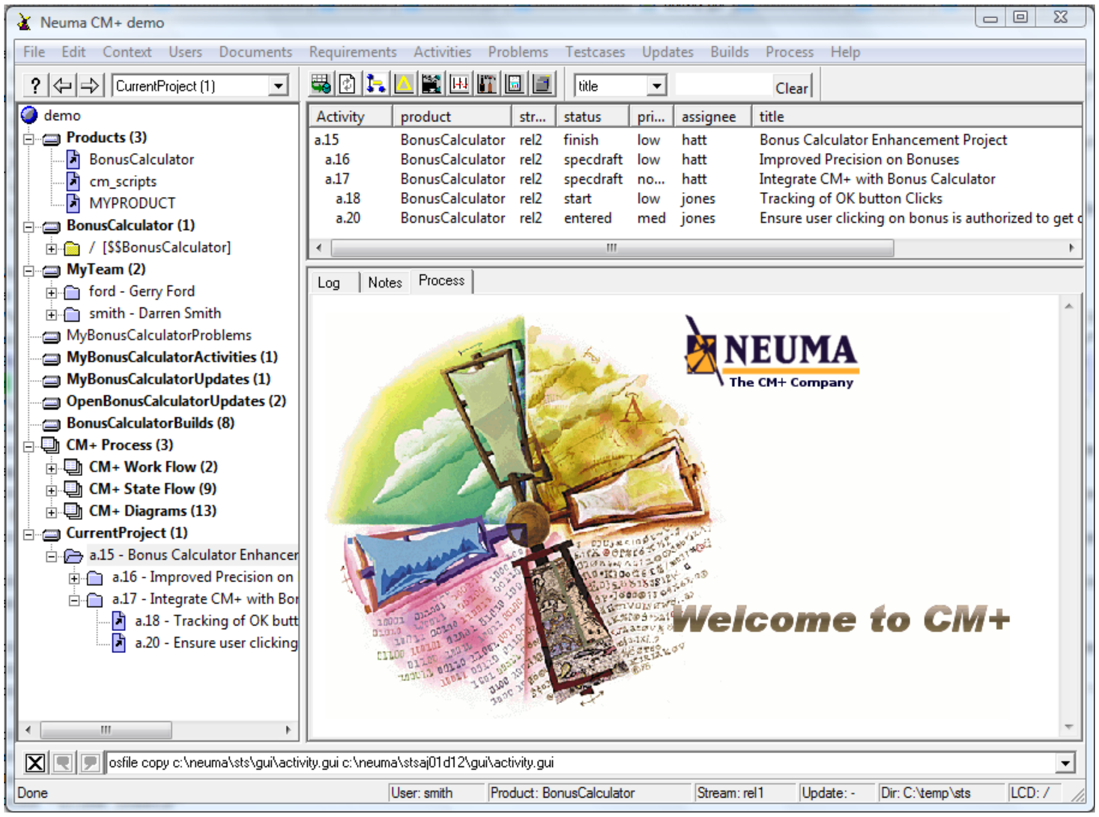
Set Product to BonusCalculator and Stream to rel2. Set the Workspace to a location in your file system where you want to deploy and edit files. After setting your context, the status bar at the bottom of the screen will change to reflect the context settings. The promotion level may be left blank for know – it permits you to omit file revisions, lower that a given promotion level, from your source tree view.

Setting a Project Context

A project manager typically works within the context of a defined project. A project may be defined de facto, such as all of the activities for a given release of a given product, or explicitly, by specifying the root of a project activity/task tree commonly known as the Work Breakdown Structure (WBS). In CM+, some of the project menus depend on the selection of a WBS project.

Establishing a project context tells CM+ that future queries and operations are relative to the project, so that the project information does not have to be specified on each form or query. In some cases, the queries or operations are available as right-click operations on activity trees or sub-trees, and do not require an explicit project context to be set.

Select **Projects/Set Project Activity…** from the **Activities** menu. A dialog will appear with a list of projects and sub projects. Select **a.15** as the project and click **OK**. Subsequent operations will use this project activity as the current project. The Current Project WBS (Work Breakdown Structure) tree will be shown in the tree browser panel. This tree may also be accessed in a 'browsetree panel' using the **Projects/Work Breakdown Structure** menu under the **Activities** menu. Both the tree browser panel of the main panel and the separate graphical browsetree panel support various right-click functions.


Neuma CM+ demo
File Edit Context Users Documents Requirements Activities Problems Testcases Updates Builds Process Help
CurrentProject (1)
title
Clear
Log
Notes
Process
NEUMA
The CM+ Company
Welcome to CM+

demo
- Products (3)
  - BonusCalculator
  - cm_scripts
  - MYPRODUCT
- BonusCalculator (1)
  - / [$$BonusCalculator]
- MyTeam (2)
  - ford - Gerry Ford
  - smith - Darren Smith
- MyBonusCalculatorProblems
- MyBonusCalculatorActivities (1)
- MyBonusCalculatorUpdates (1)
- OpenBonusCalculatorUpdates (2)
- BonusCalculatorBuilds (8)
- CM+ Process (3)
  - CM+ Work Flow (2)
  - CM+ State Flow (9)
  - CM+ Diagrams (13)
- CurrentProject (1)
  - a.15 - Bonus Calculator Enhancer
    - a.16 - Improved Precision on
    - a.17 - Integrate CM+ with Bor
      - a.18 - Tracking of OK butt
      - a.20 - Ensure user clicking

Activity	product	str...	status	pri...	assignee	title
a.15	BonusCalculator	rel2	finish	low	hatt	Bonus Calculator Enhancement Project
a.16	BonusCalculator	rel2	specdraft	low	hatt	Improved Precision on Bonuses
a.17	BonusCalculator	rel2	specdraft	no...	hatt	Integrate CM+ with Bonus Calculator
a.18	BonusCalculator	rel2	start	low	jones	Tracking of OK button Clicks
a.20	BonusCalculator	rel2	entered	med	jones	Ensure user clicking on bonus is authorized to get c

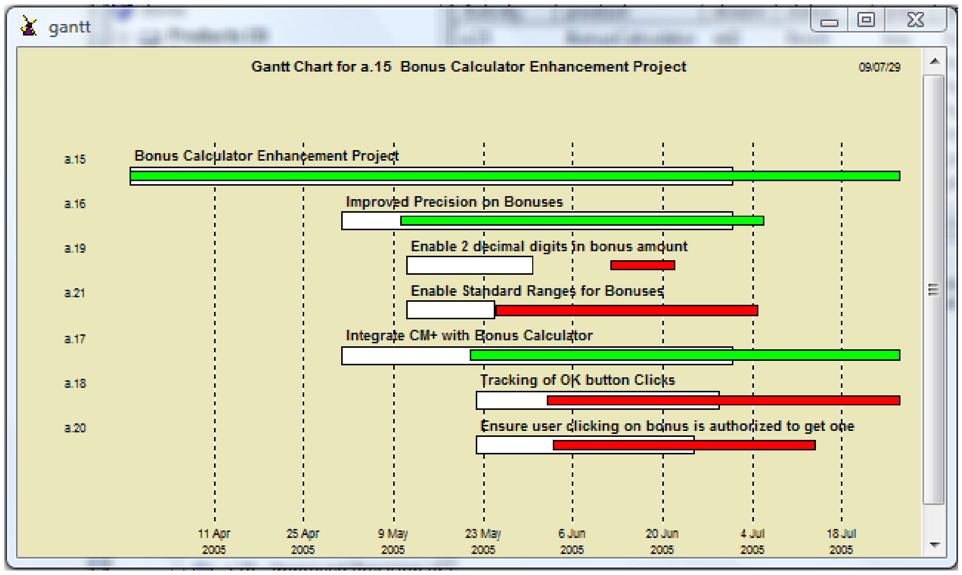
```
osfile copy c:\neuma\sts\gui\activity.gui c:\neuma\stsaj01d12\gui\activity.gui
```

Done | User: smith | Product: BonusCalculator | Stream: rel1 | Update: - | Dir: C:\temp\sts | LCD: /

Looking around this main panel screen, the key portions of the user interface are visible. At the top are the menus. Below them are navigation buttons, the tool bar, and the search bar. The left pane is the tree browser panel and contains both trees (folder icons and/or +/- expanders) and summary/to-do lists. The CM+ Process bar is also available in this pane. The top right pane is the data list pane, and lists data records, typically those selected in the tree browser pane. The search bar can be used to restrict the selection, and the titles can be used to sort the selection. The bottom pane, the tabbed pane, contains a bunch of tabs which are used to display messages, process guidance, and more complete data for a particular record. Below that is the optional command line input area, and below that, the status bar, which includes context information.

Viewing a Gantt Chart

To display a Gantt Chart for the project, **right-click** on the activity a.15 and select **Gantt Chart**. On the dialog that appears, select the **OK** button and the project Gantt chart will be shown. To show the project forecast/actual as well as the plan, select the “actual” checkbox before selecting **OK**.



Note that this demo is set up to track the traditional start and finish dates of each activity. In some configurations of CM+, activity tracking may involve multiple checkpoints (i.e. states) between the start and end of the activity. As an activity progresses through these states, the Gantt chart shows the progress along the activity's bar. For example, a development activity may be configured to move through the states: start draft reviewed implemented tested finish. If the states have been assigned colors, the actual bars will progress through the assigned colors.

In addition, CM+ permits the use of a Progress chart to demonstrate actual versus planned progress and to project completion dates based on available planned and actual data. These charts assume that activity planning and tracking data have been properly filled in for the activities.

Adding a Project Within a Development Stream

To add a new project within a particular development stream, the **Projects/Add Project**… selection from the **Activities** menu can be used. In this tutorial, we will not be adding a new project.

Adding Activities To a WBS

To add new activities to a project WBS, right-click on a parent activity and select **Add Activity** from the pop-up menu. A form will appear for adding a new activity. Fill in the various fields and select **OK** to complete the addition. For purposes of this tutorial, create an activity within the current project by right-clicking on a.16 and selecting Add Activity. Fill in the title and some notes and set the assignee as "jones". The WBS in the tree panel will be updated automatically. For a browsetree panel, select (i.e. left-click on) the parent activity to see the updated WBS.

Removing an Activity or Activity Sub-tree from a WBS

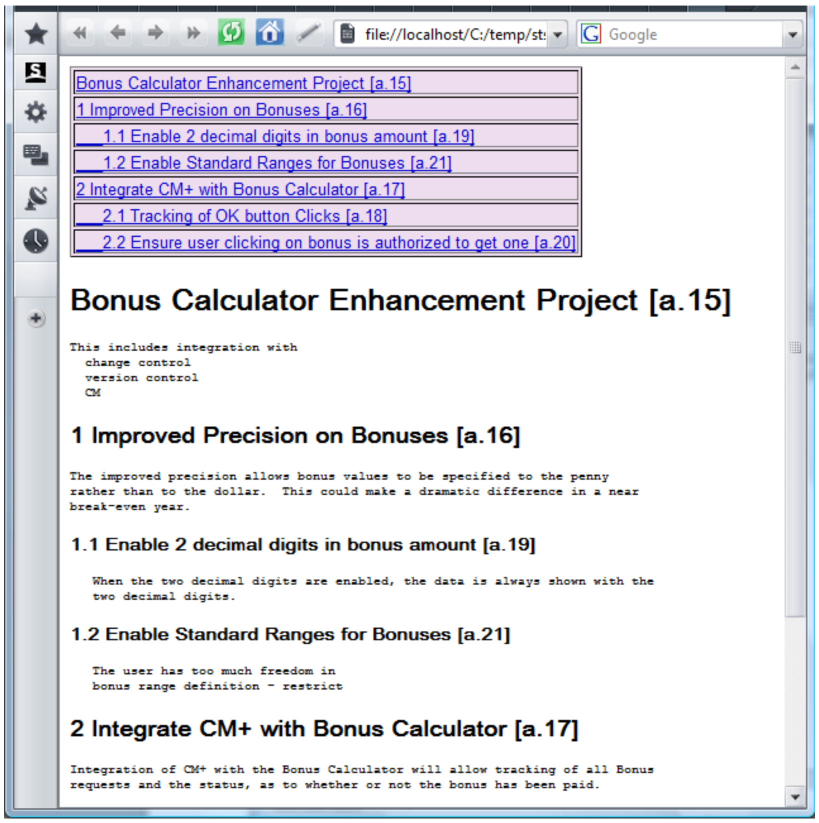
In this tutorial we do not remove any activities or sub-trees from the WBS. However, if you need to do so, display the activity you wish to remove in the WBS panel. Right click on it and select **Remove Activity**. The activity and all of its members will be removed from the WBS. The activities are not deleted from the system; they are simply removed from the project WBS. It is possible to add them into another project WBS or to add them back into the WBS at a later time.

Moving Activities in the WBS

On Windows platforms, you may move activities around in the WBS by dragging the activity you wish to from one parent activity to another parent activity. If the drag-n-drop capability is not enabled on your platform or if you prefer not to use drag-n-drop, you may right click on an activity and select Move Activity. In the tutorial, we will not be moving any activities around.

Web Display of a Project Definition

The project definition may be displayed as a Web Page generated by CM+. To display the Project Definition for the current project in a web browser, right click on a.15 and select **Project Definition**. The Web page produced shows a table of contents at the top and a numbered WBS structure expansion with full activity objectives. Selecting an item from the Table of Contents will move you to the appropriate section of the project definition. To change your Web Explorer select **Set Help Browser** from the **Help** menu.



Bonus Calculator Enhancement Project [a.15]
1 Improved Precision on Bonuses [a.16]
1.1 Enable 2 decimal digits in bonus amount [a.19]
1.2 Enable Standard Ranges for Bonuses [a.21]
2 Integrate CM+ with Bonus Calculator [a.17]
2.1 Tracking of OK button Clicks [a.18]
2.2 Ensure user clicking on bonus is authorized to get one [a.20]

Bonus Calculator Enhancement Project [a.15]

```
This includes integration with
  change control
  version control
  CM
```

1 Improved Precision on Bonuses [a.16]

```
The improved precision allows bonus values to be specified to the penny
rather than to the dollar.  This could make a dramatic difference in a near
break-even year.
```

1.1 Enable 2 decimal digits in bonus amount [a.19]

```
   When the two decimal digits are enabled, the data is always shown with the
   two decimal digits.
```

1.2 Enable Standard Ranges for Bonuses [a.21]

```
   The user has too much freedom in
   bonus range definition - restrict
```

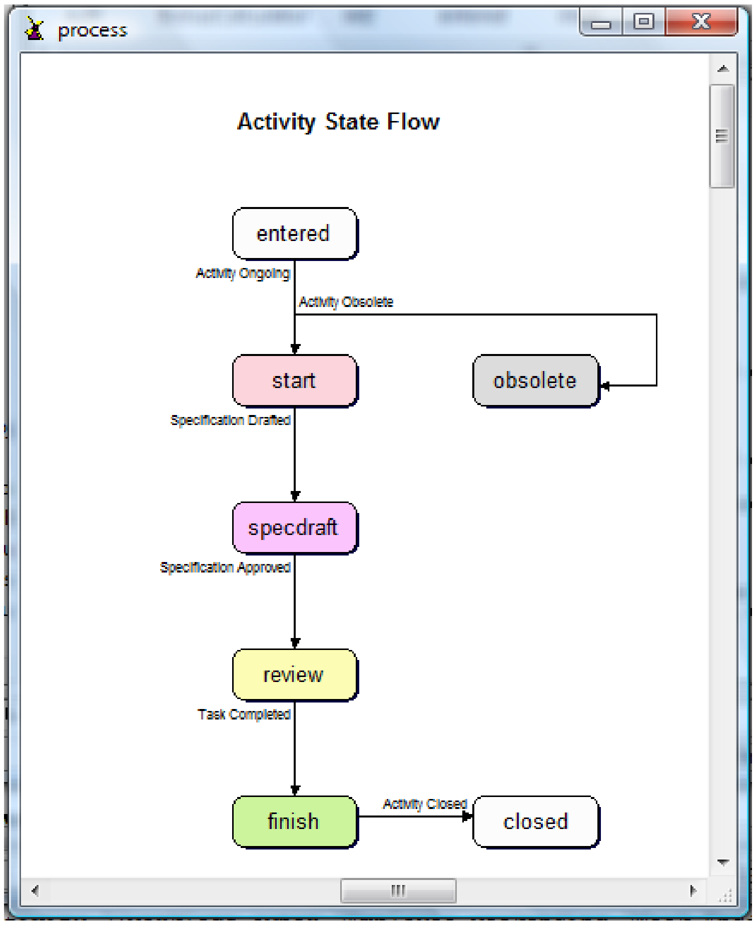
2 Integrate CM+ with Bonus Calculator [a.17]

```
Integration of CM+ with the Bonus Calculator will allow tracking of all Bonus
requests and the status, as to whether or not the bonus has been paid.
```

Project definition web pages are convenient for internet/intranet access to project information, especially for those who don't have access to CM+. Most reports can be generated in HTML format. CM+ can be customized to update your intranet reports on a regular basis. Note that some HTML format reports contain “dead” links, that is, links that cannot be followed by your web browser. In these cases, the report generator assumed that you were running the CM+ Web client which speaks with the Web Server to enable these links.

Viewing Activity Flow

To view the state-based process flow for project activities, select **View Process Flow** from the **Activities** menu. In CM+ Professional, the process flow is pre-defined and static. In CM+ Enterprise, the process flow may be modified and extended with new states, new transitions, protections and triggers. It is also possible to configure multiple process flows, depending upon the type of activity. Note that although multiple states are configured for an Activity process flow, only a subset of these states may be configured to have date/checkpoint tracking enabled (‘start’ and ‘finish’ in CM+ Professional Edition).



Activity states can be customized to add a descriptive title and a state color. The easiest way to do this is to use the **Process | Schema | Modify Range Element** menu and then select **Set Range Values**. This will bring up a display which allows you to assign titles and colors to each state. Lighter colors are recommended for best results.

Additional Project Management Operations

Some of the other operations associated with projects and activities include:

- Setting Start/End Dates
- Sizing an Activity
- Assigning an Activity
- Prioritizing an Activity

Consult the CM+ How-to documentation to learn how to perform these operations. Additional higher level operations on entire projects are also possible.

PROBLEM TRACKING

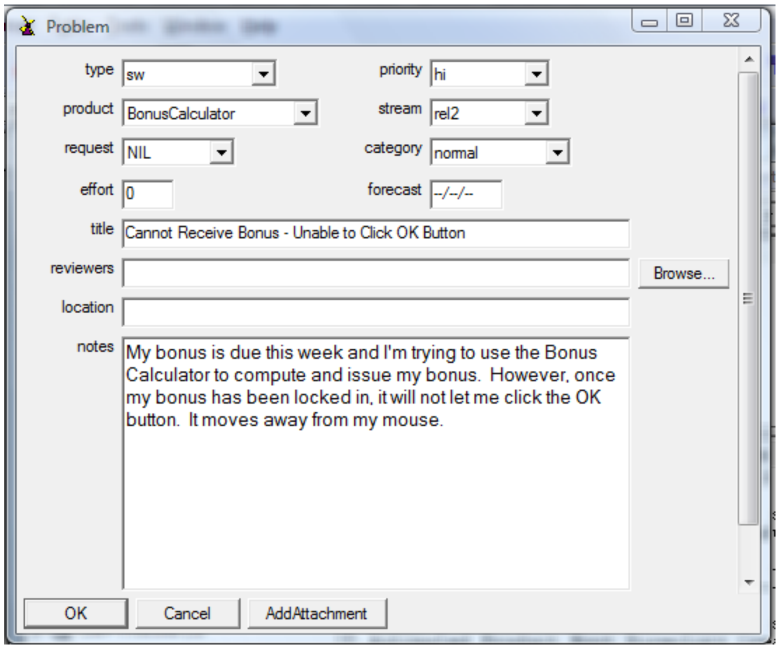
A problem report is raised to identify a non-conformance of a product or process to its specification. In this tutorial the project manager raises a problem report because the OK button does not work in the Bonus Calculator demo.

Raising a Problem Report

From the **Problems** menu select **Add Problem…** and fill in the resulting form. The Stream reflects the target stream for fixing the problem. The Priority should be set as an initial assessment (emergency for the OK button failing). This would typically be revisited by the Product Management team, Problem Review Board and/or the Change Control Board.

Note that the Product and Stream fields initially reflect the current development context as identified on the status bar at the bottom of the main window.

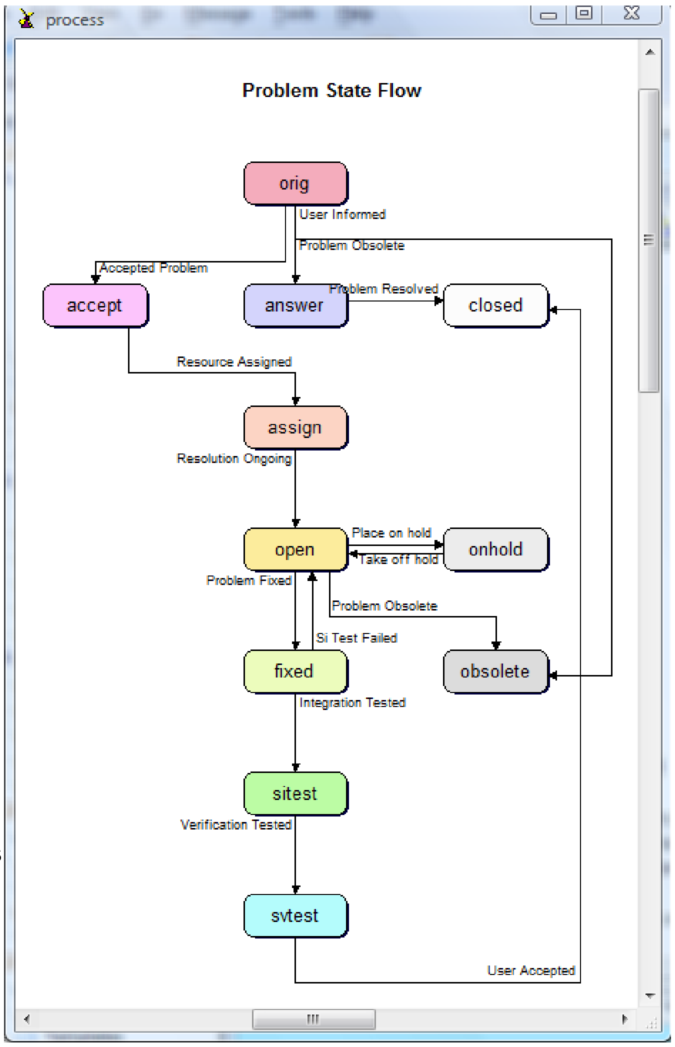
All of the fields on this form may be customized – the order, the content, the names, etc. Your project will, more likely than not, have a different set of fields available on a problem report form.



type	sw	priority	hi
product	BonusCalculator	stream	rel2
request	NIL	category	normal
effort	0	forecast	--/--/--
title	Cannot Receive Bonus - Unable to Click OK Button		
reviewers			
location			
notes	My bonus is due this week and I'm trying to use the Bonus Calculator to compute and issue my bonus. However, once my bonus has been locked in, it will not let me click the OK button. It moves away from my mouse.		

Problem Flow

To view the state-based process flow for problems, select **View Process Flow** from the **Problems** menu. In CM+ Professional Edition, the process flow is pre-defined and static. In CM+ Enterprise, the process flow may be modified and extended with new states, new transitions, protections and triggers.



