

Release version 3.0

A guide for the Plastic SCM Extensions for third party integration

Plastic SCM **Extensions**

© 2006-2010 Codice Software

Warning and Disclaimer

Every effort has been made to make this book as complete and as accurate as possible, but no warranty or fitness is implied. The information provided is on an “as is” basis. The author and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this book.

Trademarks

All terms mentioned in this book that are known to be trademarks or service marks have been appropriately capitalized. Codice Software cannot attest to the accuracy of this information. Use of a term in this book should not be regarded as affecting the validity of any trademark or service mark.

**Contents**

[1 Introduction 1](#_Toc263084695)

[2 Bugzilla Integration 2](#_Toc263084696)

[2.1 Introduction 2](#_Toc263084697)

[2.2 Configuration 2](#_Toc263084698)

[2.2.1 Server 2](#_Toc263084699)

[2.2.2 Client 3](#_Toc263084700)

[2.3 Characteristics 4](#_Toc263084701)

[2.3.1 “Task on branch” working mode 4](#_Toc263084702)

[2.3.2 Enforcing task existence 5](#_Toc263084703)

[2.3.3 “Task on changeset” working mode 6](#_Toc263084704)

[3 DevTrack integration 8](#_Toc263084705)

[3.1 Introduction 8](#_Toc263084706)

[3.2 Configuration 8](#_Toc263084707)

[3.2.1 Server 8](#_Toc263084708)

[3.2.2 Client 9](#_Toc263084709)

[3.3 Characteristics 11](#_Toc263084710)

[3.3.1 “Task on branch” working mode 11](#_Toc263084711)

[3.3.2 “Task on changeset” working mode 14](#_Toc263084712)

[4 Mantis integration 16](#_Toc263084713)

[4.1 Introduction 16](#_Toc263084714)

[4.2 Configuration 16](#_Toc263084715)

[4.2.1 Server 16](#_Toc263084716)

[4.2.2 Client 16](#_Toc263084717)

[4.3 Characteristics 17](#_Toc263084718)

[4.3.1 “Task on branch” working mode 17](#_Toc263084719)

[4.3.2 “Task on changeset” working mode 19](#_Toc263084720)

[5 Trac integration 22](#_Toc263084721)

[5.1 Introduction 22](#_Toc263084722)

[5.2 Configuration 22](#_Toc263084723)

[5.2.1 Server 22](#_Toc263084724)

[5.2.2 Client 23](#_Toc263084725)

[5.3 Characteristics 24](#_Toc263084726)

[5.3.1 “Task on branch” working mode 24](#_Toc263084727)

[5.3.2 “Task on changeset” working mode 26](#_Toc263084728)

[6 JIRA Integration 29](#_Toc263084729)

[6.1 Introduction 29](#_Toc263084730)

[6.2 Configuration 29](#_Toc263084731)

[6.2.1 Jira Server 29](#_Toc263084732)

[6.2.2 Plastic SCM Client 33](#_Toc263084733)

[6.3 Characteristics 34](#_Toc263084734)

[6.3.1 “Task on branch” working mode 34](#_Toc263084735)

[6.3.2 “Task on changeset” working mode 36](#_Toc263084736)

[6.3.3 Check in operations record 39](#_Toc263084737)

[6.4 Field mapping 39](#_Toc263084738)

[6.4.1 Mapping custom Jira fields 41](#_Toc263084739)

[7 OnTime integration 42](#_Toc263084740)

[7.1 Introduction 42](#_Toc263084741)

[7.2 Configuration 42](#_Toc263084742)

[7.2.1 Server 42](#_Toc263084743)

[7.2.2 Client 43](#_Toc263084744)

[7.3 Characteristics 44](#_Toc263084745)

[7.3.1 “Task on branch” working mode 44](#_Toc263084746)

[7.3.2 “Task on changeset” working mode 46](#_Toc263084747)

[8 Version One integration 49](#_Toc263084748)

[8.1 Introduction 49](#_Toc263084749)

[8.2 Configuration 49](#_Toc263084750)

[8.2.1 Server 49](#_Toc263084751)

[8.2.2 Client 49](#_Toc263084752)

[8.3 Characteristics 51](#_Toc263084753)

[8.3.1 “Task on branch” working mode 51](#_Toc263084754)

[8.3.2 “Task on changeset” working mode 53](#_Toc263084755)

[9 FogBugZ integration 56](#_Toc263084756)

[9.1 Introduction 56](#_Toc263084757)

[9.2 Configuration 56](#_Toc263084758)