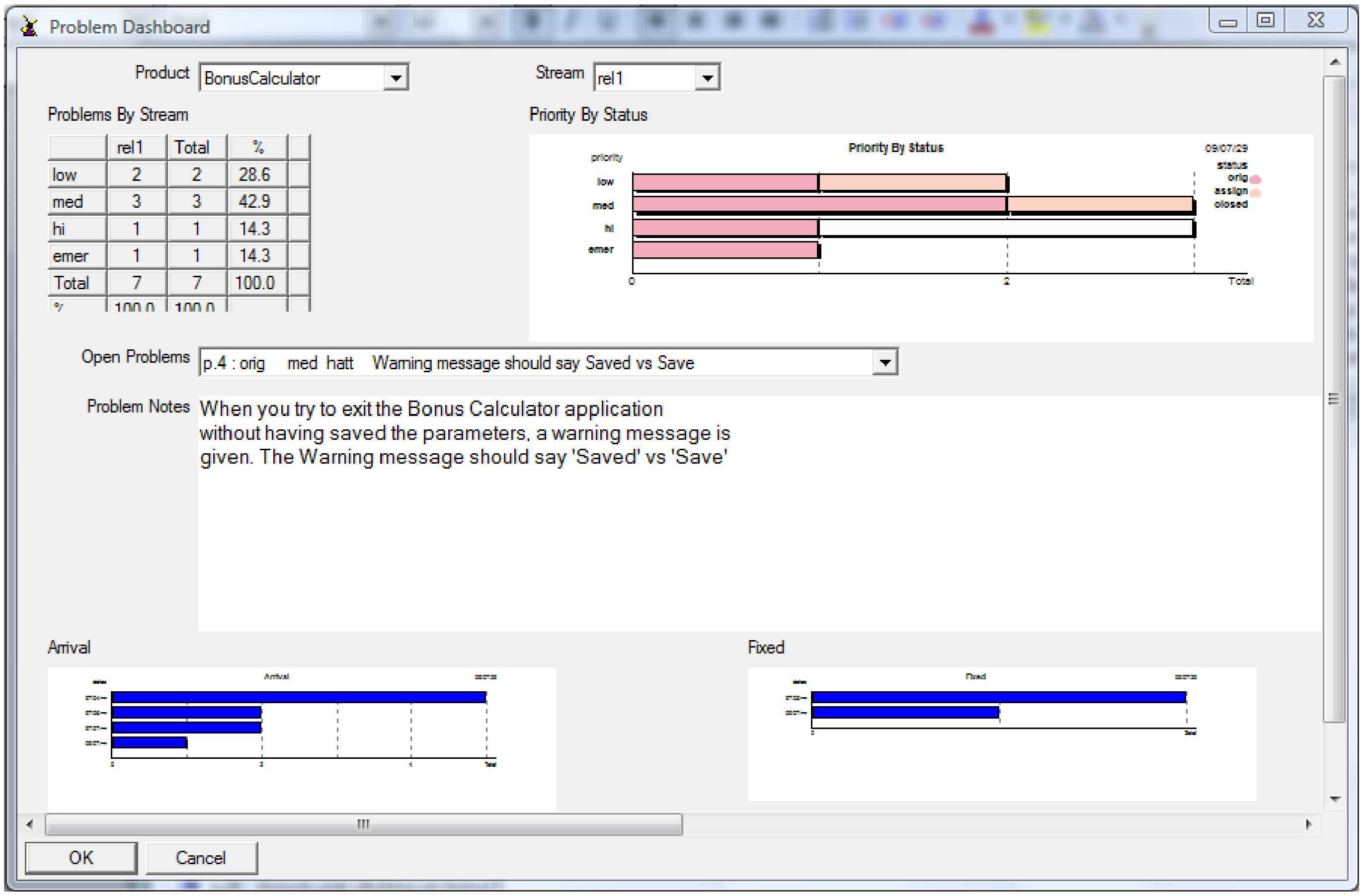
Assigning a Problem

Under the **Problems** menu, select **Assign Problem.** As project manager, you are given the option of assigning the “owner”, who is responsible for the problem, or the “assignee” who is responsible for actually developing a fix for the problem. Select Assignee and then assign the problem to “jones”, our developer in this tutorial.

At this point in the tutorial, your work as Project Manager is completed. You may now exit the CM+ session for project manager Smith.

Problem Dashboard

The default configuration of CM+ includes a problem dashboard which is most easily invoked by clicking on the tool bar icon corresponding to the Problem Dashboard. To identify the Problem Dashboard icon, roll the mouse over the tool bar icons until the toolbar tip indicating the Problem Dashboard appears. Select this icon to bring up the dashboard.



Product BonusCalculator

Stream rel1

Problems By Stream

	rel1	Total	%
low	2	2	28.6
med	3	3	42.9
hi	1	1	14.3
emer	1	1	14.3
Total	7	7	100.0
%	100.0	100.0	

Priority By Status

Open Problems p.4 : orig med hatt Warning message should say Saved vs Save

Problem Notes When you try to exit the Bonus Calculator application without having saved the parameters, a warning message is given. The Warning message should say 'Saved' vs 'Save'

Arrival

Fixed

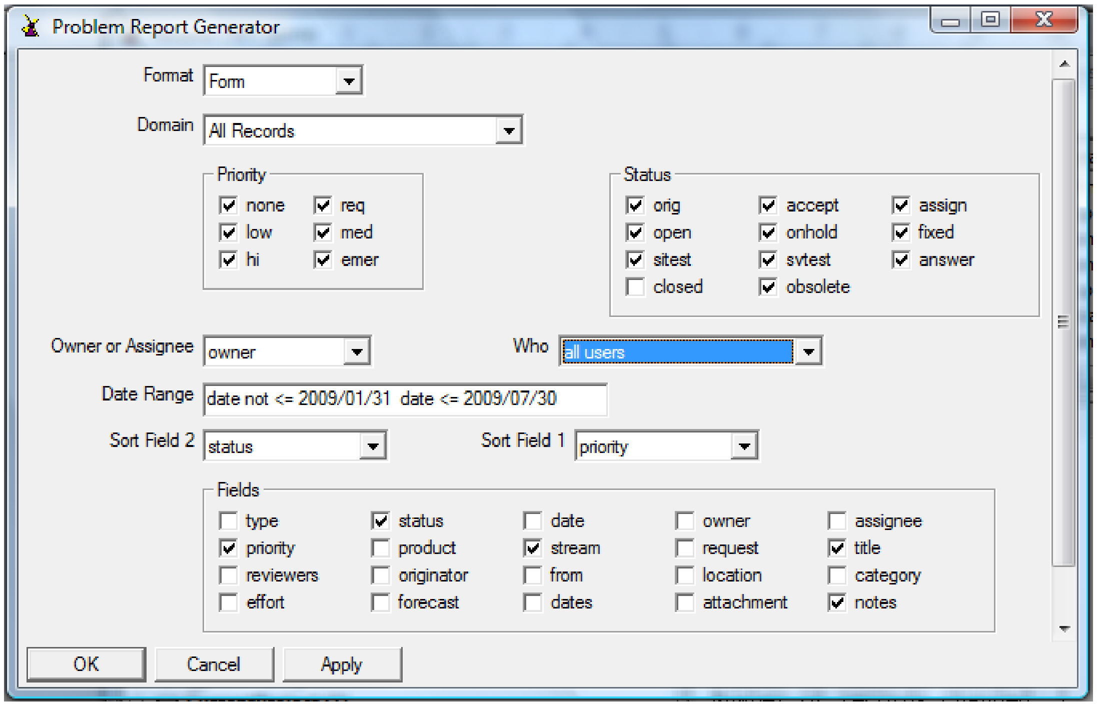
Notice that the dashboard has several different fields and displays. Some of these are used to focus the dashboard on a particular set of data. The Product and Stream selectors indicate the problems that we're interested in looking at. The table summary shows us that, in this case, for this product, there are only release 1 problems, summarized by priority. Clicking one of these summary cells will either bring up a display panel with the details, or will place those details in the main data panel, depending upon your configuration. The “Priority by Status” graphs shows a distribution of the problems according to those two factors. Clicking on a bar will zoom into a display panel for the problems contributing to that particular bar. The “Open Problems” selector allows you to zero in on the description of open problems without having to go to another panel. The final two graphs show the problem Arrival and Fix Rates, by month.

Your own project may have any number of customizations added to the Problem Dashboard. From a Project Manager perspective, this dashboard is intended to show trends, risks and other information that is valuable to management of the project. A tester may have a completely different view of problems on his/her dashboard, perhaps oriented towards the task of integration testing of new builds.

Problem Reporting

CM+ provides a number of ways to report on problems, both interactively and through various report formats. The report generator is itself a cutomizable panel from which details of the report are specified, including format, query domain, ownership, sorting, field selection and additional report selection criteria. Graphical reports are also generated in a similar fashion.

From the Problems menu select the Report item. A panel will come up with several reporting options.



Problem Report Generator

Format: Form

Domain: All Records

Priority: ☑ none ☑ req ☑ low ☑ med ☑ hi ☑ emer

Status: ☑ orig ☑ accept ☑ assign ☑ open ☑ onhold ☑ fixed ☑ sitest ☑ svtest ☑ answer ☐ closed ☑ obsolete

Owner or Assignee: owner

Who: all users

Date Range: date not <= 2009/01/31 date <= 2009/07/30

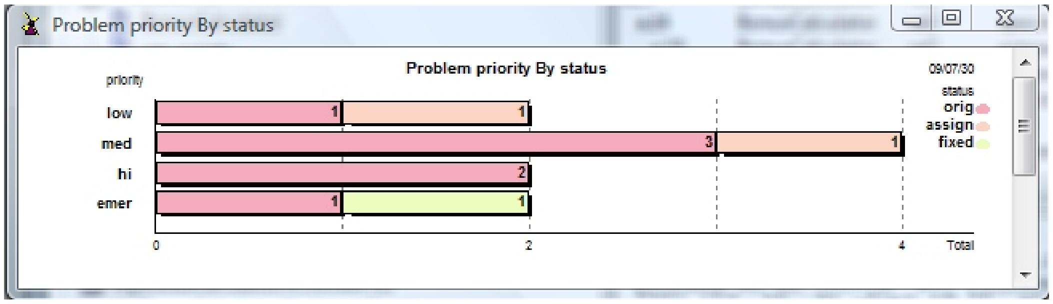
Sort Field 2: status

Sort Field 1: priority

Fields: ☐ type ☑ status ☐ date ☐ owner ☐ assignee ☑ priority ☐ product ☑ stream ☐ request ☑ title ☐ reviewers ☐ originator ☐ from ☐ location ☐ category ☐ effort ☐ forecast ☐ dates ☐ attachment ☑ notes

The problem report generation may be used to generate problem reports and queries in many different formats. First select the “Form” format, select “all users” for Who, and in addition to the already selected fields, select “title” and “notes” (right hand column). Now select Apply (which will keep the report generator up as well, as opposed to OK which will close it). A Form browser appears with Previous and Next buttons for navigating the forms. Experiment with other formats and options as well.

Now go back to the Problems menu and select Graph. A similar form appears, but this time to customize a graphical presentation of data. This time select “all users” (under “Who”) and select the “Show Data Values” option. A stacked bar graph will appear, show status, based on color coding of the states, within each priority. Click on a bar segment to explore the problems within that segment.



Additional problem queries and reports are available using the Problem Queries and Problem Reports sub-menus of the Problems menu.

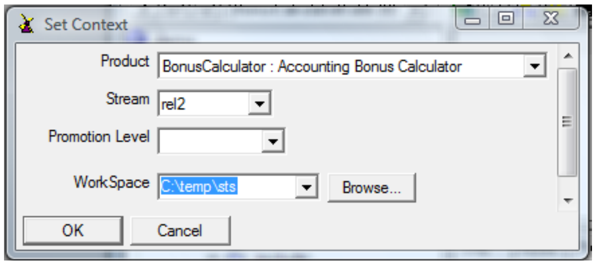
DEVELOPER ROLE

Developer Overview

The designer or developer is perhaps the most common role in most development projects. For this part of the tutorial, you will enter the CM+ demo library as user "jones", our Designer. Select the **CM+ Demo Designer** from the **CM+** menu under **Start/Programs** (or run **C:\Neuma\sts\bin\sts demo -g -u jones**). On UNIX, use the "-u jones" option when you start CM+: **sts demo -g -u jones**

Setting a Development Context

CM+ maps your product roadmap (release plans) onto tracked **products** and development **streams**. Setting a development context enables you to control your view of the repository data, and ensures you see the correct revisions of files and directories. This tutorial assumes the **BonusCalculator** product in the **rel2** stream. Set your development context by selecting **Set Context** from the **Context** menu.



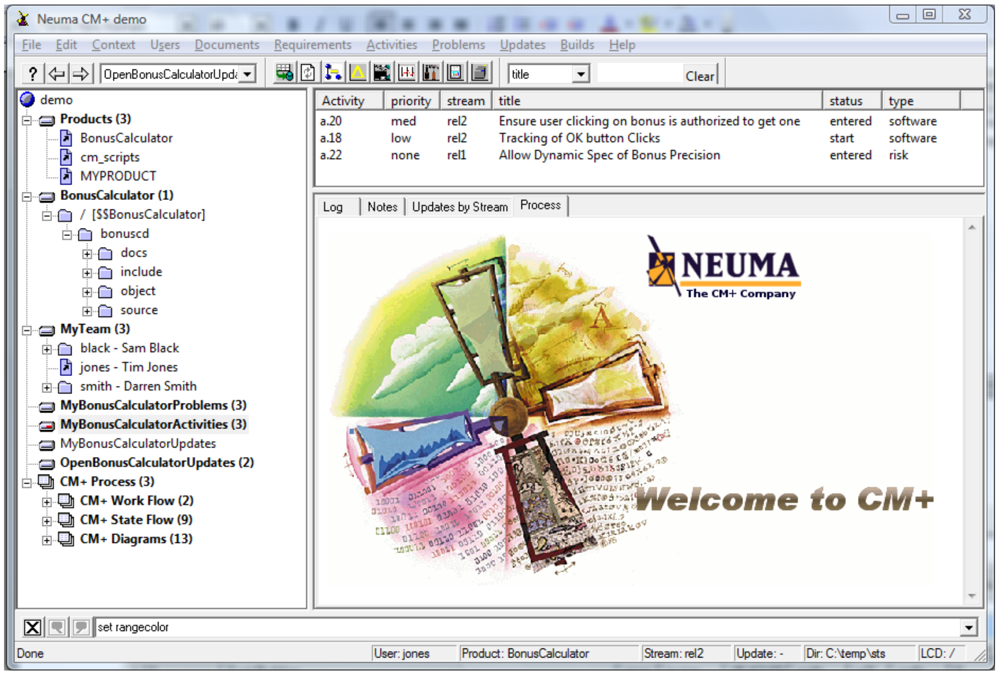
Product: BonusCalculator : Accounting Bonus Calculator
Stream: rel2
Promotion Level:
WorkSpace: C:\temp\sts

Set Product to BonusCalculator and Stream to rel2. Set the Workspace to a location in your filesystem where you want to deploy and edit files. After setting your context, the status bar "Dir" field will change to reflect the context settings.

CM+ will remember your context settings until you change them. When you attempt to select a previously active context, CM+ will default the WorkSpace to your previously used workspace, and will provide a selection of WorkSpace settings that you have previously used with the specified Product and Stream.

Identifying Assigned Activities

To identify the activities which have been assigned to you, on Windows, Linux and some Unix platforms, either right click on your name in the Staff tree and select **To Do Activities** or click on **MyBonusCalculatorActivities** in the Tree Browser panel. On any platform you may bring up the Staff tree using Users/Staff Tree, or you may select **My Activities** directly from the **Activities** menu.



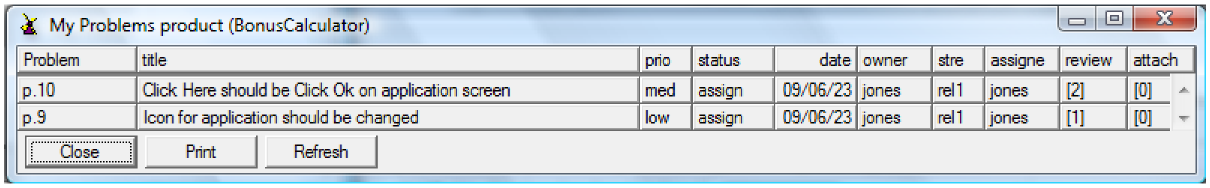
Activity	priority	stream	title	status	type
a.20	med	rel2	Ensure user clicking on bonus is authorized to get one	entered	software
a.18	low	rel2	Tracking of OK button Clicks	start	software
a.22	none	rel1	Allow Dynamic Spec of Bonus Precision	entered	risk

Done | User: jones | Product: BonusCalculator | Stream: rel2 | Update: - | Dir: C:\temp\sts | LCD: /

In your own project library, you may find that your To-Do lists are named differently (e.g. MyChangeRequests), reflecting your own corporate or project processes and terminology.

Identifying Assigned Problems

To identify the problems which have been assigned to you, on most platforms, either right click on your name in the Staff tree and select **To Do Problems** or click on **MyBonusCalculatorProblems** in the Tree Browser panel. On any platform you may bring up the Staff tree using Users/Staff Tree, or you may select **My Problems** directly from the **Problems** menu. Notice that the problem assigned to “jones” by the Project Manager is now in Jones’ problems list.

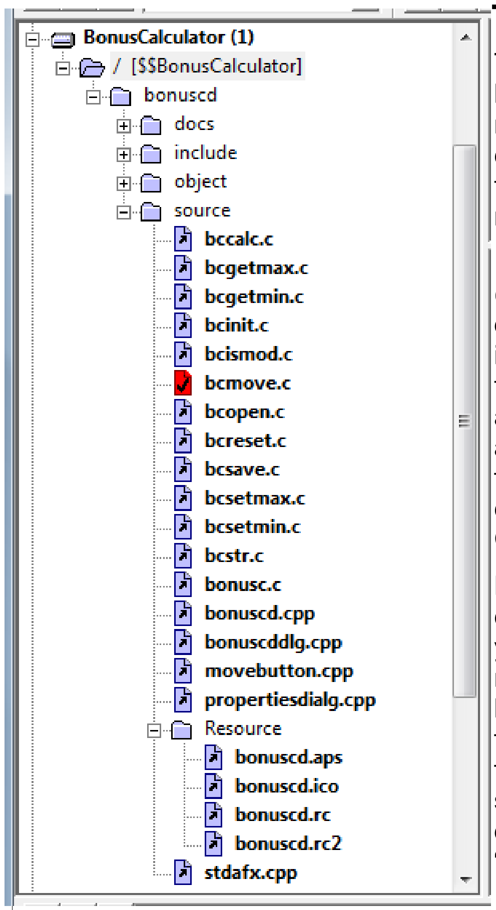


Problem	title	prio	status	date	owner	stre	assigne	review	attach
p.10	Click Here should be Click Ok on application screen	med	assign	09/06/23	jones	rel1	jones	[2]	[0]
p.9	Icon for application should be changed	low	assign	09/06/23	jones	rel1	jones	[1]	[0]

Changing your Workspace

To change your workspace only, select **Set Current Workspace** under the **Context** menu. Whenever you start work on an Update, the current workspace setting will be saved by default as the workspace referenced by that Update. Files will be retrieved to this workspace and placed in appropriate sub-directories. When you check-in the Update, files will be submitted from this workspace. This way you can work on different products or streams in parallel and always submit files from the correct location.

Note that your workspace always starts at the product root (i.e. the product root maps onto the workspace directory, and all subdirectories are positioned according to the product tree layout).


BonusCalculator (1)
/ [$$BonusCalculator]
bonuscd
docs
include
object
source
bccalc.c
bcgetmax.c
bcgetmin.c
bcinit.c
bcismod.c
bcmove.c
bcopen.c
bcreset.c
bcsave.c
bcsetmax.c
bcsetmin.c
bcstr.c
bonusc.c
bonuscd.cpp
bonuscddlg.cpp
movebutton.cpp
propertiesdialg.cpp
Resource
bonuscd.aps
bonuscd.ico
bonuscd.rc
bonuscd.rc2
stdafx.cpp

The Source Tree

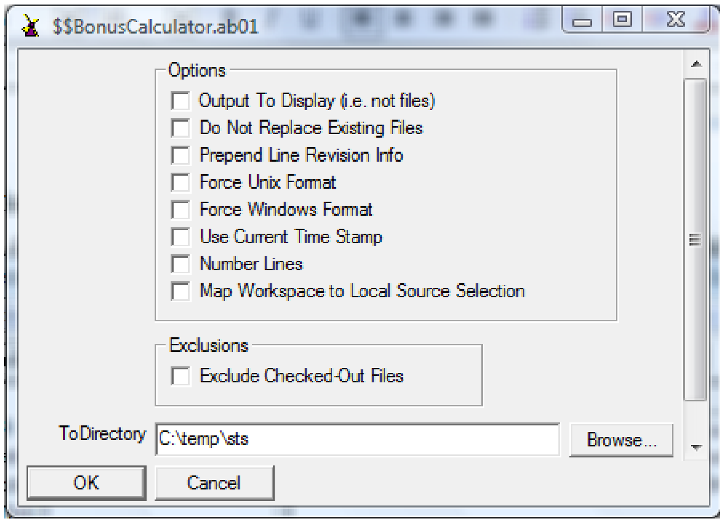
The source tree is the file-explorer like directory/file tree found in the browser portion of the main window. The root of the source tree is normally named with the product name (e.g. BonusCalculator). In some cases, you may have multiple source trees visible at once, corresponding to multiple products in your context. Underneath the root are typically multiple directories.

In the source tree shown on the left, there is a main product root directory (/) under which there is a single (project) directory bonuscd, with 4 directories under it. The "source" directory is fully expanded. The default icon shows that there is a file in the workspace for each file in the source tree, and the bold text indicates that the workspace file has read/write attributes. The red checkmark indicates that "bcmove.c" is checked out by another user. There are numerous indicators to let you know the status of files in the workspace, including a delta symbol which appears if the file changes in the workspace from what it is in your current context of the CM+ repository.

Many operations may be performed from the source tree. You may right-click anywhere in the tree and ask that that file or sub-tree be Compared to your Workspace. Or you may ask for a line count of that sub-tree. You may also deploy source for the sub-tree. Typically, one would right-click to browse history, check-out a file, or view the revision tags associated with a file. In our example, we'll right-click on the "source" directory and search for the word "MOUSE", by selecting the Search option and typing in the search term "MOUSE", followed by clicking OK. CM+ puts up a text display (or in some cases a text tab panel) to show the lines in the files of "source" which have the term MOUSE used.

NOTE: You may customize your editor integration so that you may right-click on a search line and select "Go To Line", so that CM+ will place you in your editor at the specific line. This assumes that your editor has the capability to place you at a specific line of the file (e.g. Notepad does not, but Notepad++, and vi,/vim/gvim do). This is a powerful feature if you don't have an IDE that for this, but it is also in that you may select which portion of the source tree to search (by right-clicking there) and you may customize the right-click menu to have other actions besides "Go To Line".

Deploying a Source Tree


$$BonusCalculator.ab01

Options
- [ ] Output To Display (i.e. not files)
- [ ] Do Not Replace Existing Files
- [ ] Prepend Line Revision Info
- [ ] Force Unix Format
- [ ] Force Windows Format
- [ ] Use Current Time Stamp
- [ ] Number Lines
- [ ] Map Workspace to Local Source Selection

Exclusions
- [ ] Exclude Checked-Out Files

ToDirectory: C:\temp\sts

Depending on your tools and their configuration, you may need the entire source tree deployed in your workspace or only those files which you plan to modify. CM+ supports either model. In this tutorial, the entire source tree will be deployed in your workspace.

You may deploy your source tree it by right-clicking on the root of the **BonusCalculator** tree, "/", and selecting **Get Source**. On some platforms, or configurations, you may first have to display the tree by selecting **View Context Tree** from the Context menu. Another way to deploy the source tree, or selected portions of it, is by selecting **Populate…** from the **Updates/Workspace** menu. In the resulting dialog, you select one or more directories and click OK. The source code will be deployed to your current workspace. Now try deploying code by right-clicking on the tree root and selecting Get Source. A dialog panel with a number of options appears. By default, selecting no options will deploy the source normally.

The source deployed reflects the source visible in your current workspace. To deploy source code for a particular build, you may either set your context to that build, or use the menu item **Builds | Get Build**.

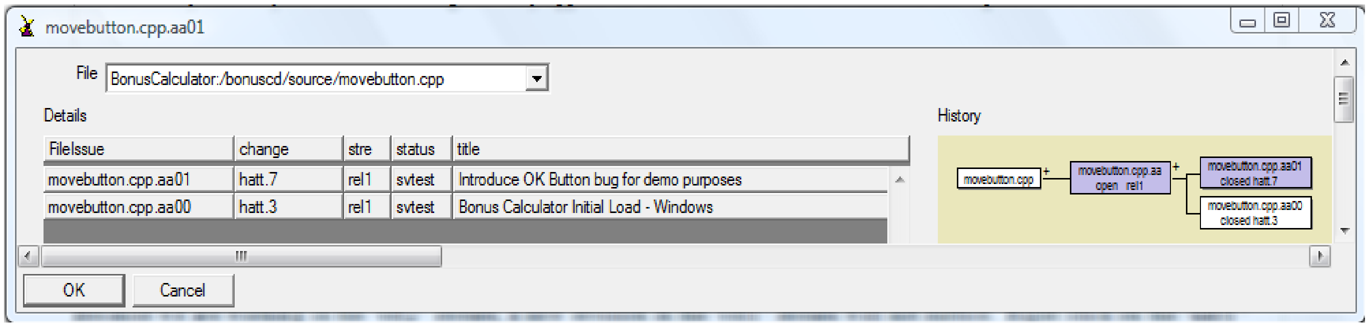
You may use some of the options on the Get Source dialog to change how you retrieve source. Many of these options are more typical for a single source file. The **Output to Display** option will bring up a text window with the source in it. The **Prepend Line Revision Info** option and the **Number Lines** options may be used to append information on what revision each line of the file was added, and line numbers, respectively. You may also override the native format when retrieving source code.

Fixing a Problem

Go back to the display of Jones's assigned problems and identify the emergency problem report that was added by the Project Manager (unable to Collect Bonus/click on OK button). Right-click on this problem and select **FixProblem** from the pop-up menu. A new Update form will appear allowing you to define an Update to fix the problem. Notice that the product, stream and directory reflect your context settings as they appear in the status bar. Fill in a title for the Update and select OK.

Checking out a File and Branching

The file to be modified in this demo is the "movebutton.cpp" file. Open the Context Tree for Bonus Calculator and expand the "source" directory. You'll notice bcmove.c is already checked out (red checkmark or red text) by another user. When you check out movebutton.cpp it will appear using green instead of red because the checkout is to the current user. Although you may select **Check Out** from the pop-up menu on "movebutton.cpp", we're going to select **Browse History** instead. This brings up a history browser. When expanded fully, notice that a single "rel1" (Release 1) branch of movebutton.cpp exists, and that it has two revisions, aa00 and aa01.


movebutton.cpp.aa01

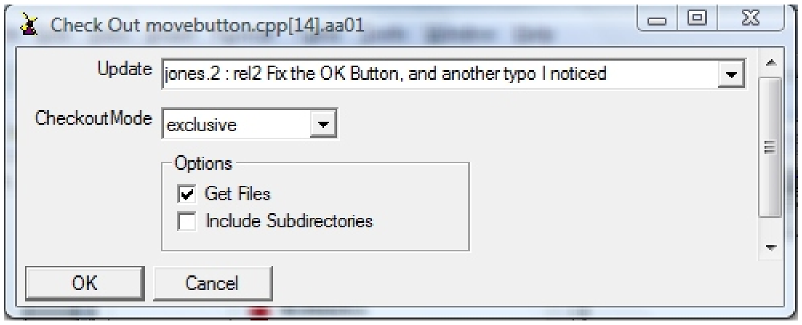
File: BonusCalculator:/bonuscd/source/movebutton.cpp

Details

FileIssue	change	stre	status	title
movebutton.cpp.aa01	hatt.7	rel1	svtest	Introduce OK Button bug for demo purposes
movebutton.cpp.aa00	hatt.3	rel1	svtest	Bonus Calculator Initial Load - Windows

Because we are working in the "rel2" stream, a new revision in the "rel1" stream will not suffice. Right-click on the "aa01" revision and select **Check Out**. The Check Out panel appears asking you to select an Update and Options. Select the Update you just created and leave the "Get Files" option checked to ensure the correct version is retrieved to your workspace. Then click OK.

Note that the CheckoutMode in this tutorial is "exclusive" meaning no one else can check out this file in this release branch until your checkout is completed or cancelled. Some configurations will allow "parallel" checkouts, where multiple users can checkout the same file revision in parallel, though this requires reconciliation on checkin. And some will allow "queued" checkouts, where an update can queue to check out a file that has already been checked out by someone else. The checkout operation will complete when the previous checkout is completed. [Note that the term "checkout" refers to reserving or registering a file for change. It does not refer to source retrieval, though this is done with the "Get Files" option.]

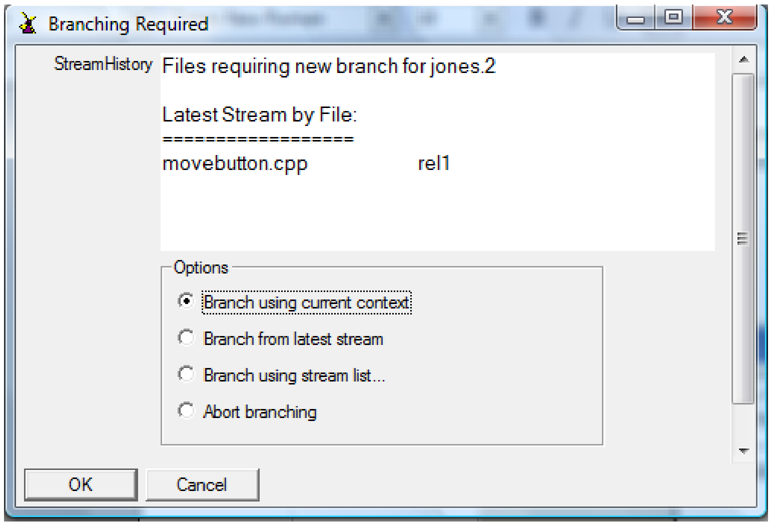

Check Out movebutton.cpp[14].aa01

Update: jones.2 : rel2 Fix the OK Button, and another typo I noticed

CheckoutMode: exclusive

Options
- [x] Get Files
- [ ] Include Subdirectories

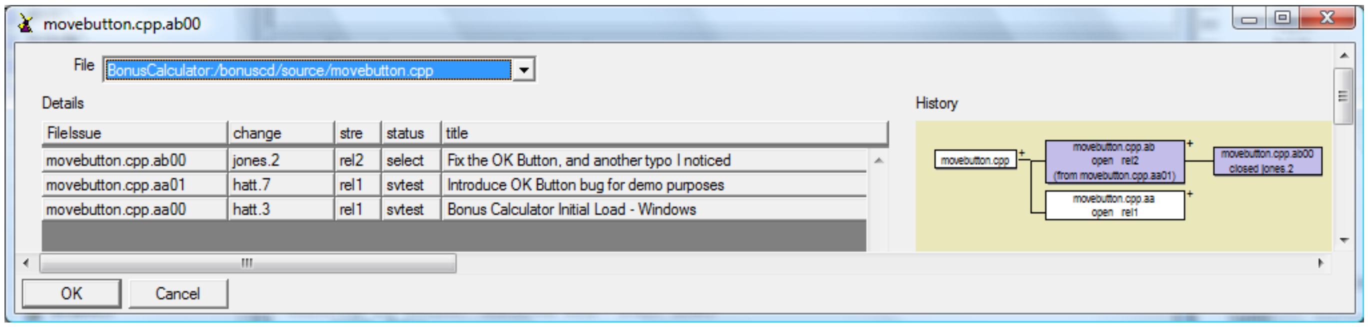
Because there is only a rel1 branch of the file, CM+ notifies the user that a branch operation is required in order to open up a rel2 branch of the file. Branching occurs from the current context by default. Select OK, and CM+ creates a new branch.



```
Files requiring new branch for jones.2

Latest Stream by File:
==================
movebutton.cpp              rel1
```

Options
- (●) Branch using current context
- ( ) Branch from latest stream
- ( ) Branch using stream list...
- ( ) Abort branching

Click on the root node of the history browser and notice that there are now two branches, a rel1 "aa" branch and a rel2 "ab" branch and that the "ab" branch has a single "ab00" revision which is labeled with the current update for Jones. The shading, which indicates which branch and revision are in your current context, has moved to highlight revision "ab00" in the rel2 branch. At this point you may close the history browser.



File: BonusCalculator:/bonuscd/source/movebutton.cpp

Details

FileIssue	change	stre	status	title
movebutton.cpp.ab00	jones.2	rel2	select	Fix the OK Button, and another typo I noticed
movebutton.cpp.aa01	hatt.7	rel1	svtest	Introduce OK Button bug for demo purposes
movebutton.cpp.aa00	hatt.3	rel1	svtest	Bonus Calculator Initial Load - Windows

Editing Source Code

Once source code is retrieved from CM+ it may be edited through CM+ or through any external tools. To edit from within CM+, right-click on the object and then select **Edit,** or just double-click on the file. Note that the Edit option will only appear if the file has been checked out (otherwise a View operation – read-only edit – will occur).

On Windows platforms, CM+ will try to use the Windows registry to infer the correct editor. However, if you wish to override this behavior, the File Class for .cpp files may be configured to specify a different editor. This must be done through the Administrator or CM Manager role. On UNIX, the edit rule from the file class definition will be used automatically if one is defined.

If CM+ does not otherwise know what editor to use, the default editor is used, as defined by the "Editor" options preference. Set your options by selecting **Options…** from the **Edit**, **Preferences** menu.

Any appropriate tool may be used to edit code outside the repository. Typically, an IDE tool is used, such as Visual Studio. In many cases, it is possible to integrate IDE tools with CM+ so that Check Out and Check In operations may be done directly from the IDE. CM+ is Microsoft SCC (Source Code Control) compliant, so any Windows IDE supporting this API can be used with CM+. CM+ also has an Eclipse plug-in for using an Eclipse IDE with the CM+ repository.

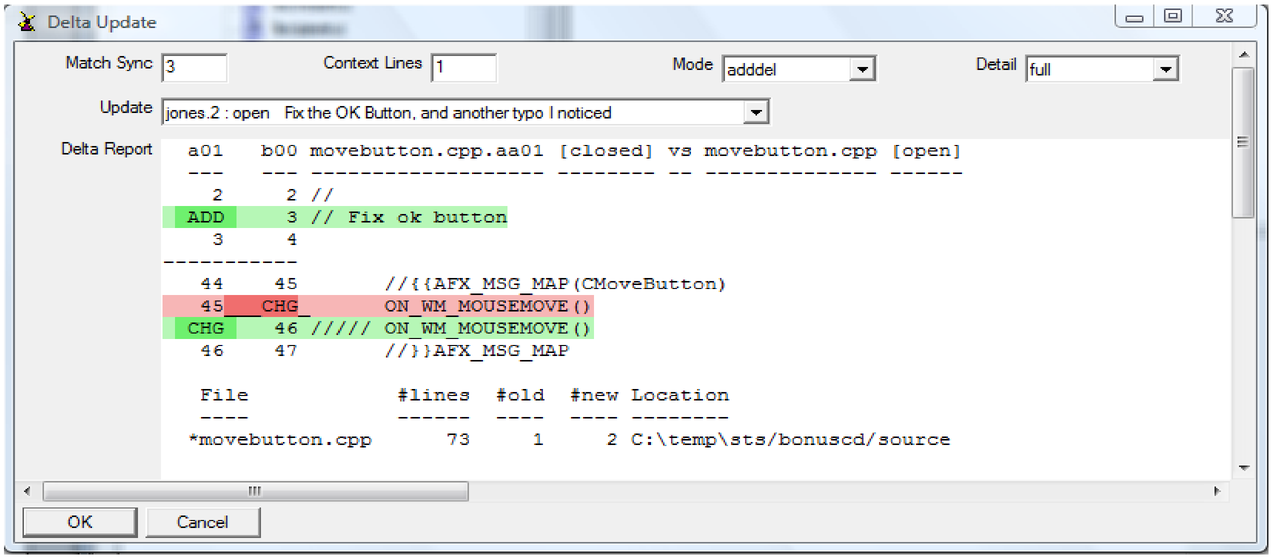
In the Source tree of the CM+ demo library, you will notice that movebutton.cpp has a green check mark indicating that you have checked it out in this development stream. Edit the "movebutton.cpp" file from the source tree. Locate the line containing "ON_WM_MOVEMOUSE" and comment it out. You may also want to add a comment there, and perhaps one to the beginning of the file. Finally, save the changes you have made.

Note that in CM+ there is no need to keep a running comment log of changes in a file. CM+ can insert such comments into the source file without any manual editing.

Creating a Delta Report

To create a delta report of the changes you have made to all of the files of your update (just movebutton.cpp in this tutorial), select **Delta Update** from the **Updates/Delta** menu. You can bring up similar forms in a number of different ways including: selecting MyBonusCalculatorUpdates in the browser panel and right-clicking on the update and selecting “Delta”; clicking the large delta symbol in the tool bar; right-clicking on the movebutton.cpp file and selecting “Delta”. The delta report will appear showing you how many lines there are in each changed file, how many old and new lines were affected by the change and the specific changes, before and after. Note that just a few lines of a single file has been modified in this demo example. If multiple files had been modified, all of the changes in all of the files would be shown and summarized at the bottom.

You may experiment with the various delta options at the top of the panel. For example, change the Context Lines value from 0 to 1 and hit Tab (or, if a Refresh button appears on the bottom right corner you may select it or hit enter). Your report will change to show some context around the change, as in the diagram below.



Match Sync: 3 Context Lines: 1 Mode: adddel Detail: full

Update: jones.2 : open   Fix the OK Button, and another typo I noticed

Delta Report:

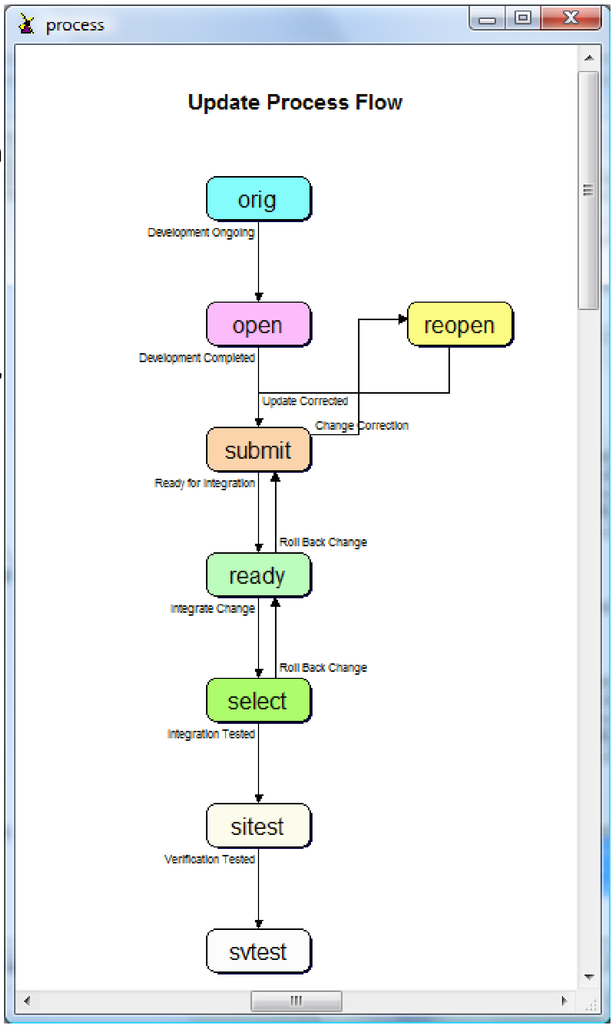
```
a01   b00 movebutton.cpp.aa01 [closed] vs movebutton.cpp [open]
---   --- ------------------- -------- -- -------------- ------
  2     2 //
ADD     3 // Fix ok button
  3     4
-----------
 44    45        //{{AFX_MSG_MAP(CMoveButton)
 45   CHG        ON_WM_MOUSEMOVE()
CHG    46 ///// ON_WM_MOUSEMOVE()
 46    47        //}}AFX_MSG_MAP

 File             #lines  #old  #new Location
 ----             ------  ----  ---- --------
*movebutton.cpp       73     1     2 C:\temp\sts/bonuscd/source
```

Checking in an Update

To check in the update, select **Check In Update** from the **Updates** menu. A panel will appear for you to select the update. Select jones.2 and then click “Keep files in folder” if you wish to keep the modified files in your workspace. You may also invoke the check-in functionality by selecting MyBonusCalculatorUpdates in the browser pane, and then right-clicking on the jones.2 updates and selecting “Check In”. When you check in the file, CM+ automatically promotes the linked problem record to “fixed”. The green check mark will disappear for the file(s) in the checked in update (in this case, movebutton.cpp).

Promoting an Update to Ready

Select **View Process Flow** from the **Updates** menu. This will reveal the state-based process flow for updates. In CM+ Professional, the process flow is pre-defined and static. In CM+ Enterprise, the process flow may be modified and extended with new states, new transitions, protections and triggers. It is also possible to configure multiple process flows, depending upon the type of update (e.g. documentation changes vs. code changes).



Notice that the update has moved to “submit” state automatically when it was checked in. However, with the default process, the designer must also promote it to the “ready” status to signal to the CM manager that it is ready to integrate into a build. This extra step allows designers to check in code without it automatically being made available for integration.

In order to promote the update, select **Mark Update Ready** from the **Updates** menu. Select Jones’s update from the list and click OK. This will promote the update to the ready status, but it will also check that there are no other updates which need to be promoted prior to the promotion of this update.

When searching for dependent updates, CM+ looks for both explicit and implicit dependencies. An explicit dependency is specified directly on the update. An implicit dependency is an update that produced an earlier revision of a file changed by your update.

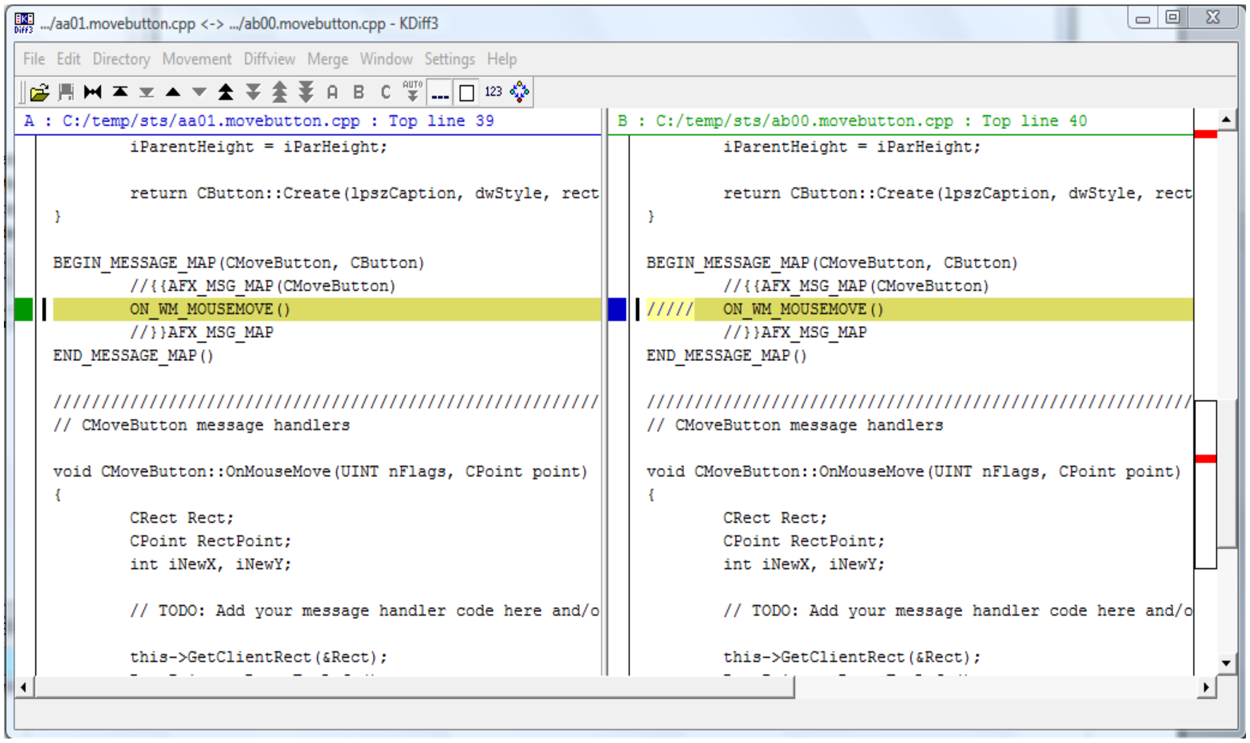
If any explicit or implicit dependencies are not yet promoted to ready, CM+ identifies them and warns you to take appropriate action. The CM manager receives the same warnings when he/she promotes updates in preparation for a build cycle.

Comparing File Revisions

If you wish to compare two revisions of a file, open the History Browser by right-clicking on the file and selecting **Browse History**. Expand the branch of the older of the two revisions you wish to compare and right-click on the revision and choose **Select Delta File**. Then navigate to the more recent revision to which you wish to compare it and choose **Delta Vs Selection** from the pop-up menu. CM+ will display the differences between the two revisions. This is useful when looking for all of the changes between the latest revision and the customer’s revision of the product.

Visual Delta

Once a file has been checked into CM+, its possible to generate a visual delta for the file. CM+ uses the term “visual delta” to refer to one using a graphical delta reporting tool. These typically give side-by-side delta reports. Right-click on the movebutton.cpp file in the Source tree and select **Visual Delta.** Or else right-click and choose **Browse History**. When the history browser appears, open a branch, right click on one of the revisions, and select **Visual Delta**. This will bring up a visual differencing tool called KDiff3. You may browse through the changes of a file revision, one by one using this tool. In this example, there is only one or two differences shown. In CM+ it is possible to configure the visual differencing tool to use any third party tool.

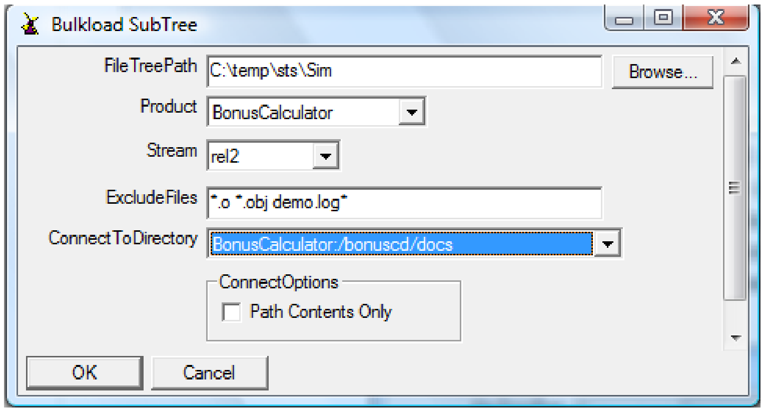

.../aa01.movebutton.cpp <-> .../ab00.movebutton.cpp - KDiff3
File Edit Directory Movement Diffview Merge Window Settings Help

```
A : C:/temp/sts/aa01.movebutton.cpp : Top line 39
          iParentHeight = iParHeight;

          return CButton::Create(lpszCaption, dwStyle, rect
}

BEGIN_MESSAGE_MAP(CMoveButton, CButton)
          //{{AFX_MSG_MAP(CMoveButton)
          ON_WM_MOUSEMOVE()
          //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////
// CMoveButton message handlers

void CMoveButton::OnMouseMove(UINT nFlags, CPoint point)
{
          CRect Rect;
          CPoint RectPoint;
          int iNewX, iNewY;

          // TODO: Add your message handler code here and/o

          this->GetClientRect(&Rect);
```

```
B : C:/temp/sts/ab00.movebutton.cpp : Top line 40
          iParentHeight = iParHeight;

          return CButton::Create(lpszCaption, dwStyle, rect
}

BEGIN_MESSAGE_MAP(CMoveButton, CButton)
          //{{AFX_MSG_MAP(CMoveButton)
/////     ON_WM_MOUSEMOVE()
          //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////
// CMoveButton message handlers

void CMoveButton::OnMouseMove(UINT nFlags, CPoint point)
{
          CRect Rect;
          CPoint RectPoint;
          int iNewX, iNewY;

          // TODO: Add your message handler code here and/o

          this->GetClientRect(&Rect);
```

Bulkloading a Sub-Tree

To load in new directory sub-tree and connect it into the product tree, select **BulkLoad SubTree** from the **Updates/New** menu. Point to the root of the sub-tree to specify the FileTreePath, and specify files and directories to be excluded (e.g. tmp *.o *.obj). Then specify where in the existing source tree the new sub-tree is to be connected, and finally whether the sub-tree directory root itself is supposed to be connected under the connection point, or if only the contents under the directory root are to be connected. In the latter case, the sub-tree root and the connection point are considered logically the same directory. To try this in the demo library, select a directory in your file system (the more the contents, the longer it will take) and connect it under the **BonusCalculator:/docs** directory.

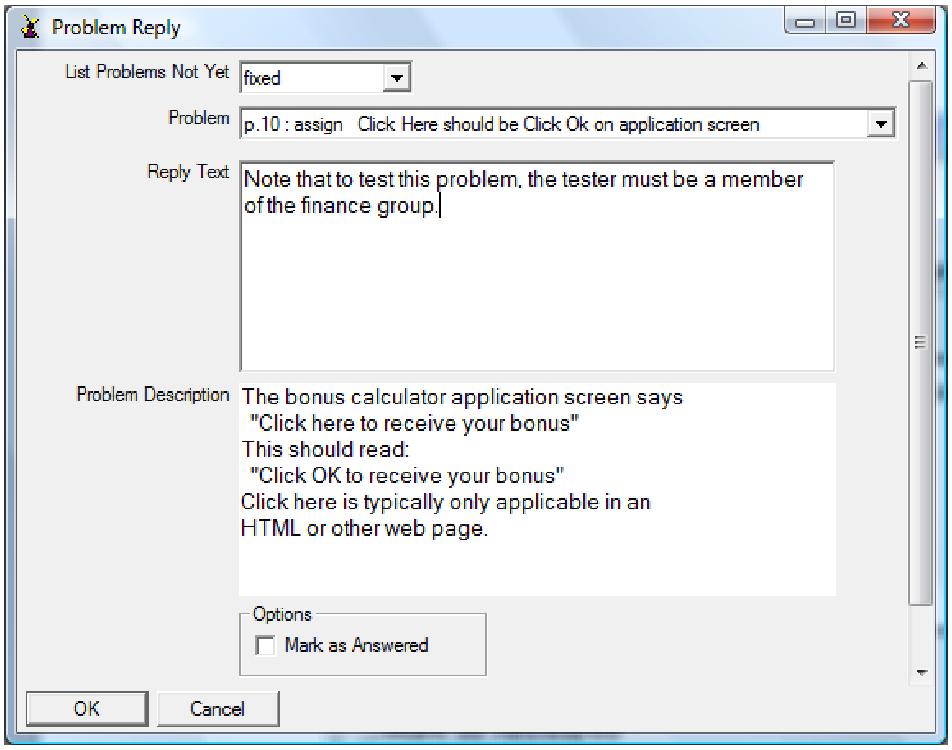


FileTreePath	C:\temp\sts\Sim
Product	BonusCalculator
Stream	rel2
ExcludeFiles	*.o *.obj demo.log*
ConnectToDirectory	BonusCalculator:/bonuscd/docs
Path Contents Only	☐

Replying to a Problem Report

Normally when you go through the fix cycle, problem reports will automatically be advanced in status and the description will get a date-stamped reply indicating the update that fixed the problem. However, there are other times when it is useful to add information to a problem report, perhaps to clarify it, specify a work-around, or to answer it without having to do a source fix.