[9.2.1 Server 56](#_Toc263084759)

[9.2.2 Client 56](#_Toc263084760)

[9.3 Characteristics 57](#_Toc263084761)

[9.3.1 “Task on branch” working mode 57](#_Toc263084762)

[9.3.2 “Task on changeset” working mode 59](#_Toc263084763)

[9.3.3 Checked in operations record 62](#_Toc263084764)

[10 Rally Integration 63](#_Toc263084765)

[10.1 Introduction 63](#_Toc263084766)

[10.2 Configuration 63](#_Toc263084767)

[10.2.1 Server 63](#_Toc263084768)

[10.2.2 Client 63](#_Toc263084769)

[10.3 Characteristics 65](#_Toc263084770)

[10.3.1 “Task on branch” working mode 65](#_Toc263084771)

[10.3.2 “Task on changeset” working mode 67](#_Toc263084772)

[11 Global extension configuration 69](#_Toc263084773)

[11.1 Creating a configuration file 70](#_Toc263084774)

[11.2 Create the global configuration folders on the server 70](#_Toc263084775)

[11.3 Move the configuration file from the client to the server 71](#_Toc263084776)

[11.4 Customize the global configuration 71](#_Toc263084777)

[11.5 Restart the Plastic SCM server 71](#_Toc263084778)

[11.6 Setup clients 71](#_Toc263084779)

# Introduction

This guide describes how Plastic SCM provides integration with diverse control tracking tools such as Bugzilla, DevTrack, Mantis, OnTime, Trac, Jira, VersionOne, FogBugZ and Rally.

Once an issue tracking system is configured, the user will be able to work with the extension and Plastic SCM seamlessly. This means that the user will be able to see the associated task information through Plastic GUI. Plastic SCM extensions allow two different working modes: “task on branch”, which links one task from the issue tracking system to a Plastic branch and allows to browse the associated issue on the issue tracking tool -where the full information is displayed-. The second working mode is “task on changeset”; in this case one or more tasks are linked to one or more changesets, and the information of the associated tasks are available from Plastic changeset view.

# Bugzilla Integration

## Introduction

This section describes the steps to follow in order to use Bugzilla integration with Plastic SCM. It also describes its advantages.

This extension is compatible with Bugzilla versions 2005, 2006 and 2007.

## Configuration

### Server

To setup the Bugzilla extension functionality, copy the script file ‘plastic.cgi’ (included on install\_path/client/extensions/bugzilla) on Bugzilla´s installation folder.

Depending on the server’s operating system, the first line of the script file may have to be changed:

#!/usr/bin/perl –wT

To:

#!c:\perl\bin\perl.exe –wT

### Client

To configure the Bugzilla extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select “Bugzilla”.

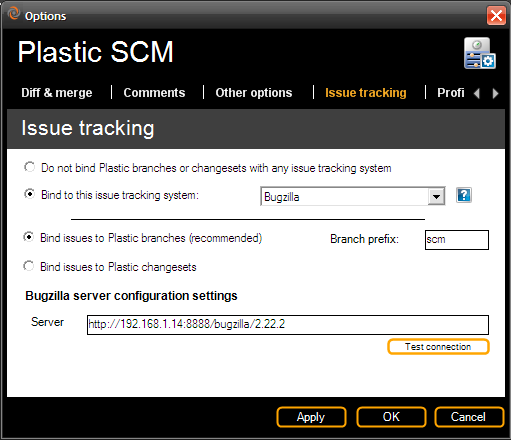


Figure 1: Issue tracking configuration: Bugzilla.

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work with this mode, a branch prefix must be specified. This branch prefix will be used to bind the Plastic branches to the bugzilla tasks. As the image above shows, the branches that start with prefix ‘scm’ will be linked to Bugzilla issues.

For instance: branch with name ‘scm012’ will be associated to Bugzilla task #12.

To work on “Task on Changeset mode” select the “Bind issues to Plastic changesets” radio button.

The “server” parameter specifies the URL where the Bugzilla server is installed and running.

## Characteristics

### “Task on branch” working mode

Using this working mode, every Bugzilla task is associated to a Plastic branch when creating a new one; the branch name for the new branch must match the configured extension prefix and the numeric identifier on Bugzilla.

Extended information of the branches will be displayed by accessing on the left hand side of the branch view. When a branch is selected on the branch view, the right pane will show the information of the associated task: its number, owner or developer the task has been assigned to, title and comments.