To reply to a problem report, select **Reply to Problem…** from the **Problems** menu. Select the problem and enter the text for the reply. Select “Mark As Answered” only if the problem is not expected to have any other work done to resolve the problem.



List Problems Not Yet: fixed

Problem: p.10 : assign   Click Here should be Click Ok on application screen

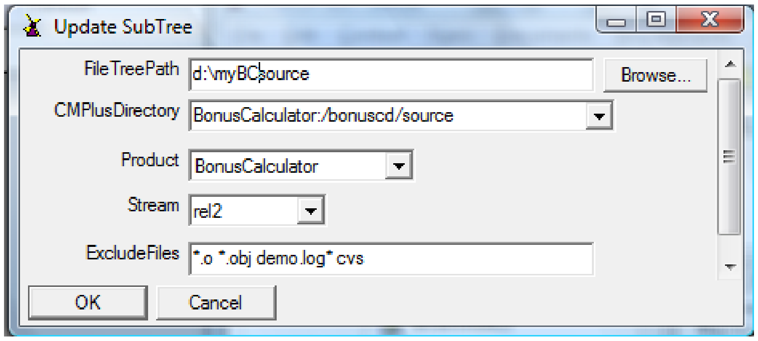
Reply Text: Note that to test this problem, the tester must be a member of the finance group.

Problem Description: The bonus calculator application screen says
 "Click here to receive your bonus"
This should read:
 "Click OK to receive your bonus"
Click here is typically only applicable in an HTML or other web page.

Options: [ ] Mark as Answered

Automatically Creating an Update from Workspace Differences

It is possible to ask CM+ to compare your workspace to your Product/Stream context, and have it automatically create an Update which reflects the changes in your workspace. For example, if you were on a trip and modified several files in your workspace, you might want to create an update reflecting these changes and to check them in on your return. Select **Update SubTree** from the **Updates/New** menu to do this. A panel such as the one below will appear for you to perform this task.



FileTreePath: d:\myBC\source

CMPlusDirectory: BonusCalculator:/bonuscd/source

Product: BonusCalculator

Stream: rel2

ExcludeFiles: *.o *.obj demo.log* cvs

Note that this is not a recommended normal mode for changes. When you work in this mode, others are not aware of the changes you are making, providing an overall poorer level of project communication.

When you have selected your file sub tree path (i.e. on your file system), and have matched it to a CM+ Directory, within the appropriate Product Stream, CM+ will attempt to create an update which identifies changed, or new files. It will NOT identify removed files or moved files. So if you only have some of the files from the sub-tree deployed, CM+ will assume that the other files have not chanegd (and not that they have been deleted).

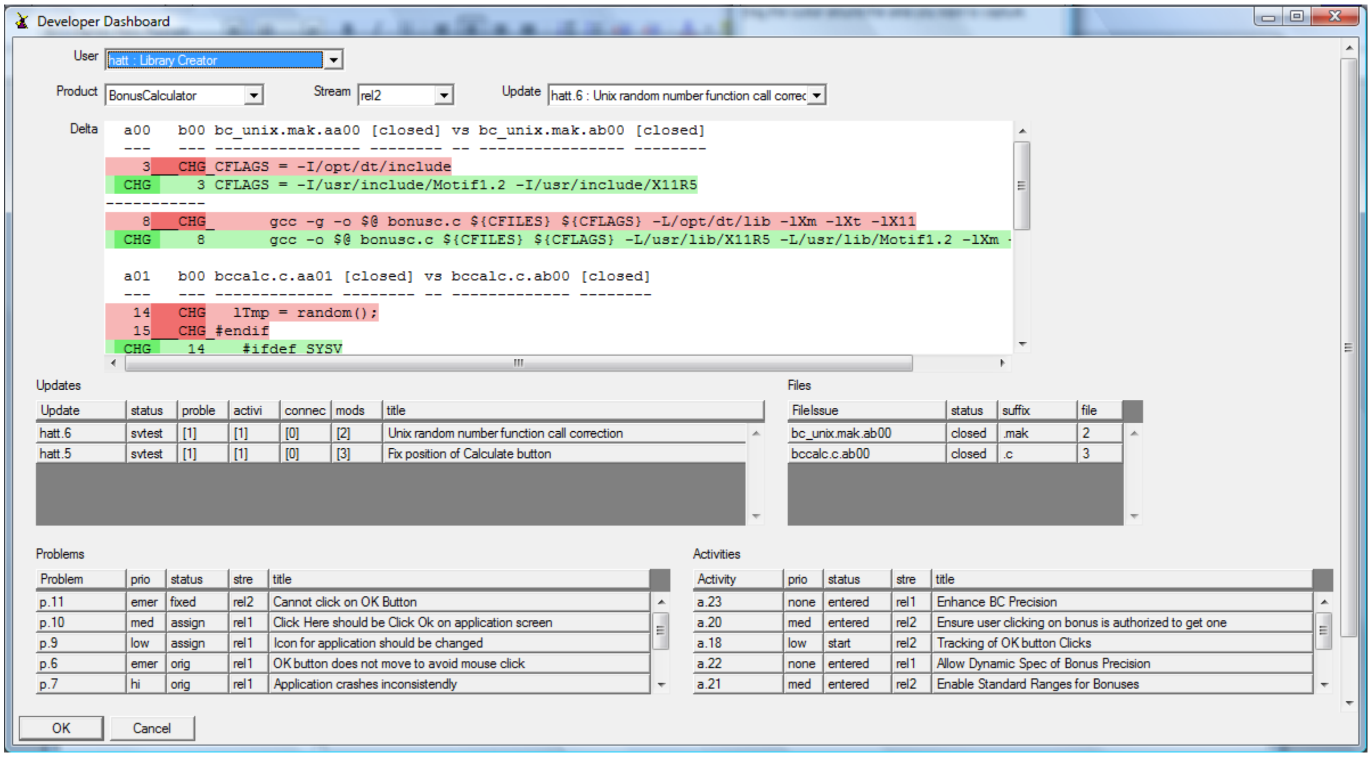
Synchronizing Your Workspace

You may need to synchronize your workspace from time to time to collect all changes that may have been submitted by other users. Do this by following a two step process: identifying the code differences and then selecting and retrieving the updated files. The first step is done by selecting **Code Differences** from the **Updates/Workspace** menu. Select the file suffix values which you wish to compare between your Workspace and the Current Context within the library and select OK. CM+ will compute a full delta report for all modified files with a summary at the end. Leave this window open for future reference. If there is only a specific part of the source tree you wish to compare, you may select that sub-tree in the Code Differences panel, or you may instead of using **Code Differences,** use the source tree right-click function **Compare to Workspace.** Either of these approaches will identify code that is different between your current context setting and your workspace.

Now select **Synchronize** from the **Updates/Workspace** menu and the files which had differences in your previous operation will appear. Select the files you do *NOT* wish to synchronize at this time. It is a good idea to review the Comparison panel produced earlier as you work through your selections. If you have files checked out for the current stream, these will likely be pre-checked so that you don't overwrite them in your workspace. If you accidentally overwrite a file, you will find backup copies in the workspace (typcially with an X after its name). After selecting OK, CM+ will present you with a confirmation list of the files to be synchronized. Click OK and the synchronization operation will pull the files from the library, replacing the revisions that were in your workspace.

You may customize your CM+ user interface to allow more complex synchronization processes, such as those which might allow you to merge differences between your current context and your workspace.

Developer Dashboard



User: hatt : Library Creator

Product: BonusCalculator   Stream: rel2   Update: hatt.6 : Unix random number function call correc

Delta

```
 a00   b00 bc_unix.mak.aa00 [closed] vs bc_unix.mak.ab00 [closed]
 ---   --- ---------------- -------- -- ---------------- --------
   3   CHG CFLAGS = -I/opt/dt/include
 CHG     3 CFLAGS = -I/usr/include/Motif1.2 -I/usr/include/X11R5
-----------
   8   CHG        gcc -g -o $@ bonusc.c ${CFILES} ${CFLAGS} -L/opt/dt/lib -lXm -lXt -lX11
 CHG     8        gcc -o $@ bonusc.c ${CFILES} ${CFLAGS} -L/usr/lib/X11R5 -L/usr/lib/Motif1.2 -lXm

 a01   b00 bccalc.c.aa01 [closed] vs bccalc.c.ab00 [closed]
 ---   --- ------------- -------- -- ------------- --------
  14   CHG    lTmp = random();
  15   CHG #endif
 CHG    14     #ifdef SYSV
```

Updates

Update	status	proble	activi	connec	mods	title
hatt.6	svtest	[1]	[1]	[0]	[2]	Unix random number function call correction
hatt.5	svtest	[1]	[1]	[0]	[3]	Fix position of Calculate button

Files

FileIssue	status	suffix	file
bc_unix.mak.ab00	closed	.mak	2
bccalc.c.ab00	closed	.c	3

Problems

Problem	prio	status	stre	title
p.11	emer	fixed	rel2	Cannot click on OK Button
p.10	med	assign	rel1	Click Here should be Click Ok on application screen
p.9	low	assign	rel1	Icon for application should be changed
p.6	emer	orig	rel1	OK button does not move to avoid mouse click
p.7	hi	orig	rel1	Application crashes inconsistendly

Activities

Activity	prio	status	stre	title
a.23	none	entered	rel1	Enhance BC Precision
a.20	med	entered	rel2	Ensure user clicking on bonus is authorized to get one
a.18	low	start	rel2	Tracking of OK button Clicks
a.22	none	entered	rel1	Allow Dynamic Spec of Bonus Precision
a.21	med	entered	rel2	Enable Standard Ranges for Bonuses

CM+ includes a number of default dashboards. The Developer Dashboard allows you to scroll through a user's changes, and to look at the changes performed, delta reports, problems fixed and activities addressed. All dashboards may be customized to suite a project's requirements. New dashboards may also be added. To access the dashboard, roll the mouse over the toolbar icons until the Developer Dashboard is revealed by a tool tip.

Select the user "hatt" for this dashboard and feel free to scroll through the updates, or zoom into the details of any of the records in the display panels found on the dashboard.

WorkSpace Status Dashboard

This section is not part of the demo tutorial, but appears for information purposes only.

The WorkSpace Status Dashboard is used to help analyze the content and state of a user's workspace or a portion thereof. First available with CM+ 7, the dashboard will have variations that evolve both over time and across projects. The Workspace holds the source code used for "local" builds, as well as for the edit-compile-test cycles which accompany such builds. Some terminology:

Source Code:	A file which is generally compiled or otherwise used as part of a run-time or build environment.
Workspace:	The root directory, and sub-directories, containing source code required for a product builds.
Checkout:	Registering a file in CM+ for a change, with an option of retrieving the file to the workspace.
Get:	A retrieval operation to place source code in the workspace.
Checkin:	The submission to the CM+ repository of an Update containing previously Checked Out files.
Context:	The "current" view of repository files in CM+, as specified by the user.
Synchronize:	The operation of bringing the workspace in line with the context view.

[Note: An important distinction needs to be clarified because of the different common usages of the word "Checkout". In CM+, a Checkout operation registers a file for change (i.e. against an Update), typically reserving that branch of the file until the changes are Checked In. The term used for retrieving source code to the Workspace is "Get" or "Get Source". A Checkout operation may or may not indicate that source code is to be retrieved.]

Workspace Status Operation

A CM+ user may right-click on any directory in the source tree and select "Workspace Status". This will bring up a dashboard that outlines the various states of the files in that part of the source tree. The states are identified as follows:

Context:	Workspace file matches (date and code) in current CM+ context.
Missing:	File in the CM+ context is missing from the Workspace.
OldRev:	Workspace file matches an older revision than the one in the CM+ context.
Later:	Workspace file matches a later revision than the one in the CM+ context.
Modified:	Workspace file differs from the CM+ context, with changes on top of the context revision.
Merge:	Workspace file differs from the CM+ context, and was derived from an earlier revision.
Unknown:	CM+ does not know the origin of the file in the Workspace.

The count of the number of files in each of these states is displayed in the dashboard:

These are the various fields of the Workspace Status dashboard:

Workspace: Your current workspace setting, to which this is being applied. (Restart this dashboard if you change your workspace). This is a read-only field for information purposes only.

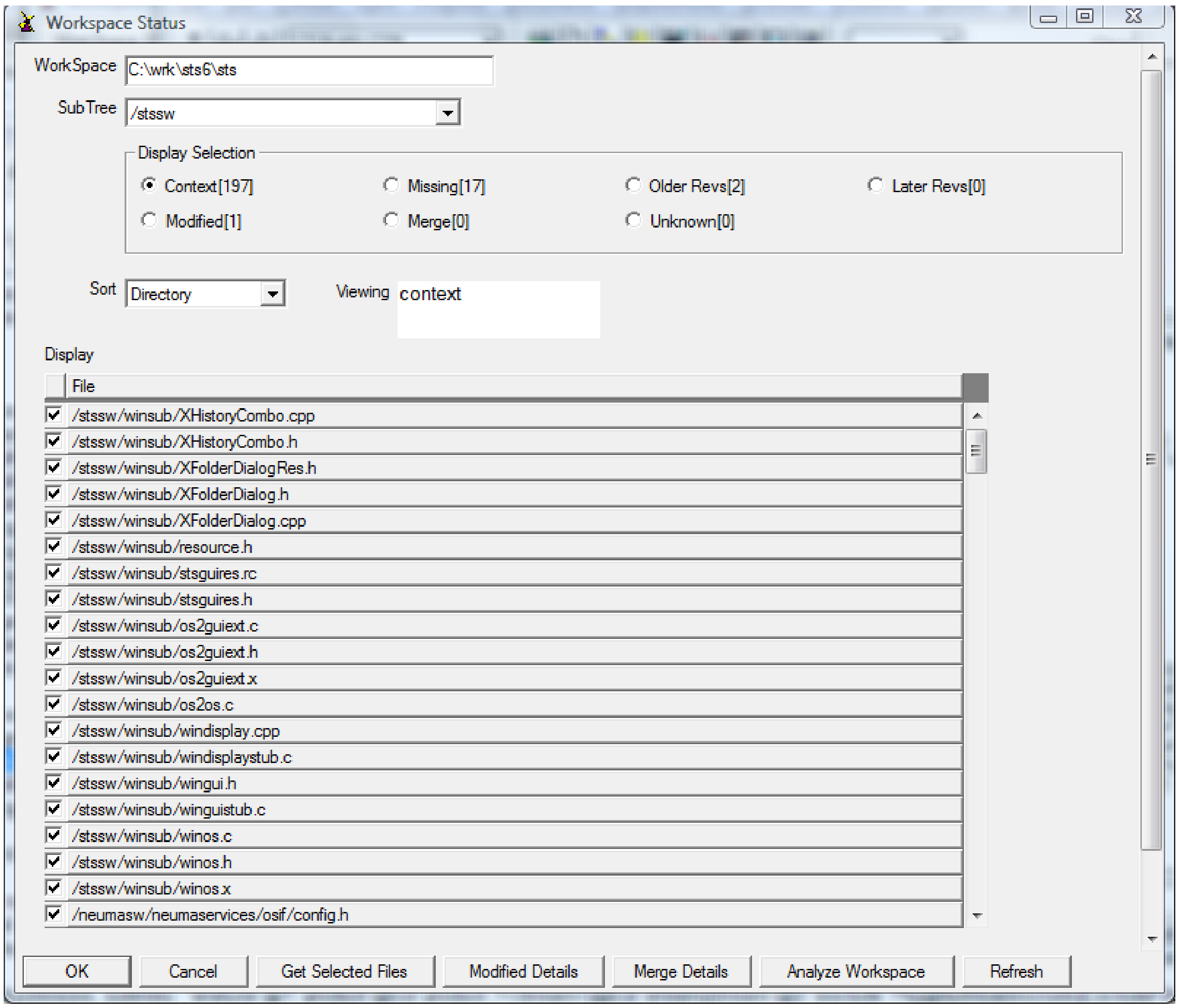
SubTree: The sub-tree of your product source tree on which the workspace status operation is being applied. This may be the entire product tree, but in larger projects, this may result in some significant delays. If you change your sub-tree selection, you must select Refresh to have the information updated.

Display Selection: This selects the workspace status which will be displayed in the Display area. Whenever you switch workspace status, all of the files with that status will be selected (for use in Get operations).

Sort: The Display of files is sorted by the Directory tree (same order as in the tree), by filename, or by file type.

Viewing: (currently not used - information purposes only).

Display: This is the set of files in the context tree whose corresponding workspace status matches the Display Selection. The "Get Selected Files" button works on the displayed set of files which have check marks beside them.


Workspace Status

WorkSpace: C:\wrk\sts6\sts

SubTree: /stssw

Display Selection: [● Context[197]] Missing[17] Older Revs[2] Later Revs[0] Modified[1] Merge[0] Unknown[0]

Sort: Directory   Viewing: context

Display

	File
✓	/stssw/winsub/XHistoryCombo.cpp
✓	/stssw/winsub/XHistoryCombo.h
✓	/stssw/winsub/XFolderDialogRes.h
✓	/stssw/winsub/XFolderDialog.h
✓	/stssw/winsub/XFolderDialog.cpp
✓	/stssw/winsub/resource.h
✓	/stssw/winsub/stsguires.rc
✓	/stssw/winsub/stsguires.h
✓	/stssw/winsub/os2guiext.c
✓	/stssw/winsub/os2guiext.h
✓	/stssw/winsub/os2guiext.x
✓	/stssw/winsub/os2os.c
✓	/stssw/winsub/windisplay.cpp
✓	/stssw/winsub/windisplaystub.c
✓	/stssw/winsub/wingui.h
✓	/stssw/winsub/winguistub.c
✓	/stssw/winsub/winos.c
✓	/stssw/winsub/winos.h
✓	/stssw/winsub/winos.x
✓	/neumasw/neumaservices/osif/config.h

Action Buttons: There are a number of action buttons which may evolve or be customized over time.

OK - Exit the dialog

Cancel - Exit the dialog (not really different from OK - included for GUI consistency)

Get Selected Files - Dialog for retrieving the currently selected files in the Display area (i.e. same status).

Modified Details - Dialog showing the Code vs. Meta-Data differences for files of status "modified"

Merge Details - Dialog showing the various partitions of "merge" files (Meta Data and Code differences)

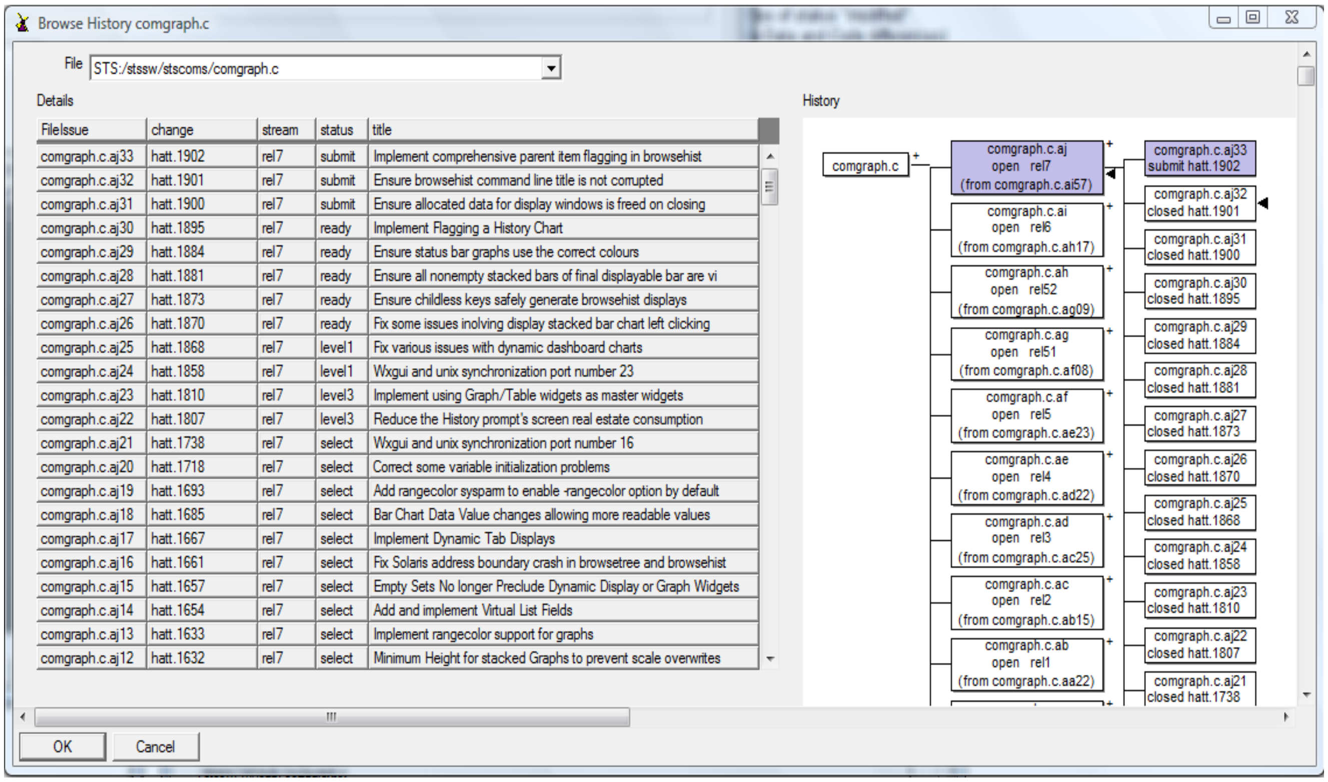
Analyze Workspace - Dialog to identify New (not in CM+) files with options to perform on the workspace.

Refresh - Refreshes the dashboard to take into account workspace, repository or Sub-Tree selection changes.

Note that generally, actions taken in other dialogs may require a Refresh operation before they are visible on the Workspace Dashboard.

As well as the array of buttons at the bottom of the dashboard, there are also right-click (context menu) operations that are available to further analyze the workspace files. These include "Browse History", "Get Source" and "Compare to Workspace" functions, which may be executed on individual files by right-clicking on them.

The "Browse History" operation will bring up a dashboard which shows an interactive display of file revisions in a scrollable panel, as well as a history chart. On the history chart, you will see the File, the list of Branches, and the list of Revisions in the currently open Branch. The shading of a single Branch and Revision, indicates that they correspond to the current context setting within the user's CM+ view. The arrows, if present, indicate the Branch/Revision which may be found in the workspace. In some cases (modified and merged files), subsequent modifications may have been made to the workspace file. So, in this case, the arrows represent the revision from which these modifications were started.



Browse History comgraph.c
File
Details
History
comgraph.c
comgraph.c.aj open rel7 (from comgraph.c.ai57)
comgraph.c.ai open rel6 (from comgraph.c.ah17)
comgraph.c.ah open rel52 (from comgraph.c.ag09)
comgraph.c.ag open rel51 (from comgraph.c.af08)
comgraph.c.af open rel5 (from comgraph.c.ae23)
comgraph.c.ae open rel4 (from comgraph.c.ad22)
comgraph.c.ad open rel3 (from comgraph.c.ac25)
comgraph.c.ac open rel2 (from comgraph.c.ab15)
comgraph.c.ab open rel1 (from comgraph.c.aa22)
comgraph.c.aj33 submit hatt.1902
comgraph.c.aj32 closed hatt.1901
comgraph.c.aj31 closed hatt.1900
comgraph.c.aj30 closed hatt.1895
comgraph.c.aj29 closed hatt.1884
comgraph.c.aj28 closed hatt.1881
comgraph.c.aj27 closed hatt.1873
comgraph.c.aj26 closed hatt.1870
comgraph.c.aj25 closed hatt.1868
comgraph.c.aj24 closed hatt.1858
comgraph.c.aj23 closed hatt.1810
comgraph.c.aj22 closed hatt.1807
comgraph.c.aj21 closed hatt.1738
OK
Cancel

File: STS:/stssw/stscoms/comgraph.c

FileIssue	change	stream	status	title
comgraph.c.aj33	hatt.1902	rel7	submit	Implement comprehensive parent item flagging in browsehist
comgraph.c.aj32	hatt.1901	rel7	submit	Ensure browsehist command line title is not corrupted
comgraph.c.aj31	hatt.1900	rel7	submit	Ensure allocated data for display windows is freed on closing
comgraph.c.aj30	hatt.1895	rel7	ready	Implement Flagging a History Chart
comgraph.c.aj29	hatt.1884	rel7	ready	Ensure status bar graphs use the correct colours
comgraph.c.aj28	hatt.1881	rel7	ready	Ensure all nonempty stacked bars of final displayable bar are vi
comgraph.c.aj27	hatt.1873	rel7	ready	Ensure childless keys safely generate browsehist displays
comgraph.c.aj26	hatt.1870	rel7	ready	Fix some issues inolving display stacked bar chart left clicking
comgraph.c.aj25	hatt.1868	rel7	level1	Fix various issues with dynamic dashboard charts
comgraph.c.aj24	hatt.1858	rel7	level1	Wxgui and unix synchronization port number 23
comgraph.c.aj23	hatt.1810	rel7	level3	Implement using Graph/Table widgets as master widgets
comgraph.c.aj22	hatt.1807	rel7	level3	Reduce the History prompt's screen real estate consumption
comgraph.c.aj21	hatt.1738	rel7	select	Wxgui and unix synchronization port number 16
comgraph.c.aj20	hatt.1718	rel7	select	Correct some variable initialization problems
comgraph.c.aj19	hatt.1693	rel7	select	Add rangecolor syspam to enable -rangecolor option by default
comgraph.c.aj18	hatt.1685	rel7	select	Bar Chart Data Value changes allowing more readable values
comgraph.c.aj17	hatt.1667	rel7	select	Implement Dynamic Tab Displays
comgraph.c.aj16	hatt.1661	rel7	select	Fix Solaris address boundary crash in browsetree and browsehist
comgraph.c.aj15	hatt.1657	rel7	select	Empty Sets No longer Preclude Dynamic Display or Graph Widgets
comgraph.c.aj14	hatt.1654	rel7	select	Add and implement Virtual List Fields
comgraph.c.aj13	hatt.1633	rel7	select	Implement rangecolor support for graphs
comgraph.c.aj12	hatt.1632	rel7	select	Minimum Height for stacked Graphs to prevent scale overwrites

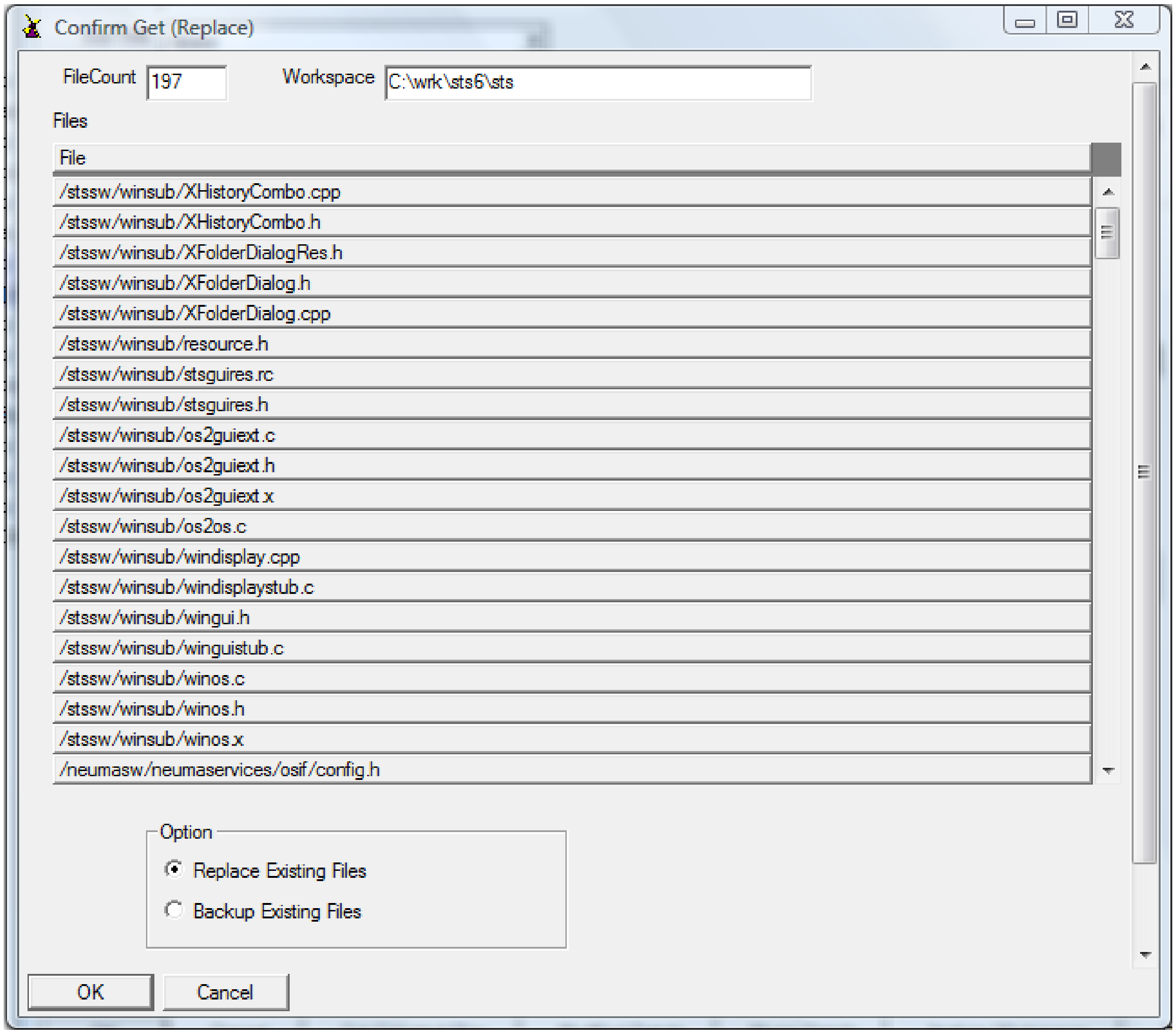
Note that both the Details panel and the History chart are interactive. You may right-click on any item and select one of the operations to be performed. To compare two arbitrary revisions, right-click on the "old" revision first and "Select Delta File". Then right-click the "new" revision and "Delta vs. Selection".

Get Selected Files

When the "Get Selected Files" button is clicked, a dialog appears.

This is a confirmation dialog which also allows you to indicate what happens to existing files. The File Count indicates the number of files that were selected at the time you clicked the "Get Selected Files" button. The Workspace identifies the root directoy of the workspace to which these files will be retrieved. The list of Files are shown in a scrollable list. If you wish to change this list, return to the previous dialog first, by selecting Cancel. The Option field allows you to select between overwriting (i.e. replacing) existing files, and backing up existing files by appending the "backup" character to them (typically "X"). By default, two backups (X and XX) will be kept. When you click OK, the retrieval will begin immediately.

The “Get Selected Files” dialog is shown below.


Confirm Get (Replace)

FileCount 197 Workspace C:\wrk\sts6\sts

Files

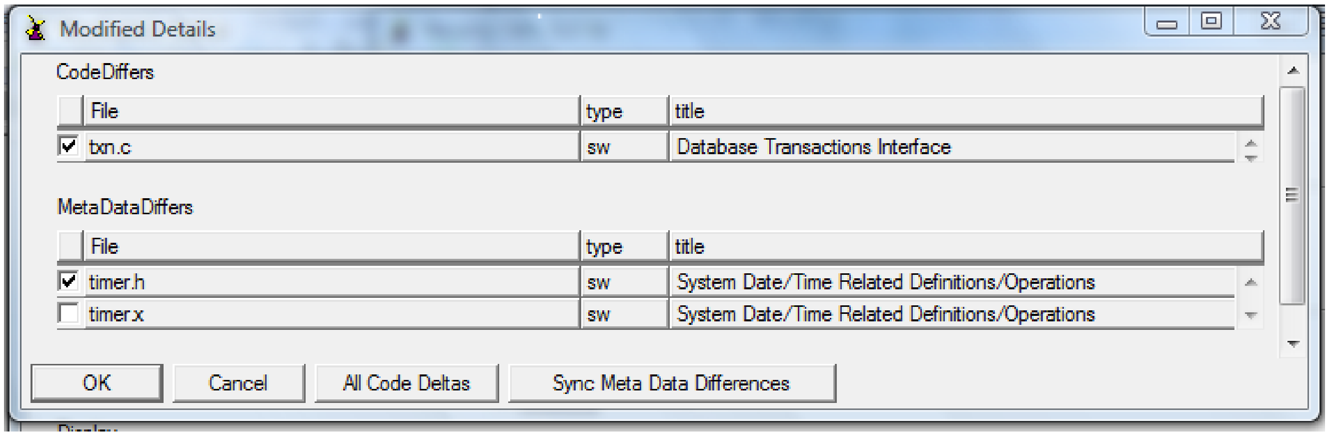
File
/stssw/winsub/XHistoryCombo.cpp
/stssw/winsub/XHistoryCombo.h
/stssw/winsub/XFolderDialogRes.h
/stssw/winsub/XFolderDialog.h
/stssw/winsub/XFolderDialog.cpp
/stssw/winsub/resource.h
/stssw/winsub/stsguires.rc
/stssw/winsub/stsguires.h
/stssw/winsub/os2guiext.c
/stssw/winsub/os2guiext.h
/stssw/winsub/os2guiext.x
/stssw/winsub/os2os.c
/stssw/winsub/windisplay.cpp
/stssw/winsub/windisplaystub.c
/stssw/winsub/wingui.h
/stssw/winsub/winguistub.c
/stssw/winsub/winos.c
/stssw/winsub/winos.h
/stssw/winsub/winos.x
/neumasw/neumaservices/osif/config.h

Option

● Replace Existing Files
○ Backup Existing Files

Modified Details

When you select the "Modified Details" button, a dialog appears showing which files differ in Code, and which differ only in Meta Data (such as the date stamp, or the identification information).



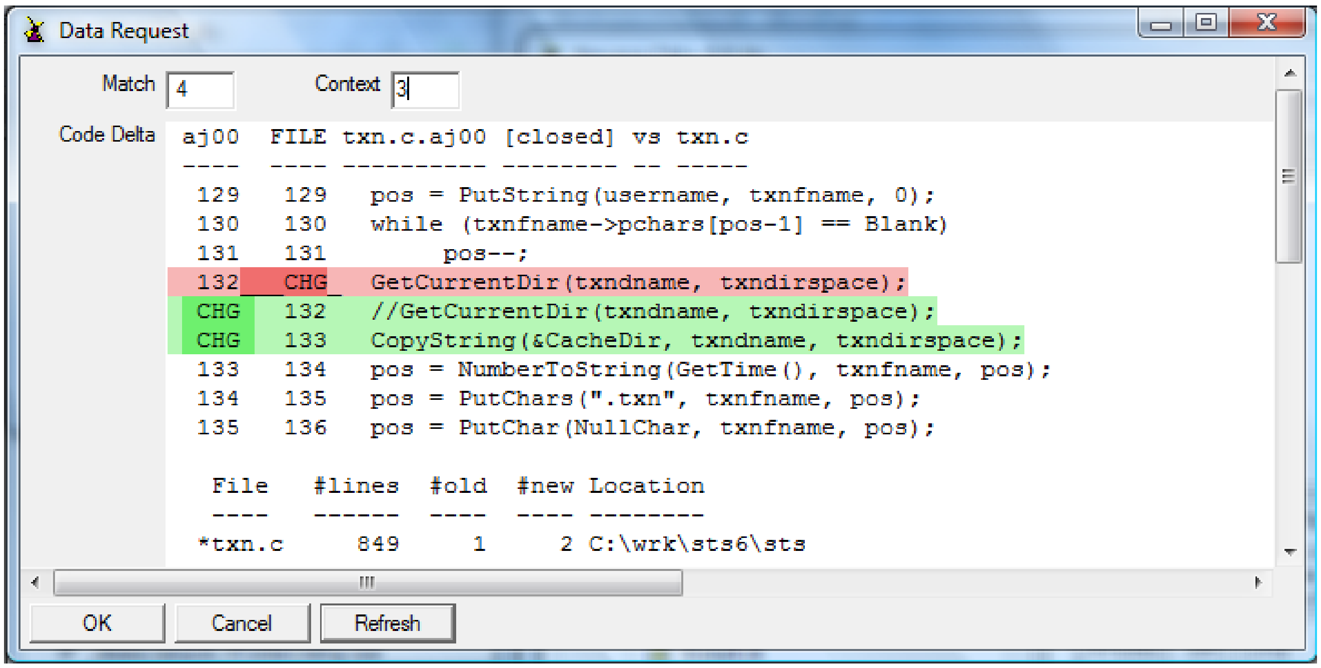
CodeDiffers

	File	type	title
✓	txn.c	sw	Database Transactions Interface

MetaDataDiffers

	File	type	title
✓	timer.h	sw	System Date/Time Related Definitions/Operations
	timer.x	sw	System Date/Time Related Definitions/Operations

The files in the "CodeDiffers" list have actual source code differences between the Repository Context revision and the Workspace revision. You may right-click and select Compare to Workspace for an individual file, or you may view the code deltas (i.e. differences) for all of the files by selecting the "All Code Deltas" button. The following dialog then appears.



Match 4 Context 3

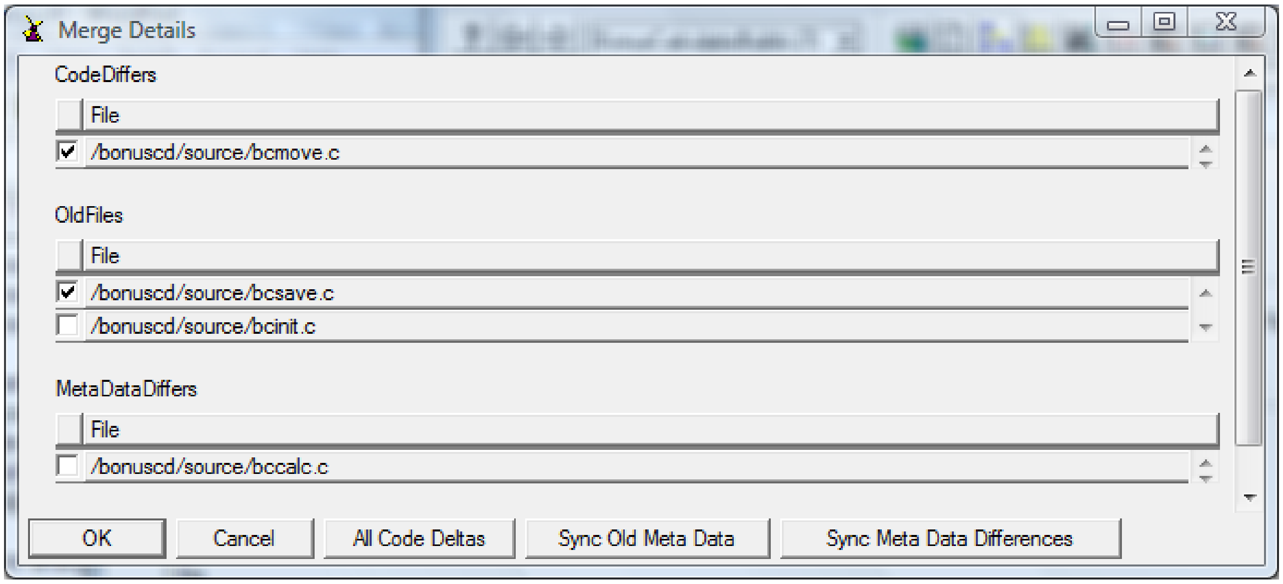
Code Delta
```
aj00  FILE txn.c.aj00 [closed] vs txn.c
----  ---- ---------- -------- -- -----
 129   129   pos = PutString(username, txnfname, 0);
 130   130   while (txnfname->pchars[pos-1] == Blank)
 131   131        pos--;
 132   CHG   GetCurrentDir(txndname, txndirspace);
 CHG   132   //GetCurrentDir(txndname, txndirspace);
 CHG   133   CopyString(&CacheDir, txndname, txndirspace);
 133   134   pos = NumberToString(GetTime(), txnfname, pos);
 134   135   pos = PutChars(".txn", txnfname, pos);
 135   136   pos = PutChar(NullChar, txnfname, pos);

  File    #lines  #old  #new Location
  ----    ------  ----  ---- --------
 *txn.c      849     1     2 C:\wrk\sts6\sts
```

This will show the set of file deltas for all files and will give a summary at the end. By default, deletions will be shown in red and additions in green. You may adjust the number of context lines or the number of lines that must match prior to a change being considered a separate change block. When you do, however, CM+ will recompute the delta report. In the case a large or a large number of files, this may take a few seconds.

If you select the "Sync Meta Data Differences", CM+ will perform a "Get" operation by bringing up the same dialog as it does when you click "Get Selected Files" in the workspace status dashboard. Again, the operation will be restricted only to those files that have been selected in the "MetaDataDiffers" list of the Modified Details dialog. Files which actually differ in code will NOT be retrieved by the Sync operation. You may do so (and overwrite your modifications) individually using a right-click "Get Source" operation, or in batch from the workspace status dashboard, using the "Get Selected Files" button. Note that you may also perform individual Deltas by right-clicking on a file and selecting "Compare to Workspace".

Merge Details

Workspace files with a workspace status of merge are different both from the current context view in CM+ and from earlier revisions. The differences may be due to code changes, or to meta data (e.g. date stamp, identification info). The Merge Details button launches a "Merge Details" panel which further analyzes these files to partition them into one of three groups: a valid merge may be required; the file matches an Older Revision even though the meta-data indicates it may have changed; the file matches the Current Context Revision even though the meta-data indicates it may have changed.



CodeDiffers

	File
✓	/bonuscd/source/bcmove.c

OldFiles

	File
✓	/bonuscd/source/bcsave.c
☐	/bonuscd/source/bcinit.c

MetaDataDiffers

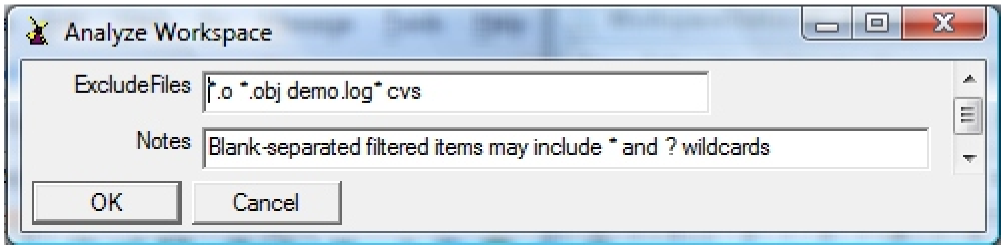
	File
☐	/bonuscd/source/bccalc.c

The "All Code Deltas" button presents a code delta report for all of the files in the "CodeDiffers" list, similar to the one for Modified Details, above. The "Sync Old Meta Deta" button will replace files which match Old Revisions (i.e. those in the "Old Files list), with those actual file revisions so that the meta data now matches. The "Sync Meta Data Differences" button will replace files in the MetaDataDiffers list, with current context revisions of those files, to ensure that the meta data differences disappear.

Right-click on a file in the CodeDiffers list should reveal a Workspace Merge operation. This will merge the current context with the file in the user's directory. If there is a common ancestor, a three-way merge is performed, otherwise, a two-way merge is done.

Analyze Workspace

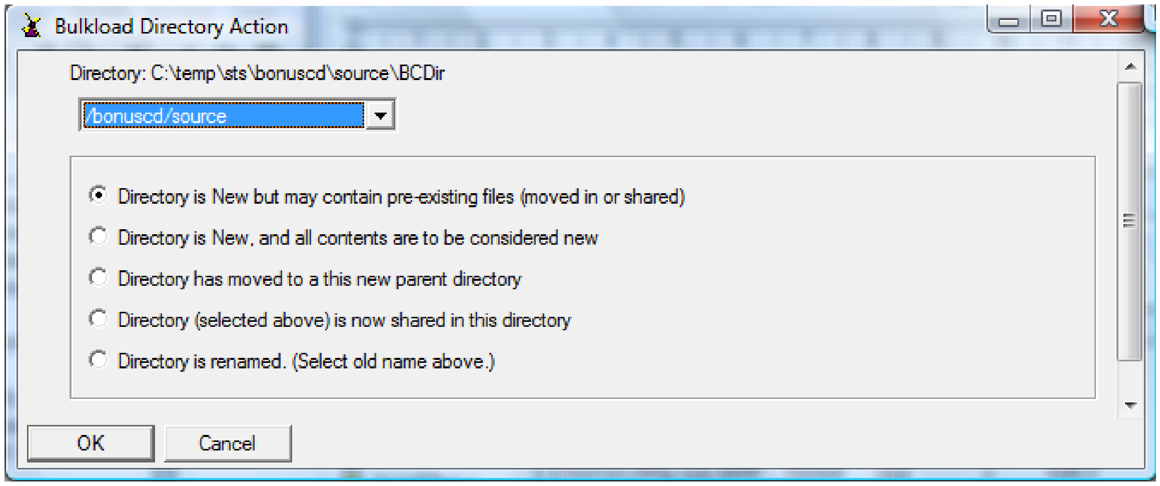
When you analyze a workspace, CM+ goes through the workspace and compares the files it finds with those in your current context. It ignores files which match the list of excluded files, and otherwise identifies new files and directories, as well as files that match the context source tree.



ExcludeFiles: *.o *.obj demo.log* cvs

Notes: Blank-separated filtered items may include * and ? wildcards

The analysis is performed after presenting a dialog panel for specifying the list of excluded files. If you use your workspace directory as your build directory, you might have exclusions such as *.exe and *.dll. If you allow backups, either through CM+ or through an editor, you may have exclusions such as *X and *%, depending on what backup characters are used.

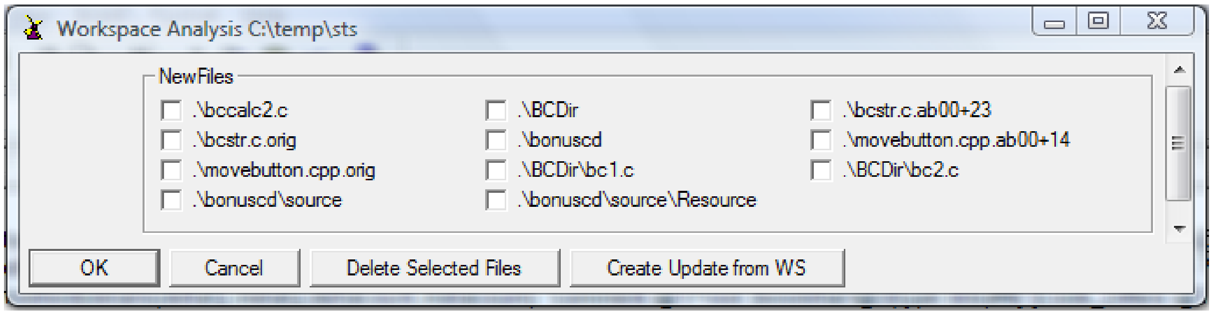


Directory: C:\temp\sts\bonuscd\source\BCDir

/bonuscd/source

- [✓] Directory is New but may contain pre-existing files (moved in or shared)
- [ ] Directory is New, and all contents are to be considered new
- [ ] Directory has moved to a this new parent directory
- [ ] Directory (selected above) is now shared in this directory
- [ ] Directory is renamed. (Select old name above.)

As CM+ analyzes the workspace, it may come across directories that don't match the context source tree. In such cases it will ask you how to treat them. Perhaps you've created a new directory and moved some files into it. Or perhaps you've renamed a directory. By indicating what you've done, CM+ can match files from the source tree to the workspace. Otherwise, it has to assume that the files found in "new" directores are themselves new files with no prior history. The top line of the Action panel indicates the item that is causing CM+ to prompt for information. The selector may be used with some of the responses, from which you select one, but is otherwise ignored.



NewFiles

- [ ] .\bccalc2.c
- [ ] .\bcstr.c.orig
- [ ] .\movebutton.cpp.orig
- [ ] .\bonuscd\source
- [ ] .\BCDir
- [ ] .\bonuscd
- [ ] .\BCDir\bc1.c
- [ ] .\bonuscd\source\Resource
- [ ] .\bcstr.c.ab00+23
- [ ] .\movebutton.cpp.ab00+14
- [ ] .\BCDir\bc2.c

On completion, CM+ identifies the set of new files and directories. You may select files to be deleted from the workspace. You may also select directories, if empty. CM+ will not update the display until a subsequent analysis is requested. CM+ also gives you the option of creating an update based on the differences it finds in the workspace.   This assumes that you have first reconciled your workspace to your context. Otherwise, you may find some unintended files as part of your Update. In such a case, you will need to perform a Cancel Checkout operation.

Neuma recommends that you check out (i.e. register against an Update) files as you need to change them. This both helps with the granularity of your traceability, and makes it easier for you and others to track your Update. The **Create Update from Workspace** operation is a tool that can be easy to use if you work on sub-trees that are wholly "owned" by you, so that the workspace sub-tree does not need constant re-basing (i.e. synchronization with other changes). Prior to using it on a project repository, get a feel for it by using it on a testbed, or on a copy of your production repository.

CONFIGURATION MANAGER ROLE

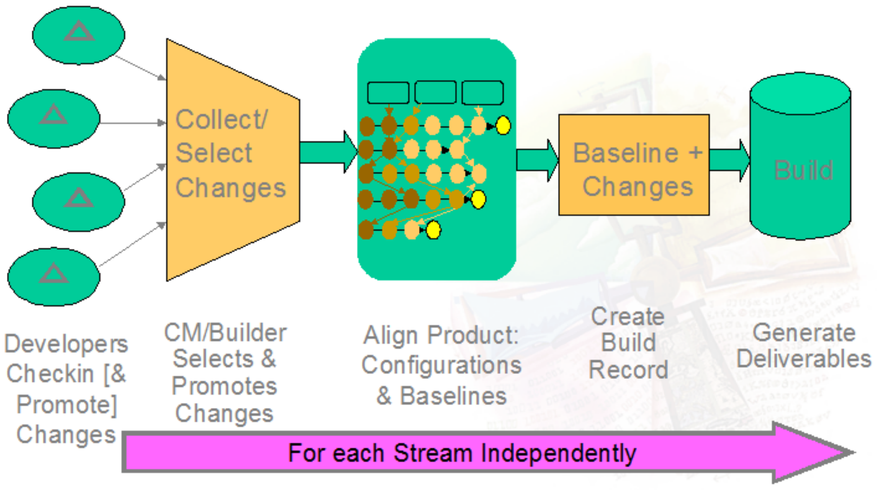
CM Manager Overview

The demonstration continues with the CM Manager role. The CM Manager will review the set of updates that have been readied by the development team, then prepare a Build for System Integration testing.

CM+ tracks Builds as first-order artifacts, with their own attributes, descriptions and state-based process flow. Build management lets you track and manage an exact configuration of your software without relying on branching or labeling.

A build consists of a baseline plus a number of arbitrarily selected updates. A baseline is a frozen snapshot of a specific set of file revisions, whereas an update applies specific changes to that frozen set. In some cases an update serves only to move files around in the tree. CM+ uses this combination of a baseline plus updates to determine an exact configuration of your software. The CM Manager's job is simply to approve changes, whereas CM+ handles the difficult task of figuring out the product configuration.

With CM+ Build tracking not only can you manage releases but you can also manage patches, variants, customization and nightly builds. In this tutorial, you will create a baseline and a new build record based solely on the baseline. You will also compare this build against a previous build defined in the system.

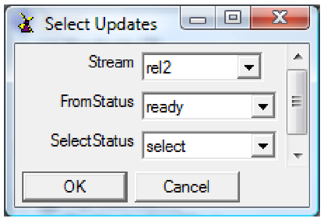

Collect/ Select Changes
Baseline + Changes
Build
Developers Checkin [& Promote] Changes
CM/Builder Selects & Promotes Changes
Align Product: Configurations & Baselines
Create Build Record
Generate Deliverables
For each Stream Independently

The above diagram illustrates the general build cycle that a CM manager moves through. This cycle applies to each iteration of each release stream for each product. In CM+ the Changes are defined as Updates.

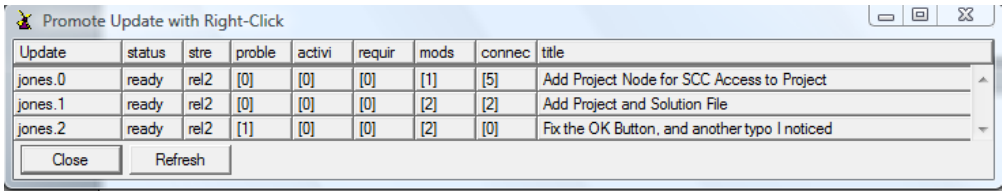
For this part of the tutorial, enter the demo library as user "black", our CM Manager. Select the **CM+ Demo CM Manager** from the **CM+** menu under **Start/Programs**. On UNIX, specify "-u black" when you start CM+: **sts demo -g -u black**

Selecting Updates for a build

First the CM manager will review the updates that have been marked ready by the development team. To identify the ready updates, choose the **Select Updates** function from the **Builds** menu. In the resulting dialog, select rel2 as the Stream, ready as the FromStatus and select as the SelectStatus, then click OK. In the next dialog, right click on jones.0, .1 and .2 to advance them to "select". Because the status is being promoted, CM+ verifies that there are no dependent updates which have not yet been promoted.



Stream: rel2
FromStatus: ready
SelectStatus: select


Promote Update with Right-Click

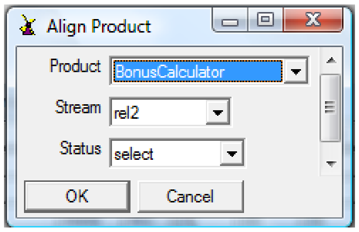
Update	status	stre	proble	activi	requir	mods	connec	title
jones.0	ready	rel2	[0]	[0]	[0]	[1]	[5]	Add Project Node for SCC Access to Project
jones.1	ready	rel2	[0]	[0]	[0]	[2]	[2]	Add Project and Solution File
jones.2	ready	rel2	[1]	[0]	[0]	[2]	[0]	Fix the OK Button, and another typo I noticed

Backing Out Updates

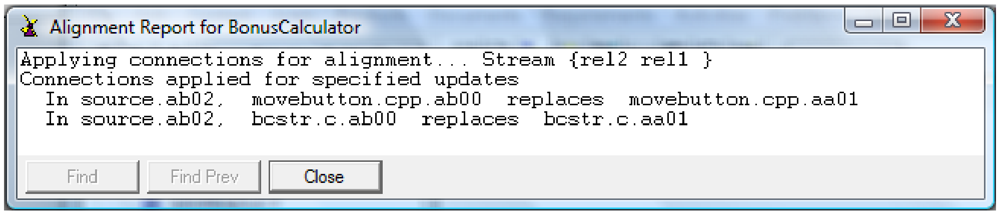
When an update is promoted, only to later cause problems in a build, or when an update has dependent updates which cannot be promoted at this time, it may be necessary to demote an update. To do this, select **Roll Back Updates** from the **Builds** menu. Right click on an update to roll it back. In this tutorial, we do not need to roll back any updates.

Defining a New Product Configuration

Once the updates are selected, the next step is to *align* the development stream to include the selected updates. The align operation tells CM+ to determine the next product configuration based on the changes you have selected. To do this, select **Align Product** from the **Builds** menu. In the resulting dialog, select the BonusCalculator product, the rel2 stream and the select status and click OK. The product will be aligned to include all changes in the current stream which are *at least at* the status select.


Align Product

Product: BonusCalculator
Stream: rel2
Status: select

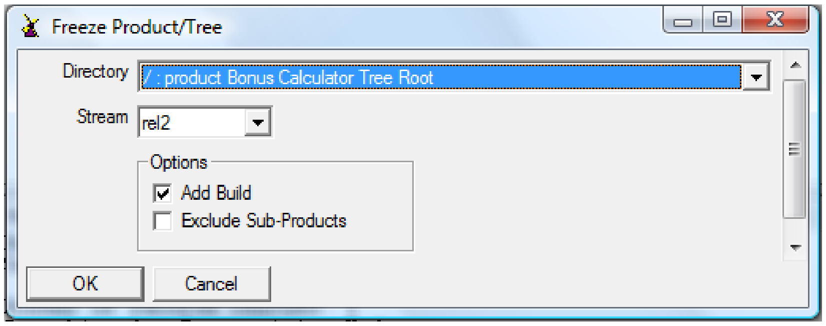

Alignment Report for BonusCalculator

```
Applying connections for alignment... Stream {rel2 rel1 }
Connections applied for specified updates
  In source.ab02,  movebutton.cpp.ab00  replaces  movebutton.cpp.aa01
  In source.ab02,  bcstr.c.ab00  replaces  bcstr.c.aa01
```

The align operation creates a new, open configuration, or if an open configuration already exists, it changes the configuration to include new revisions. In our example, the open configuration already exists, so the parent node, source.ab00 is simply changed to include a revised child node, movebutton.cpp.ab00. Changes made in any of the selected updates may cause additional replacements in the open configuration. When an align is done against a frozen configuration (i.e. a baseline), you will always see new revisions of directory nodes being created and replaced in the configuration. The configuration may be aligned multiple times, after selecting additional updates and/or rolling some back.

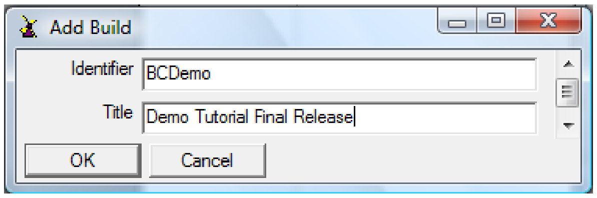
Freezing the Product Alignment into a Baseline

Once an align operation has been done, it is possible to freeze the configuration, thereby establishing a new baseline. Do this for the Bonus Calculator by selecting **Freeze Lineup** from the **Builds** menu. It is possible to freeze only a sub-tree of the product, but in this demo we will freeze the whole tree. Leave the Add Build Record option checked to allow the creation of a build record corresponding to the baseline.



Directory: / : product Bonus Calculator Tree Root
Stream: rel2
Options
[✓] Add Build
[ ] Exclude Sub-Products

Strictly speaking, the baseline is itself a permanent record, with all enumerated file revisions being accessible through the root level, frozen directory revision. However, a build record gives you the advantage of managing the workflow associated with the baseline as well as tracking attributes such as build type and description. When you click OK, you will be prompted for an Identifier (an alphanumeric name with no blanks or special characters) and a Title (brief description). Fill these in and click OK to create the build record (use a simple identifier such as BCDemo).



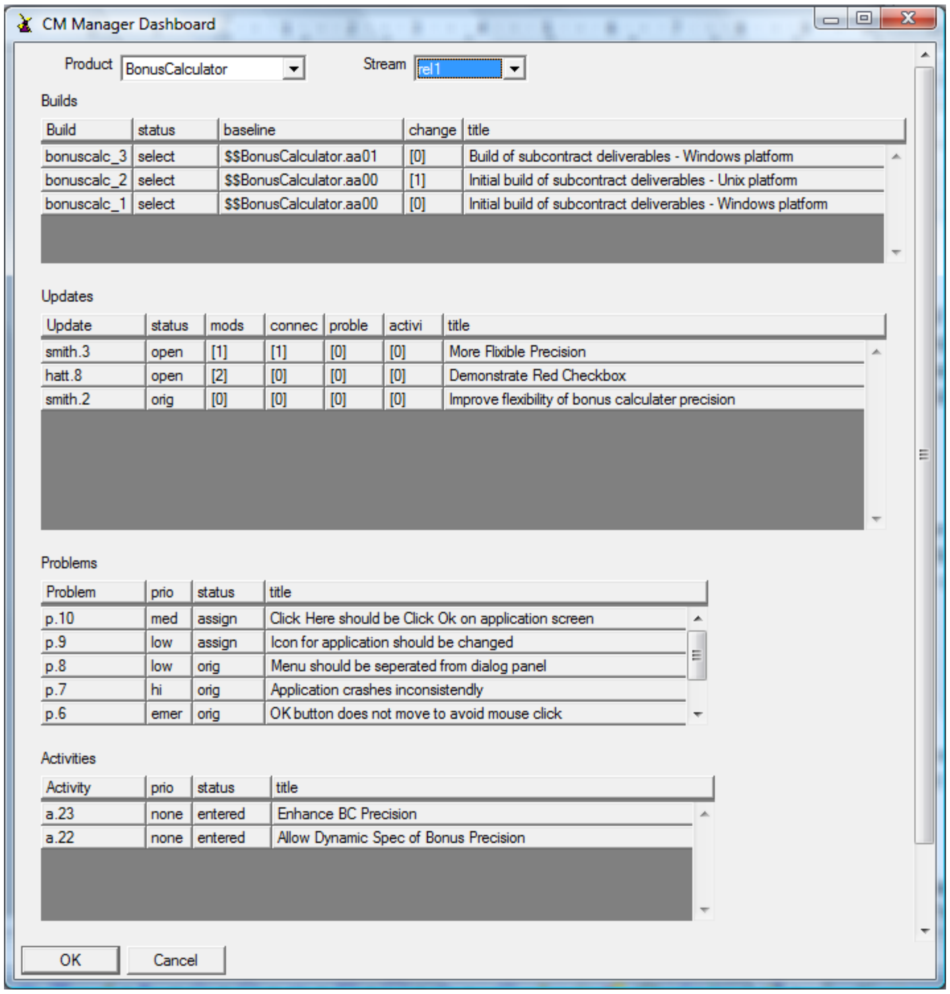
Identifier: BCDemo
Title: Demo Tutorial Final Release

Other Reasons for Creating a Build Record

The Freeze Lineup operation is just one case where a build can be created, and is used primarily when a major release is being generated. More often, for internal builds, a build record is created from an existing frozen baseline plus a list of updates. The list of updates is completely arbitrary and depends on your change management practices and the purpose of the build (nightly build, variant, patch release etc.) For more information see Registering a Build, in the Baselines, Releases and Builds section of the How-to Documentation.

CM Manager Dashboard

The CM Manager Dashboard can be accessed from the Toolbar by running the cursor over the icons until the CM Manager Dashboard tool tip pops up. Selecting the icon will bring up a dashboard which can be used to look at a number of current work for a particular stream of a specified product. The product and stream can be modified to switch between contexts for the dashboard.



Product: BonusCalculator   Stream: rel1

Builds

Build	status	baseline	change	title
bonuscalc_3	select	$$BonusCalculator.aa01	[0]	Build of subcontract deliverables - Windows platform
bonuscalc_2	select	$$BonusCalculator.aa00	[1]	Initial build of subcontract deliverables - Unix platform
bonuscalc_1	select	$$BonusCalculator.aa00	[0]	Initial build of subcontract deliverables - Windows platform

Updates

Update	status	mods	connec	proble	activi	title
smith.3	open	[1]	[1]	[0]	[0]	More Flxible Precision
hatt.8	open	[2]	[0]	[0]	[0]	Demonstrate Red Checkbox
smith.2	orig	[0]	[0]	[0]	[0]	Improve flexibility of bonus calculater precision

Problems

Problem	prio	status	title
p.10	med	assign	Click Here should be Click Ok on application screen
p.9	low	assign	Icon for application should be changed
p.8	low	orig	Menu should be seperated from dialog panel
p.7	hi	orig	Application crashes inconsistendly
p.6	emer	orig	OK button does not move to avoid mouse click

Activities

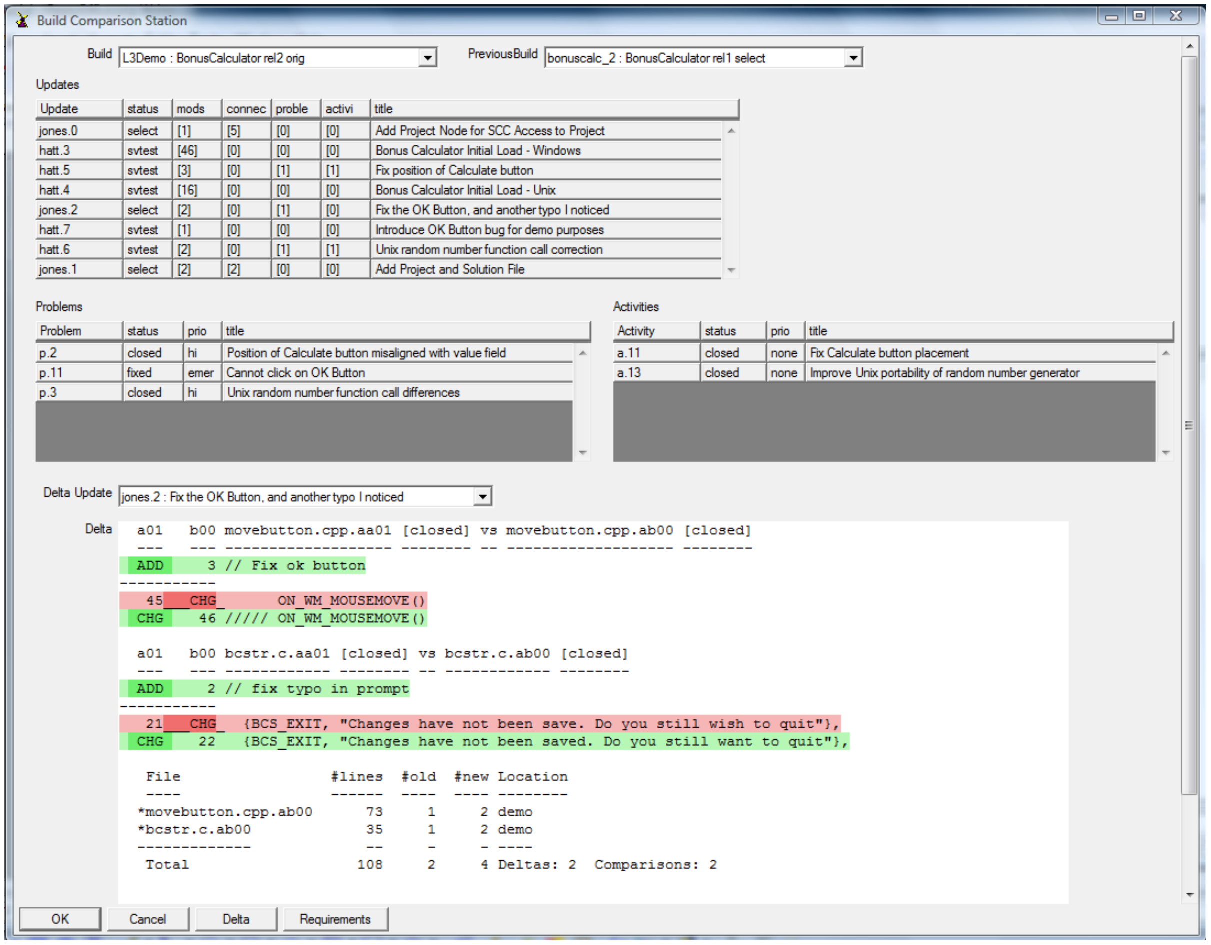
Activity	prio	status	title
a.23	none	entered	Enhance BC Precision
a.22	none	entered	Allow Dynamic Spec of Bonus Precision

Build Flow

Select **View Process Flow** from the **Builds** menu to view the state-based process flow for a build record. This process models a deployment, testing and release cycle. Typically, most builds do not make it through the entire process to production.

Comparing Builds

CM+ offers powerful traceability functions to support configuration auditing and project reporting. There are two ways to compare builds. The first is to use the Build Comparison Dashboard. Roll the cursor over the toolbar icons until the tool tip reveals the Build Comparison Dashboard.



Build: L3Demo : BonusCalculator rel2 orig
PreviousBuild: bonuscalc_2 : BonusCalculator rel1 select

Updates

Update	status	mods	connec	proble	activi	title
jones.0	select	[1]	[5]	[0]	[0]	Add Project Node for SCC Access to Project
hatt.3	svtest	[46]	[0]	[0]	[0]	Bonus Calculator Initial Load - Windows
hatt.5	svtest	[3]	[0]	[1]	[1]	Fix position of Calculate button
hatt.4	svtest	[16]	[0]	[0]	[0]	Bonus Calculator Initial Load - Unix
jones.2	select	[2]	[0]	[1]	[0]	Fix the OK Button, and another typo I noticed
hatt.7	svtest	[1]	[0]	[0]	[0]	Introduce OK Button bug for demo purposes
hatt.6	svtest	[2]	[0]	[1]	[1]	Unix random number function call correction
jones.1	select	[2]	[2]	[0]	[0]	Add Project and Solution File

Problems

Problem	status	prio	title
p.2	closed	hi	Position of Calculate button misaligned with value field
p.11	fixed	emer	Cannot click on OK Button
p.3	closed	hi	Unix random number function call differences

Activities

Activity	status	prio	title
a.11	closed	none	Fix Calculate button placement
a.13	closed	none	Improve Unix portability of random number generator

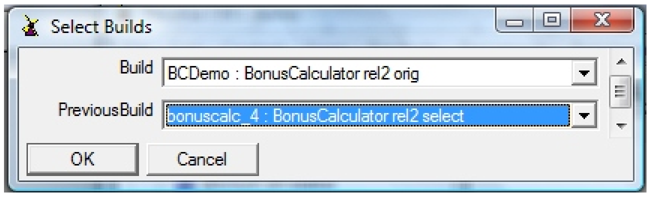
Delta Update: jones.2 : Fix the OK Button, and another typo I noticed

Delta

```
  a01    b00 movebutton.cpp.aa01 [closed] vs movebutton.cpp.ab00 [closed]
  ---    --- ------------------- -------- -- ------------------- --------
  ADD      3 // Fix ok button
-----------
    45   CHG          ON_WM_MOUSEMOVE()
  CHG     46 ///// ON_WM_MOUSEMOVE()

  a01    b00 bcstr.c.aa01 [closed] vs bcstr.c.ab00 [closed]
  ---    --- ------------ -------- -- ------------ --------
  ADD      2 // fix typo in prompt
-----------
    21   CHG      {BCS_EXIT, "Changes have not been save. Do you still wish to quit"},
  CHG     22      {BCS_EXIT, "Changes have not been saved. Do you still want to quit"},

   File                  #lines  #old  #new Location
   ----                  ------  ----  ---- --------
  *movebutton.cpp.ab00       73     1     2 demo
  *bcstr.c.ab00              35     1     2 demo
  -------------              --     -     - ----
   Total                    108     2     4 Deltas: 2  Comparisons: 2
```

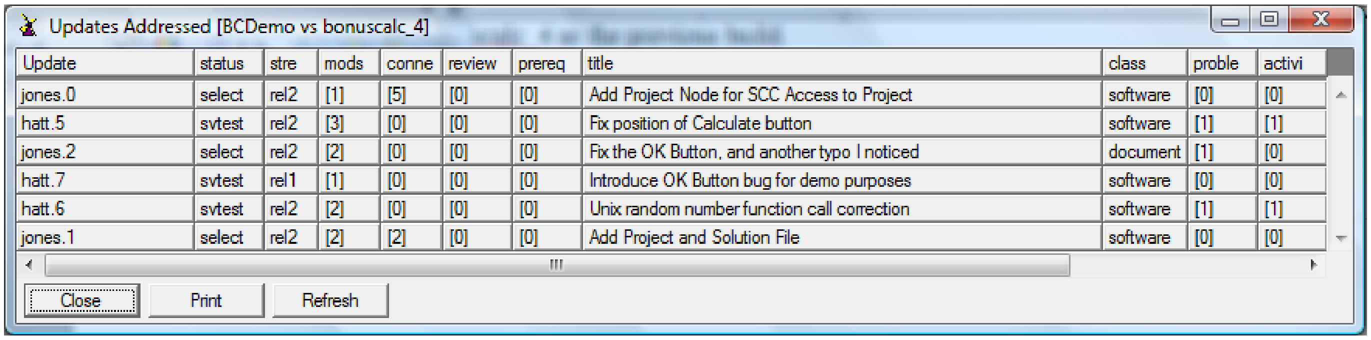
The two selectors at the top of the dashboard are used to select the builds to compare. Select your recently created build in the left side selector, and “bonuscalc_2” for the right side selector. The other fields will be updated. The Delta field will only show results when the update selected has had some source code changes. Select jones.2, the update you just completed, an look at the delta result here. The build comparison dashboard lets you zero in on changes made between the builds, problems fixed, features implemented, and specific delta reports.

A slightly less convenient, but more comprehensive, way to compare two builds is to first choose **Select Builds** from the **Builds/Compare Builds** menu (or select any of the Compare functions). From the “Select Builds” panel pick BCDemo as the current build and bonuscalc_4 as the previous build.



Build: BCDemo : BonusCalculator rel2 orig
PreviousBuild: bonuscalc_4 : BonusCalculator rel2 select

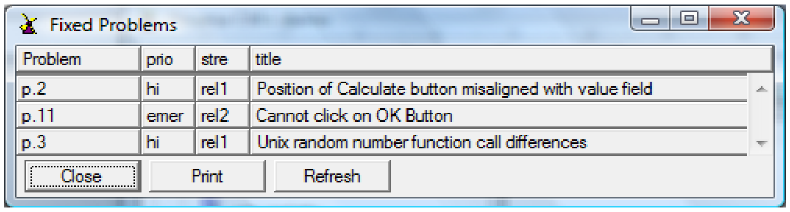
You are then presented with a list of the updates (changes) that went in to the new build.


Updates Addressed [BCDemo vs bonuscalc_4]

Update	status	stre	mods	conne	review	prereq	title	class	proble	activi
jones.0	select	rel2	[1]	[5]	[0]	[0]	Add Project Node for SCC Access to Project	software	[0]	[0]
hatt.5	svtest	rel2	[3]	[0]	[0]	[0]	Fix position of Calculate button	software	[1]	[1]
jones.2	select	rel2	[2]	[0]	[0]	[0]	Fix the OK Button, and another typo I noticed	document	[1]	[0]
hatt.7	svtest	rel1	[1]	[0]	[0]	[0]	Introduce OK Button bug for demo purposes	software	[0]	[0]
hatt.6	svtest	rel2	[2]	[0]	[0]	[0]	Unix random number function call correction	software	[1]	[1]
jones.1	select	rel2	[2]	[2]	[0]	[0]	Add Project and Solution File	software	[0]	[0]

You may click on the headings to see an aggregate of problems, activities, file modifications, etc. Or you may zoom into a particular record key or field for particular details about it.

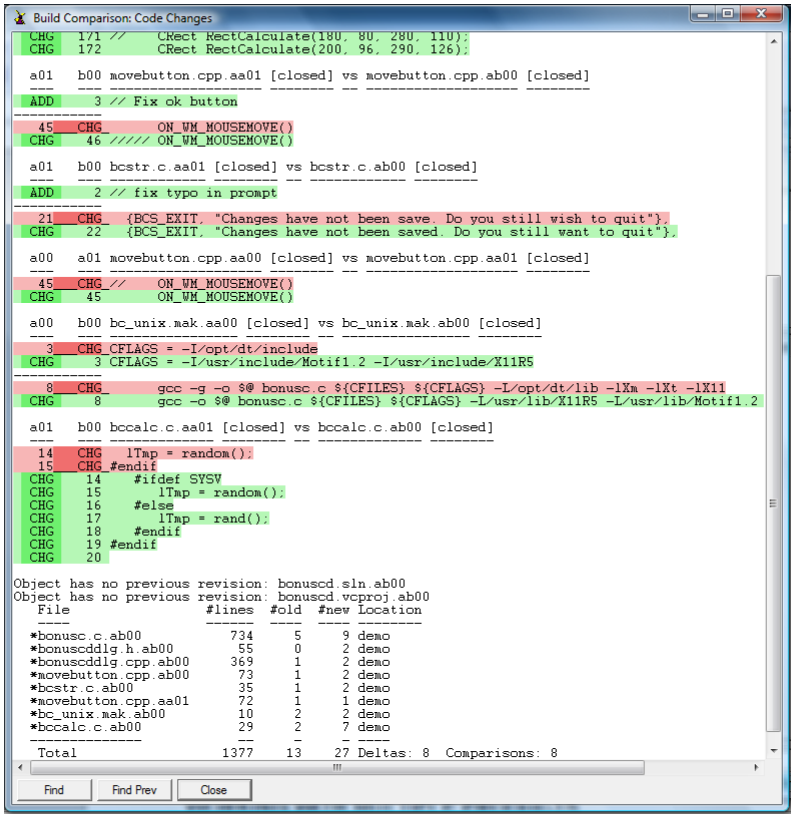
You may also pick any of the other comparison functions available under the Builds, Compare Builds menu. For example, to look at the problems fixed in BCDemo since bonuscalc_4, select **Problems** from the **Builds, Compare Builds** menu. Note that the problem reported for the OK button is in this list.


Fixed Problems

Problem	prio	stre	title
p.2	hi	rel1	Position of Calculate button misaligned with value field
p.11	emer	rel2	Cannot click on OK Button
p.3	hi	rel1	Unix random number function call differences

Creating a Build Delta Report

To look at all of the code changes made between the two builds, select **Code Changes** from the **Builds/Compare Builds** menu. CM+ will quickly compute and display all lines of code modified between the two builds.


Build Comparison: Code Changes

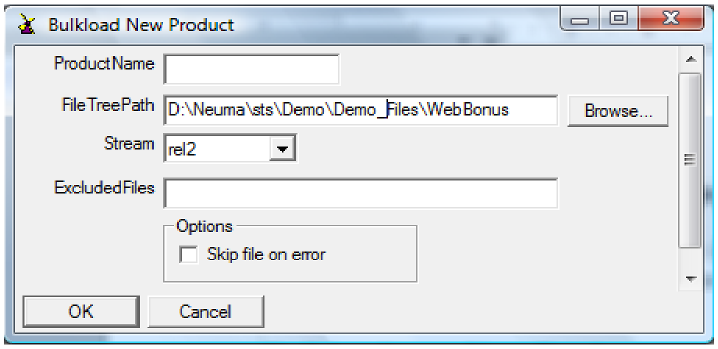
```
 CHG   171 //     CRect RectCalculate(180, 80, 280, 110);
 CHG   172        CRect RectCalculate(200, 96, 290, 126);

 a01   b00 movebutton.cpp.aa01 [closed] vs movebutton.cpp.ab00 [closed]
 ---   --- ------------------- -------- -- ------------------- --------
 ADD     3 // Fix ok button
-----------
  45___CHG_       ON_WM_MOUSEMOVE()
 CHG    46 ///// ON_WM_MOUSEMOVE()

 a01   b00 bcstr.c.aa01 [closed] vs bcstr.c.ab00 [closed]
 ---   --- ------------ -------- -- ------------ --------
 ADD     2 // fix typo in prompt
-----------
  21___CHG_   {BCS_EXIT, "Changes have not been save. Do you still wish to quit"},
 CHG    22    {BCS_EXIT, "Changes have not been saved. Do you still want to quit"},

 a00   a01 movebutton.cpp.aa00 [closed] vs movebutton.cpp.aa01 [closed]
 ---   --- ------------------- -------- -- ------------------- --------
  45___CHG_//     ON_WM_MOUSEMOVE()
 CHG    45        ON_WM_MOUSEMOVE()

 a00   b00 bc_unix.mak.aa00 [closed] vs bc_unix.mak.ab00 [closed]
 ---   --- ---------------- -------- -- ---------------- --------
   3___CHG_CFLAGS = -I/opt/dt/include
 CHG     3 CFLAGS = -I/usr/include/Motif1.2 -I/usr/include/X11R5
-----------
   8___CHG_       gcc -g -o $@ bonusc.c ${CFILES} ${CFLAGS} -L/opt/dt/lib -lXm -lXt -lX11
 CHG     8        gcc -o $@ bonusc.c ${CFILES} ${CFLAGS} -L/usr/lib/X11R5 -L/usr/lib/Motif1.2

 a01   b00 bccalc.c.aa01 [closed] vs bccalc.c.ab00 [closed]
 ---   --- ------------- -------- -- ------------- --------
  14   CHG    lTmp = random();
  15___CHG_#endif
 CHG    14    #ifdef SYSV
 CHG    15       lTmp = random();
 CHG    16    #else
 CHG    17       lTmp = rand();
 CHG    18    #endif
 CHG    19 #endif
 CHG    20

Object has no previous revision: bonuscd.sln.ab00
Object has no previous revision: bonuscd.vcproj.ab00
   File                     #lines  #old  #new Location
   ----                     ------  ----  ---- --------
  *bonusc.c.ab00               734     5     9 demo
  *bonuscddlg.h.ab00            55     0     2 demo
  *bonuscddlg.cpp.ab00         369     1     2 demo
  *movebutton.cpp.ab00          73     1     2 demo
  *bcstr.c.ab00                 35     1     2 demo
  *movebutton.cpp.aa01          72     1     1 demo
  *bc_unix.mak.ab00             10     2     2 demo
  *bccalc.c.ab00                29     2     7 demo
  --------------                --     -     - ----
   Total                      1377    13    27 Deltas: 8  Comparisons: 8
```

WORKING WITH MULTIPLE PRODUCTS

CM+ tracks multiple products, each with its own development team, source tree, builds etc. Products can be completely independent or dependent sub-products to a main product.

Bulkloading a New Product

To create a new product and bulkload in the source tree, launch CM+ as the CM Manager and use **Bulkload** from the **Products** menu. In the Bulkload New Product dialog, set the FileTreePath by browsing to the root of the directory tree to be loaded. In this demo you may browse to sts\demo\demo_files under your installation directory and select WebBonus as the root. The ProductName will be set automatically (based on the root directory name). Select rel2 as the stream. The ExcludedFiles field is used to specify file types or directories to be excluded from the bulkload (e.g. tmp, *.o, *.obj). Leave this blank and leave the Skip file on error checkbox unchecked. Click OK to start the bulkload.



ProductName

FileTreePath D:\Neuma\sts\Demo\Demo_Files\WebBonus

Stream rel2

ExcludedFiles

☐ Skip file on error

For more information, see Loading in Your Product Code in the on-line How-To Documentation.

Working with the New Product

On completion of the Bulkload operation, the new product will appear in the Products browser. The context will NOT switch to the new product however.

Set your context to the new product by selecting **Set Context …** from the **Context** menu or by right-clicking the product in the Products tree view and selecting **Set Context …** Select the product you just added and the rel2 stream. Your view will change to the source tree, updates and builds for the new product.

In this example we are going to treat this new product as a third party add-in to Bonus Calculator and deploy it to common area for the rest of the development team to use. First find the updates used to bulkload the product by clicking on **MyWebBonusUpdates** in the browser panel. There should be two (black.0 and black.1), one to create the product and another to bulkload the source. Right-click on each and select **Mark Ready**.

Before aligning the WebBonus product and creating a new baseline, promote the same updates to status of select. Do this through the **Select Updates…** function under the **Builds** menu. Choose rel2 as the Stream, ready as the FromStatus and select as the SelectStatus. Click OK then right click on each update in the subsequent display to promote it to select.

Align the WebBonus product by selecting **Align Product…** from the **Builds** menu. Freeze the alignment and create a new build by selecting **Freeze Lineup…** from the **Builds** menu, choosing the product root directory as the starting point for the freeze. Check the Add build checkbox and click OK. After the product tree is frozen an Add Build dialog will appear. Enter a name and a title for the new build and click OK.

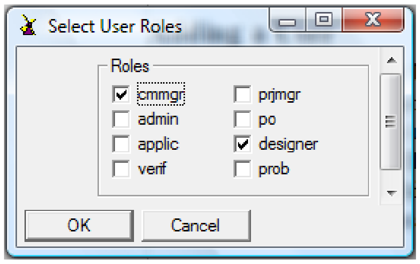
You may now deploy the new build by selecting **Get Build…** from the **Builds** menu. Choose the build you added in the last step and choose a location on the file system to deploy to.

At this point, you have loaded in a new product, created the first baseline and first build record, and deployed the source for that build record. The next step would be to actually create a build deliverable from the deployed source code. If your make files are part of the source code, this might be as easy as executing the make files. But there are many other features of CM+ that you may wish to study to see what procedures best suit your operating environment and project.

USER AND LIBRARY MANAGEMENT

Adding a User

The project manager may be responsible for the addition of users and assignment of user roles. In some organizations, this responsibility will fall upon the Administrator of the library. To add a new user, select **Add User** from the **Users** menu. A form will appear with the user's particular's to be filled in. The "UserId" field is the user's login name for CM+. To enable automatic login, the user ID should be the same as the user's OS login account or a single token subset of it. Otherwise, the user will have to specify his or her userid every time they log into CM+. After the form has been filled in, click **OK**. The user is added and a dialog appears requesting the roles for this user. A user's role determines their access to menus functions as well as their access to commands and data.



✓ cmmgr	☐ prjmgr
☐ admin	☐ po
☐ applic	✓ designer
☐ verif	☐ prob

It is also possible to add a new user directly reporting to another user by right-clicking on the user to which the staff is to be added and selecting **Add User**.

Changing a User's Roles

To change a user's roles, select **User Roles** from the **Users** menu and then select the user and **OK** from the dialog box which appears. A second dialog box will appear. Adjust the roles as required by clicking on the check boxes and then select **OK**.

ENABLING MULTIPLE USER ACCESS

The demo library installs rapidly for evaluation by a single user. If you wish to evaluate a multiple user environment, you may create a separate library or you may modify the access to the demo library to support multiple users.

Generally, multiple user access is accomplished by giving read-only file system access to the STS installation directory (typically **C:\Neuma\sts** on Windows, **/usr/neuma/sts** on Unix), and providing a transaction server to provide transaction handling. [In the single user demo environment, the installer was given write access to the transaction directory – this is not the normal operation. This was to simplify server operation.]

To modify access for the demo library, perform the following:

1) In the **STS/demo/demo.sts** file, insert the line: set sitemode net
2) In the **STS/projects.dat** file, make sure that for the txn server, a valid IP name or address is specified that can be interpreted by all clients.
3) From the administrator account (stsmgr), use the **Administrator | TxnServer | Start Net Server** button to start the transaction server.
4) Make sure that the STS root directory has shared **read** access to every client. On Unix, also make sure that the files in the "bin" directory have shared **execute** access for every client.
5) Make sure that all clients have their STS environment variable pointing to the STS root directory. On Unix platforms, this will happen automatically if clients use the "sts" script found in that directory to launch CM+. On Windows platforms, the "sts.bat" script will set the STS environment variable to a default value if it is not otherwise set. So either have clients set the STS environment variable, or better yet, set the default value for STS in the "sts.bat" script. This must be done in a client independent manner (e.g. use a common mount point: **S:\Neuma\sts**, or use universal naming convention, **\\servername\C\Neuma\sts**). The default value will replace the C:\Neuma\sts default value normally used in the sts.bat script.

This is the formal end of the Demo Tutorial. At this point, you may wish to contact Neuma to determine how you might proceed to either acquire licenses for your project, or, if you already have them, how you might set up your own project library.

Additional notes are included below for some common operations you may wish to perform after your demo evaluation is completed.

DEPLOYMENT STRATEGY FOR CM+

Once initial evaluation of CM+ has been completed and a decision has been made to acquire CM+ for your project, there are a number of steps that must be taken to deploy CM+. It is expected, though not absolutely necessary, that a Neuma support person will spend 2 or 3 days on site, helping you with your initial deployment and on-site training. Unless you have complex requirements, such as transferring all existing revisions from your existing CM tool, or extensive modifications to the out-of-the-box process that need to be completed before user training, you should not require extensive consulting and implementation services from Neuma.

Deployment consists of seven basic steps:

1. Understand the scope of CM+ functionality you wish to use initially
2. Identify initial projects that will use CM+, including, if applicable, projects across multiple sites.
3. Install CM+ on the target server(s)
4. Load in your existing data (apart from files)
5. Load in your existing software product baselines
6. Adjust the Pre-defined CM+ processes and user interface
7. User Training

Steps 1 to 3 should be done prior to visits by Neuma. As well, preparation for steps 4, 5 and 6 should be done prior to a Neuma visit. If you wish, you could attempt these steps on your own, but you might want a bit of training before trying to adjust your processes and make any associated user interface changes. Step 7 includes both end user training and CM manager training.

These steps should be used as a guideline for deployment. Once you have made a decision to purchase, you should work with Neuma to ensure that deployment will be both low risk and swift.

Scope of CM+ Functionality

The first step in deploying CM+ is to decide which parts of CM+ are going to be used initially. Neuma recommends the use, at a minimum, of Problem Tracking, Change Management, Version Control and Build Management. We also recommend at least very basic Activity Management. This could be in much the same form as for Problem Tracking (i.e. activity title, objectives, priority, [target development] stream, owner and assignee), or it can be more elaborate if you prefer.

The fact that you use Problem Tracking or Activity Management initially will not increase the learning curve significantly. Users will most likely just see their "To-Do" lists and click on them and select operations from the objects displayed (e.g. Fix Problem, Reply to Problem, Implement Activity). These will lead them through to the rest of the Change Management functions while ensuring traceability.

You may want to delay extending your processes into new territory initially (other than the above) unless your processes are already well understood by the development team. For example, if you're already using Requirements Management, but doing so by publishing a spread sheet, it might be good to move this function into CM+ sooner than later. You may even wish to delay Change Management and just start out with a simple Problem Tracking to get your users used to the tool. However, if they are not doing a lot of Problem management, that might not help.

When considering the scope of CM+ functionality, it is important to identify any functionality that you have not had a chance to evaluate. If this is key functionality for your site, it's important that you understand how it works. Most CM+ functionality is highly configurable. But some is much less so. For example plug-ins or interfaces to other tools, which are restricted to working with 3rd party interfaces, will naturally have a lower level of configurability. In any event, make sure key functionality meets or exceeds your expectations before committing to it. If there are areas that need improvement, discuss this with Neuma as Neuma is usually in a position to address your requirements in a priorty fashion.

In any event, remember that CM+ has been designed so that both the process configuration and the user interface can easily be changed over time. This allows you to select an appropriate starting point and move forward from there over time.

Identify initial projects to use CM+

One of the more difficult decisions is to decide which projects are going to be the first to use CM+. We recommend selecting projects where the participants are well suited for helping with the migration of CM+ to other projects. You should expect very little disruption in your migration from your current CM methods to the use of CM+ (unless you're making sweeping process changes at the same time, in which case there will be some process learning curve). However, we would also recommend that you select projects that are at reasonable cut-over points (e.g. end-of-release, new project, lull in development, etc.).

Although CM+ has been designed to match most projects well, not all projects will be a perfect match for CM+. For example, a combined hardware/software project may raise more issues than a pure software project. When considering your initial projects, select an initial one that can be deployed easily. Leave more difficult projects to subsequent deployments. This allows you both to improve your training in CM+ before having to address more difficult issues, and provides a successful model for use in other projects. And remember that the difficulty level is not often a technology factor, but is often a personnel factor, whether on the management side or with the team member buy-in (e.g. resisting change or familiarity with older less functional tools).

Install CM+ on the target server

Neuma recommends a Windows, Linux or Solaris server for CM+, although any Unix server will do (Windows, Linux and Solaris have the most advanced user interfaces at this point in time). The power of the server is not really important until the number of users rises above a couple hundred, provided that the server is not already nearly fully saturated with other applications running.

Neuma also recommends that the server is one to which your clients have read access through the file system (e.g. NFS or SAMBA, or native Windows networking). This eliminate the need to have to maintain multiple CM+ environments (including customizations and upgrades), and provides for a single machine install operation (instead of an install operation for each user workstation).. Instead, a single environment per site should be sufficient.

Finally, Neuma recommends installing CM+ using the default settings to simply any support communications. However, this is not necessary and will not affect operation or performance. Installation should be a 5 minute process.

Loading in your data

There are a number of steps to loading in your existing data, but they are all straightforward. Remember, when you do bulk loading of data, it is a good idea to first disable your Automatic Save ("autocommit") feature, using Edit | Preferences | Options (and disable the "Automatically Save" option), and to view the results before you explicitly commit them to the repository. Generally, it's easier to fix up your data entry file than to have to re-do or correct parts of it. If you have errors or are not satisfied with your results, use the File | Cancel menu item to cancel your transaction and to try again.

You may have data in other repositories that you wish to continue to maintain in the other repositories for any number of reasons. We first point out that having the data integrated into a single library will give you much more capability than having it separate. This includes both data navigation and handling cross-application process issues (e.g. problem promoted when update submitted). So in these cases, you may want to consider one of a number of options including:

- Migrating the data to CM+
- Replicating part of the data in CM+ (e.g. title, priority, assigee, status), and maintaining that replication.
- Exploring the level of integration between CM+ and your other tool. This will likely be more expensive and will not result in seamless integration (and hence will be less user friendly) but may be your hard requirement.

Load in your Users

The best way to load in your existing users is to create a text file in tabular fashion. You'll probably want the following fields to start. You can add in others later. (Note that you can always modify the user data schema at any time.)

Userid: their Unix/Windows login ID. However, if their login id has dots or spaces in it, we recommend taking the part after the space (eg. "smith" for John Smith). Traditionally userids are all lower case, but you may use mixed case if you wish.

Name: Their full name in "quotes"

Title: Perhaps their current position in the organization, in quotes

Phone: or Extension: Their current phone number and/or extension. Note that the current CM+ application configuration requires that a 0 or 1 is the middle digit of the area code. So you may wish to leave this field out if you have phone numbers with modern area codes (not x0x or x1x), or if you have international numbers. You can change the configuration later to support these (e.g. use a numeric string field).

Staff: A list of user ids (in quotes), defined higher up in this file, which are members of this user's staff

So for example, you might have a file, named "users.dat" such as:

```
jones   "Davey Jones"   “Application developer"  2133
ford    "Gerry Ford"    "Process Engineer"       2216
black   "John Black"    "CM Manager"             2214  ford
smith   "Joe Smith"     "Application Manager"    2210  "black jones"
```

Then if you execute the following command, it will load in these users:

runfile users.dat {add users (name title extension staff) -data '^1;^2;^3;^4;^5' }

it will add in all of these users.

Load in your Current Activities/Tasks

Similar to the Users loading, you may add in your initial set of activities. These would normally be dumped out of some existing spreadsheet or database. So for example, "acts.dat" might look like this:

```
"Creation of plug-in option 1"  hi  ABC rel4 smith jones
"Creation of plug-in option 2"  med ABC rel4 smith jones
"Plug-in creation"              hi  ABC rel4 smith smith
"Enhancement of Update Dialog"  hi  ABC rel4 smith ford
"Release 4 Activity WBS"        hi  ABC rel4 smith smith
...
```

You would then execute a command such as:

runfile acts.dat {add acts (title priority product stream owner assignee) -data '^1;^2;^3;^4:^5;^6' }

CM+ would then run that file and add activities to your repository, assigning activity numbers to each of them. You could subsequently drag and drop the activities onto one another to form the WBS (work breakdown structure), or you may choose not to use a WBS and just view your activities based on release and priority..

Load in your Current Set of Problem Reports

Loading in problem reports is very similar to the above example. You might have the following "probs.dat" file:

```
"Plug-in 1 crashes when configuration is minimal" hi   ABC rel4 smith jones
"When default left blank, unexpected behaviour"   med  ABC rel4 smith jones
"Typo on page 4 of the user guide"                low  ABC rel3 smith jones
```

You would then execute a command such as:

```
runfile probs.dat {add probs  (title priority product stream owner assignee) -data '^1;^2;^3;^4:^5;^6' }
```

Load in software products

Generally, but not always, all software controlled by the same release time-lines (i.e. sharing the same product road map) should be treated as a single product. A single product in CM+ may have multiple deliverables (e.g. client software; server software). The note in the "Loading in your Data" section on disabling automatic saves applies here as well. You'll probably want to commit your steps at various checkpoints (e.g. every time a bulkload operation is completed and you've reviewed it and are happy with it). In the case of software, you may want to do a few trial runs and look at the resulting trees to see if they turned out the way you wanted. This is often best done in a parallel test library. It's sometimes easier to change the original input data than to change it in the repository. However, this is not always the case. So experiment.

Loading Latest Baseline Only

To load in the latest baseline for each product, use: Product | Bulkload | Load Product Tree

Repeat this operation for each product to be loaded. At a later time you can worry about whether or not some products should be handled as sub-products of other products. Browse to the root directory to be loaded for each product. By convention, and to help minimize conflicts, the product name is specified as a fairly compact, all uppercase name with no special characters or blanks. Examples: MYPRODUCT MSWCLIENT MSWSERVER MYSQL.

Note that each product tree must have a single root node. If you have several deliverables in the same product, they might be grouped under this root node. If you have some products that don't have the source available for bulkloading at this time (for whatever reason), you may load those sub-trees in at a later time using the BulkLoad SubTree function found under the Updates menu (New sub-menu). However, if you want these to be part of the initially loaded baseline, you will have to add these in before you freeze the initial baseline.

Generally, it's a good idea to freeze a baseline as soon as it's been (bulk-)loaded in. This gives you a permanent record of what was loaded. You may freeze a baseline by either right-clicking the root of the tree and selecting Freeze, or by using the Builds | Freeze menu item. Optionally you may add a build record as part of the freeze operation to make it slightly simpler to compare future builds to this baseline and to make it slightly easier to create your next build in the same release stream. However you can always compare a build to a baseline or use the Add First Build menu item to create your first build off a baseline.

As part of this, you should establish a build naming convention. CM+ gives you a starting point for default build names, but this may be modified to suit your preferences.

Loading Multiple Baselines of a Product

Load in the earliest baseline for each product using: Product | Bulkload | Load Product Tree

Subsequent baselines may be loaded in as subsequent revisions of an earlier baseline, or as a new branch off an earlier baseline. However, the earlier baseline should be frozen prior to loading in a subsequent one. Generally, if you are loading the earlier baseline for reference purposes only (i.e. no more development on them), you may load your next baseline on top of it. If you are loading old baselines to continue support on them, only baselines which supercede others, from a support perspective, should be loaded on top of the superceded baselines. An actively supported baseline should maintain itself as the tip baseline of a branch. Subsequent baselines should be added into a separate branch or branches, again, according to the support requirements.

So, for example, if 6 baselines of product ABC are being loaded, rel1, rel2-1 rel2-2 rel2-3, rel3-1 and rel3-2, you would likely want to continue support of rel1, rel2-3 and rel3.2. So you would load in re1 baseline in stream rel1, then load rel2-1, -2 and -3 all into the rel2 stream, and then rel3-1 and -2 into the rel3 stream. This would give you the six baselines arranged as follows:

```
ABC
  ABC.aa    rel1  from  -
      ABC.aa00
  ABC.ab    rel2  from: ABC.aa00
      ABC.ab00
      ABC.ab01
      ABC.ab02
  ABC.ac    rel3  from ABC.aa02  (or ABC.aa01 if it were branched off earlier, and hence loaded prior to ABC.aa02)
      ABC.ac00
      ABC.ac01
  ABC.ad    rel4   (new development would start here)
```

After loading ABC.aa00, you would freeze that baseline, and then use the:

Products | Bulkload | Analyse Tree Revision, and
Products | Bulkload | Load Tree Revision

menu items to load in each successive baseline. You would specify the appropriate stream for each load operation, and do your best to help with the Analyze step, where CM+ will try to reconcile newly added or moved directories and/or files to their positions in the previous baseline to maintain better history traceability.

Adjust the Pre-defined CM+ Processes and User Interface

CM+ comes pre-customized with a fairly reasonable set of processes. However, you are likely going to want to tweak these processes. We recommend that you first view the process flow diagrams (Process | Process Flow | View State Flow) and identify the changes you might like to see. Keep in mind that there may be a few states that are not so easily configured as they are referenced in various triggers, rules, scripts, etc. But these are few and generally will not need to be renamed.

Other state flow changes can be identified and Neuma can review these changes and either apply them or help you to apply them. Once you have identified your candidate changes, Neuma can likely make the changes in a few minutes to a few hours. Process may be modified at any time, it does not have to be perfect to start. What we do recommend is that the process that affects most of the users be near perfect so that there is not the need for significant incremental process training for your users.

You may wish to use the process flow diagrams to inspect and possibly adjust the process flow in CM+. There are two ways to do this. One is to adjust the process flow in the "schema" files (see the bottom of the .sts files) where they are initially defined. This will allow you to define the schema for any new libraries you may create in the future.

A second way is to adjust the process directly in the library using the process flow diagrams for each type of object. In this way, you are affecting only the processes for the specific library that you are adjusting. When you make your adjustments, you can export the changes for one or all modules using the Process | Schema | Dump... buttons. These can then be placed in the schema directory for use during the creation of other libraries.

The process definitions also involve things such as high-lighting (criteria and colors) and documentation of symbolic range elements (e.g. Priority "hi" means that "it must/should be addressed before the next release"). These you may deal with at a later time when you're fine-tuning your CM+ library.

You may also wish to tweak the CM+ user interface. This will likely involve adding and/or removing a few menu items, to-do lists or other items from each role. CM+ default roles are most easily modified through graphical user interface (GUI) configuration. The GUI configuration affects who sees what menus, what's in each menu for each role, what menus are defined, what tree browsers and to-do lists appear in the top left window, and what the contents of each of the system's dialog boxes are.

Normally there are few changes. For example, one customer may prefer to retrieve files to the workspace by default when checked-out, while another may wish the default does not retrieve them. You may want to pre-can a couple of reports or metrics that aren't already available, or you may wish to simplify the menus for certain roles. These are all fairly straight forward operations that normally (unless you require extensive changes) will not require additional consulting costs from Neuma. The GUI can be customized by role, or even at a finer level (e.g. user) where it might be necessary in some cases.

Some customizations are easy to perform without affecting existing Neuma-supplied GUI configuration files. This makes it easy to incorporate future advances in Neuma's GUI configuration (although you're always free to ignore future changes to the GUI configuration). Other changes may need to be re-applied with each release of CM+. The CM+ merge facility can be used to automate most of this work.

Yet in other cases, you may wish to add additional fields to your data schema as part of your initial customization. These might be necessary to define your process or user interface more specifically. As will all other changes, these may be done prior to deployment or at any time subsequent. Some customers perform continuous improvements, while others like to save them up and hold a user session when a significant number of changes are being made.

User Training

The last part of the deploy procedure deals with training your users. Neuma offers training courses on the basics of using CM+. There is a 1/2 day user course which covers Problem Tracking, Change Management/Version Control and Context/Workspace Management (plus a touch of Activity Tracking). This is intended to be customized to each project as necessary. It can be delivered by Neuma, or can, through the Training Kit, be licensed for delivery by the customer. The course should also be customized to work with a copy of your populated repository, although it can run on Neuma's default training library.

Training of Power Users is a full day course and covers more of the power features of CM+. It allows users greater flexibility in exploring CM+ and expanding their knowledge of CM+ capabilities. The goal here is for power users to gradually, over time, pass on the features they find most useful to their peers who have had minimal training.

The other aspect of training that should be completed up front is the Configuration Management and Administration course. This is a two-day course and it is recommended that at least two personnel take the course so that they can spare off one another (vacations, sickness, departure). That being said, once the basic set up of the CM process is completed (e.g. setting up nightly compiles to run automatically, etc.), the CM and Administrator jobs will be a part time effort, and might focus initially on bringing new users on board with CM+. It will also involve interaction with Neuma for specific support queries and customization requests.

Up and Running

At this point in time, your CM+ library(s) for the trained user base should be up and running. If you need to immediately provide CM+MultiSite capability, a few simple steps are required to replicate an initial checkpoint of your library for use at other sites. Once the replication is complete, your initial site can continue running and your other sites can be brought on-line as required. It is recommended that you run at least one other site to provide a warm stand-by disaster recovery. Neuma provides a specially priced server that can be used only for disaster recovery (i.e. no users are allowed at the site until it is switched to become an active user site).

Neuma recommends having a Neuma consultant on site for 2 or 3 days for your initial start-up and training, but this is not essential. Keep in mind, however, that switching to a new tool is a significant change for your team, and anything that can be done to make the transition easier is well worth it. Neuma's consulting rates are priced to make this affordable. If you require support at any time, Neuma can be reached most effectively by email at cmsupport@neuma.com, or through direct telephone support.

CONVERTING TO CM+MULTISITE

(Note licensing is done independently for each site)

These instructions assume that you have already created your library running in "single site" mode. CM+ MultiSite requires that you have fixed IP addresses assigned to each server machine.

It also assumes that you have the same versions of CM+ running at each site (i.e. for CM+ MultiSite, you need to install CM+ at each site, on the server machine). You should verify that you can use the demo library at each site before proceeding. This ensures that your CM+ MultiSite installation is working properly.

When you convert a library to MultiSite mode, you will:

- Configure the library for use as a CM+MultiSite library (including shutting down the servers when instructed)
- Create a synchronized image of your library for transport to the other site(s).
- Restart your library on the original site (normally the master site)
- Transport your synchronization images to the other site(s)
- Startup your library on the other site(s)

The configuration will involve the following files (STS stands for your CM+ root installation directory):

STS/<libraryname>/stssites - a new file to be created.

STS/<libraryname>/<library.sts> - your library preferences file

STS/projects.dat - your registry of libraries

Note that if you are doing a substantial amount of initial data fill in a library, it is usually quicker to do so prior to conversion synchronization. This results is less data transfer during bulkload operations. It may be beneficial to discuss this with a Neuma representative.

To convert the library to multi-site, you need to perform the following:

(M) At the MASTER SITE

(where your library resides already)

M1. Create an "stssites" file (see below) in your LIBRARY ROOT directory, "STS/<libraryname>". Specify the same port in the stssites file as you specified for the transaction server in the projects.dat file. Otherwise, you will need to change both files.

M2. Stop your servers and insert the line "set sitemode multi" in "<libraryname>.sts", replacing any other line that "set sitemode".

M3. Optionally modify your port number in your projects.dat file (if you did not keep the same port you had previously).

M4. Create a site image for your library. To do this, with all of the servers stopped, zip or tar up the entire library directory. We recommend that you identify this "archive" by appending a date stamp, or the current transaction session/id numbers shown by the status command.

M5. Now restart your servers using the Administrator menu and verify proper local site operation. For the Master site (on which are are currently working), "Administrator | TxnServer | Start MultiSite Master". After waiting a few seconds, verify that the "status" command shows "multi".

M6. Ship your site image file to your slave site.

(S) At a new SLAVE SITE

[at any time before you do a Txn Session creation, and typically within a few hours]:

S1. Install the same version of CM+ as at the Master site (or perhaps zipped from the master site)

S2. Unzip your library into the STS directory so that it parallels the Master site.

S3. Edit your projects.dat to reflect the local slave site. We recommend that you use the same txn ports for a given library at all sites, but this is by no means required. It just eliminates one degree of potential confusion.

S4. If you have fixed user client licenses, you may wish to assign differnt users at each of your sites. This can be done by modifying the "fixedlic.users" file at each site (one user per line), or by using the "Administrator | Licenses | Set Fixed Licenses" menu item. This can be done at this time or at a later time.

S5. Start up your servers using the Administrator menu and verify proper local site operation. For a Slave site: "Administrator | TxnServer | Start MultiSite Slave". After waiting a few seconds, verify that the "status" command shows "multi".

“stssites” File

CM+MultiSite setup requires a configuration file named "stssites" in your library root directory (e.g. C:\Neuma\sts\demo\stssites). This file could use a numeric IP address or a domain-name based address. Each line contains the following items:

SiteName: The name of your site - a single alphanumeric token

Master/Slave: One site must be designated the master site, and the others designated as slave sites.

IP Address: The IP address of each server machine. Ideally these are named the same across all sites.

TXN Port: This port number is the same as specified in the "projects.dat" file for the txn port.

Local Txn Directory: The directory, local to each site, containing the library transaction directory.

Fields are blank or tab separated.

The txn directory path is the local path to the txn folder of the library on each site. If may be specified using forward slashes only (as directory separators) or using the host convention for directory separators.

The "stssites" file should be identical on all sites. However, it is allowed for slave sites to omit other slave sites if convenient. When the synchronization image is created, the stssites file will be in it so that it is copied to the other site(s).

```
*SITENAME    M/S       SERVER-IPADDRESS  TXN-PORT  LOCAL TXN DIRECTORY
richardson   master    satellite.adva.com    9450  C:\neuma\sts\mslib\txn
china        slave     chinacm.adva.com      9450  /chinaapps/neuma/sts/mslib/txn
texas        slave     houston.adva.com      9450  C:\neuma\sts\mslib\txn
```

See the CM+MultiSite Administrator Guide (Help | CM+ Documents | CM+ MultiSite Guide) for additional information.

CM+ MultiSite Verification

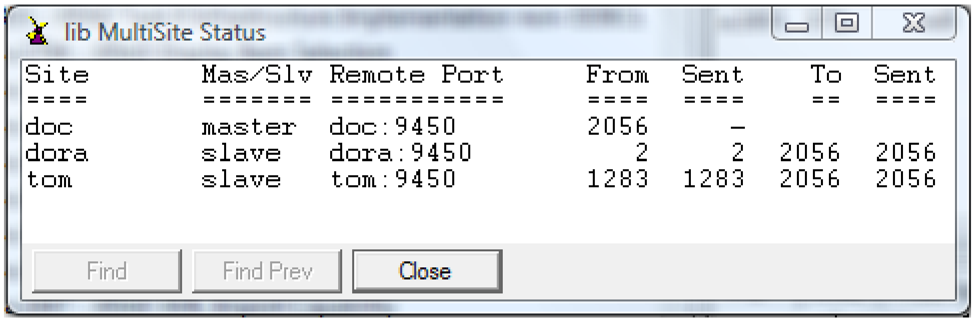
When your master and slave sites are both up and running, you may verify operation of the sites by committing a simple transaction from each site, and verifying that it has completed at all sites.

You will notice the following files in your txn directories.

MASTER: queue.log master.index <slaveN>.index

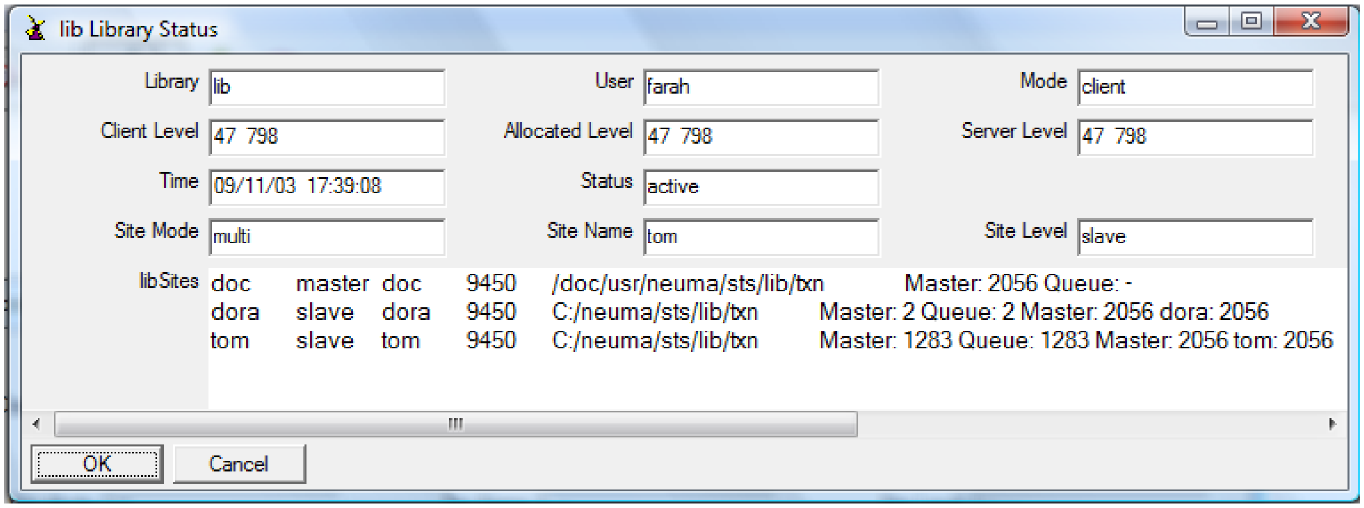
SLAVE: queue.log master.index queue.index

The contents of these files should be shown to you if you perform: Administrator | TxnServer | MultiSite Status. If this takes longer than a few seconds or returns invalid data, you may have a problem with your MultiSite setup.



```
Site        Mas/Slv  Remote Port    From  Sent    To  Sent
====        =======  ===========    ====  ====    ==  ====
doc         master   doc:9450       2056     -
dora        slave    dora:9450         2     2  2056  2056
tom         slave    tom:9450       1283  1283  2056  2056
```

In the above example, the master, "doc", has sent out 2056 messages from its queue. Slaves "dora" and "doc" have both received all of the messages. As well, "dora" has sent out 2 messages, and "tom" has sent out 1283 messages to the master site.



Library: lib
User: farah
Mode: client
Client Level: 47 798
Allocated Level: 47 798
Server Level: 47 798
Time: 09/11/03 17:39:08
Status: active
Site Mode: multi
Site Name: tom
Site Level: slave
libSites:

```
doc    master  doc    9450   /doc/usr/neuma/sts/lib/txn      Master: 2056 Queue: -
dora   slave   dora   9450   C:/neuma/sts/lib/txn     Master: 2 Queue: 2 Master: 2056 dora: 2056
tom    slave   tom    9450   C:/neuma/sts/lib/txn     Master: 1283 Queue: 1283 Master: 2056 tom: 2056
```

The above screen contains similar information from the File | Library Status menu item, and shows the site mode, name and level (multi, tom and slave). It also shows the content of the stssites file.

NOTES

Upgrades to all sites should be done at the same time to ensure libraries don't diverge.

Txn session creation must be done at the same time at all sites in a strict order. For Unix-only sites, this should happen naturally. Windows servers will sometimes not allow a txn session creation to properly complete because of process references to the txn directory. When this happens, manual intervention may be required to complete the operation.

CM+ MultiSite Detailed Instructions

To convert a single site CM+ installation to CM+ MultiSite, do the following:

- Create your "stssites" file, filling in all site entries, each txn directory relative to its own site.
- Stop your current transaction server (Adminstrator | Txn Server | Stop Net Server) and Library Server (Administrator | Stop Server)
- Change the sitemode in your "<library>.sts" file (e.g. c:\Neuma\sts\mslib.sts) to: set sitemode multi [previously this was net or nfs or absent]
- Zip or Tar up your library directory.
- Start up the library server: Administrator | Start Server
- When you restart your transaction server, perform Administrator | Txn Server | Start MultiSite Master on the master site (you will use Start MultiSite Slave on the other site(s)).

You are now running CM+ MultiSite with a single site. Your slave sites may be started at any time prior to your next "txn session" creation. They will catch up from this point in time as long as the sites are already defined in the stssites file.

Send the library zip/tar to the slave site(s). When a slave site receives it, proceed as follows:

1. Either unzip it at the new site, or

- Install CM+ (same version as at Master) and install the site specific license key file
- Apply any "coms" or "gui" file customizations you have made to your master site to the slave.
- Unzip/untar the library into the "sts" directory of the CM+ installation root

2. Modify the projects.dat file to specify the correct IP address (and port if using a different port number - recommend using same ports across sites for a given library)

3. Startup your slave serversm either through an "stsstart" script or manually as follows:

- Start up the multisite server: Administrator | Txn Server | Start MultiSite Slave
- Start up the library server: Administrator | Start Server
- Start up the license server if you are using floating licenses (Administrator | Licenses | Start License Server). Otherwise, select the users at the slave site who are fixed users using the Adminstrator | Licenses | Set Fixed Users

CM+ MultiSite Transaction Server

The CM+ MultiSite Transaction Server is started/stopped on Unix as follows:

```
Start Master:  $STS/bin/txnserv <libraryname> -m
Start Slave:   $STS/bin/txnserv <libraryname> -s
Stop Master:   $STS/bin/txnserv <libraryname> -m -k
Stop Slave:    $STS/bin/txnserv <libraryname> -s -k
```

Windows has a more complex structure due to the way in which it spawns processes when a new connection appears. On Unix, a new process is spawned as needed to serve a connection. On Windows, separate processes are available for txn server interaction with the client and txn dequeuing operations, for interaction between sites.

The following options can be used with the transaction server batch file: %STS%\bin\txnserv <libarayname> <options>, where options are as follows:

-m -t   starts master txnserver only (client interaction)

-m -f   starts master dequeuers only

-m -f <site>   starts master <site> dequeuer only

-s -t starts slave txnserver only

-s -f starts slave dequeuer only

-k will stop all processes

"queue.stop" in the txndir will stop all dequeuing processes. "<site>.stop" in the master txndir will stop dequeuing process for <site>. All .stop files will remain until txnserver queues restarted

TheTxnServer menus use a different user interface for starting and stopping CM+ MultiSite, with separate menu items for Master and Slave sites. Finer granularity functionality is supported through command line invocation. Automatically named .log files are created for dequeuers: <site>.log (on master) or slave.log. Txnserv.log is for the txnserver (site message server) only.

To start up the Master multisite transaction servers on WIndows, the following STS command is executed:

```
runfile @libdir/stssites \+
  {if $^take 1 ^1 -not -eq $*; \+
      if $^2 -eq $master; \+
         oscom @stsbin^txnserv @library -m -t; \+
      elsif $^2 -eq $slave; \+
         oscom @stsbin^txnserv @library -m -f ^1; \+
      end; \+
   end }
```

This command loops through all of the entries of the stssites file and starts a (master) txnserver process and one dequeuer process for each slave site. The Slave side has a simpler command structure:

```
oscom @stsbin^txnserv @library -s -t; \+
oscom @stsbin^txnserv @library -s -f
```

This is because it only has to dequeue messages from the Master site and from its own clients.

Key Log Files for CM+ MultiSite

There are a number of log files that may be used in the case of trouble-shooting CM+ MultiSite. These include the following:

Log Files:

txnserv.log (and txnserv.logX) - transaction server log files.

<site>receipt.log [slave only] - a list of messages received by that site from the master

slavedq.log [slave.only] - log file for a slave dequeue process

A number of other files are used to drive the transaction server operation. These files should not be locked, as they may be changing frequently in a CM+ MultiSite environment.

Live transaction server files:

queue.log	- Live file - do not write lock this file... list of transaction messages sent to the master/slaves.
*.index	- Live Index files: queue.index, master.index, <site>.index

NEUMA SERVICES

There are multiple server programs involved in the use of CM+. These include the License Server, the Transaction Server, Query Server, and the Library Server. The latter 3 of these may have multiple instances running, such as one or more per library. These processes may also have pre-conditions to starting. For example, a library server must not start if there is an "inservice" file in the library directory, except on a re-boot.

CM+ servers may be started and stopped manually, using the Administrator menu items. Such server processes will automatically terminate should you log out of Windows. This is a reasonable thing to do when just doing evaluations of the software. However, once CM+ is running in production, it is preferable to have permanent servers running, which can be automatically restarted after a system reboot. The NeumaServices.exe in the STS/bin directory is used to install Neuma Services.

Neuma Services provides a Windows capability which is essentially is used to "execute" server start-up and shutdown scripts. Script execution is simulated using Windows batch files to start or shut down the services. The fact that the execution is simulated means that each line is executed in sequence, without a "script environment" for script local variables.

There are two command scripts which are used to control these services. These are:

StsStart.bat - Starts the STS servers after a re-boot or after an StsStop operation.

StsStop.bat - Ensures that the Servers are Shut down and ready to be restarted.

Each of these batch files contain simple commands used to start and stop the servers. The example below assumes that the demo library needs a transaction server running and that floating licenses are being used so that the license manager is required.

StsStart.bat

```
del %STS%\demo\inservice
del %STS%\demo\shutdown.sts
start /min %STS%\bin\licserv.bat
start /min %STS%\bin\sts.bat demo -s
start /min %STS%\bin\txnserv.bat demo
```

StsStop.bat

```
echo shutdown > %STS%\demo\shutdown.sts
%STS%\bin\licserv –k
%STS%\bin\txnserv demo –k
```

If you had multiple libraries operating on your server, you would include all pertinent start-up and shutdown commands in these two files.

The Neuma Services start up after a system reboot and runs all of the commands in the StsStart.bat file. If it is necessary to shutdown the service, the StsStop.bat file is effectively executed. Because of the occasional need to shutdown the CM+ servers (e.g. moving to a new release) and the robustness of the servers, the service does not attempt to restart the processes on its own if they have been terminated.

The NeumaServices.exe image is used to install or remove the Neuma Services. With no parameters, the CM+ services will be installed and the services will be started up, according to the contents of the StsStart.bat file in the STS\bin directory (where STS is the STS root directory, a.k.a. the CM+ installation directory). With a single "-k" parameter, the services will be stopped using the StsStop.bat file. With "-k -u" the services will be stopped and uninstalled. Note that if a library server is involved in a long transaction, the servers may take a while to shut down.

Although you may wish to install a service to automatically start your CM+ servers on your server box, we recommend that you defer this step until you have completed verification of your environment.

Neuma Services Special Considerations

These notes cover special conditions with respect to starting NeumaServices.exe on Windows Server 2000/2003/2008 and Windows 7 operating systems. Note that Neuma's 32 bit binaries work on Win2008Server which is a 64 bit OS.

Windows Server 2000:

A user other than Administrator but with administrator privileges must be specified in the services.msc applet

Note: When adding or modifying Windows Users/Groups you must log on to that account before changes are recognized by the system.

Windows Server 2003:

Only the Windows Administrator account can initially start it. The default LocalSystem account fails as it does not have the rights to create the service.

Once the service has been created, it is added to the Windows Service Control Manager (SCM) database. Only then can LocalSystem start/stop the service via the services.msc applet.

Use the Windows MMC snap-in (services.msc) and specify the user Administrator. After that, LocalSystem can handle the automatic startup at system boot.

Windows Server 2008:

Similar to Win2003, use the services.msc applet and change the default user. The default LocalSystem account will fail and you must specify a user who belongs to the administrators group or the Administrator user.

Windows 7:

Similar to Win2003, use the services.msc applet and change the default user. The default LocalSystem account will fail and you must specify a user who belongs to the administrators group.

Neuma Servers on Unix Platforms

On Unix and Linux platforms these servers may be started manually or from the Unix boot initialization code. It is recommended that you use the "stsstart" and "stsstop" shell scripts, found in your STS/bin directory, to control start-up and shutdown of your libraries on Unix. Typical Unix commands for stsstart and stsstop are shown below.

E.G. Filename: STS/bin/stsstart

```
#!/bin/sh
# STS startup script (stsstart) DO NOT RUN AS ROOT.  RUN AS LIBRARY OWNER
# STS environment variable must first be defined or will be extracted from path of $0
if [ X$STS = X ]; then
# stsstart shell must be started with /bin/ explicitly (or by alias) in the path.
# (e.g. /serv/neuma/sts/bin/stsstart )
   LEN=`expr $0 : '.*/bin/' - 4`
   STS=`echo $0 | cut -c 1-${LEN}`
   export STS
fi
```

Filename STS//bin/stsstop

```
echo Stopping Active Servers...
sleep 5
### Clearing unwanted Demo Library Files  - REPEAT for each library
rm -f ${STS}/demo/inservice ${STS}/demo/shutdown.sts
cd ${STS}
echo Restarting Servers...
### License Server
${STS}/bin/licserv > licserv.log&
### Starting Demo Server (library and transaction servers)
${STS}/bin/sts demo -s
${STS}/bin/txnserv demo > txnserv.log&
### Start any other library servers here.
```