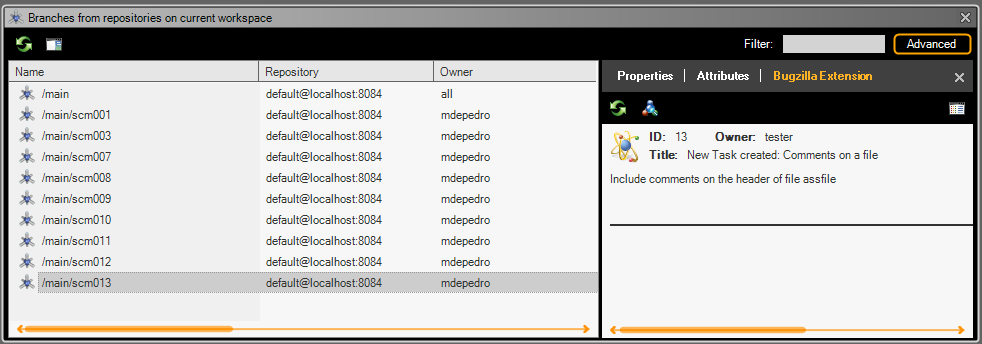


Figure 2: Branches extended information.

Performing a double click on the task pane, a new browser window will be opened and will display the full information of the task.

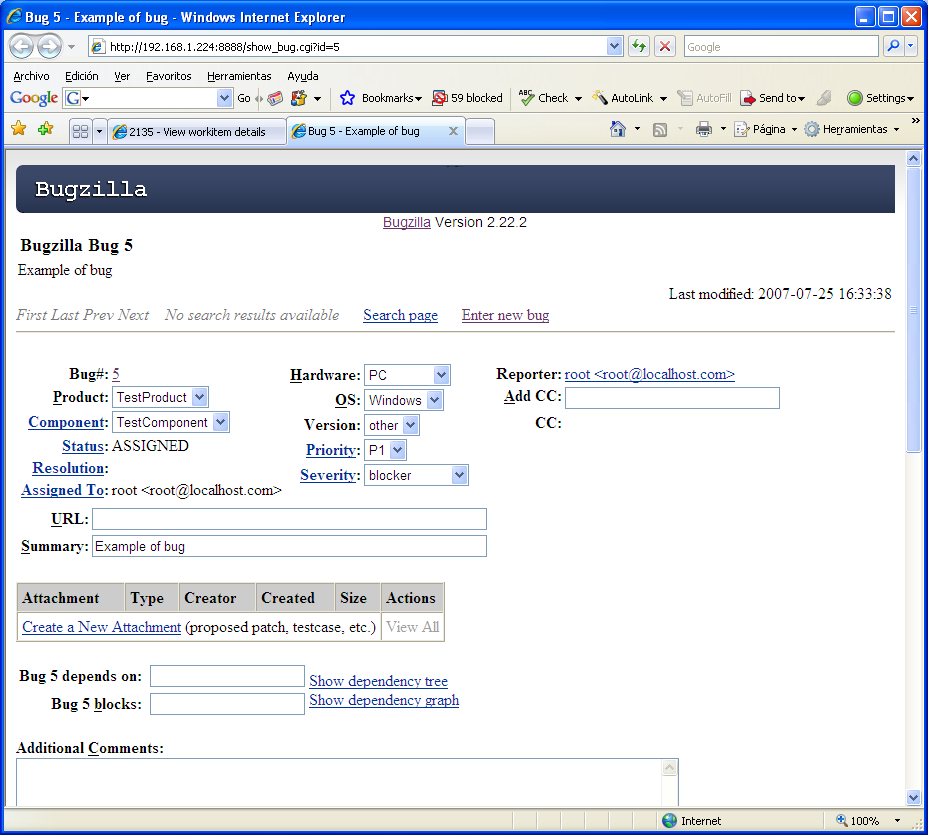


Figure 3: Bugzilla web interface example.

### Enforcing task existence

It is possible to setup Plastic SCM repositories so that numbered branches associated to tasks are checked against Bugzilla. Only if there is a bug number defined in Bugzilla the working branch will be allowed. This check is performed on the branch creation operation, only for child branches.

In order to setup the branch creation check on a repository, create an attribute named “plastic-enforce-task-branch” on the desired repositories, like in the following sample using the command line:

cm makeattribute plastic-enforce-task-branch

With this attribute created, each time a child branch is created on the repository where it is defined, the extension will check the name of the new branch. If it starts with the task prefix, it will then pick the number contained on it and ask Bugzilla if there is a matching bug number. Otherwise an error is printed and the operation is cancelled. This is a sample of the error that is displayed:

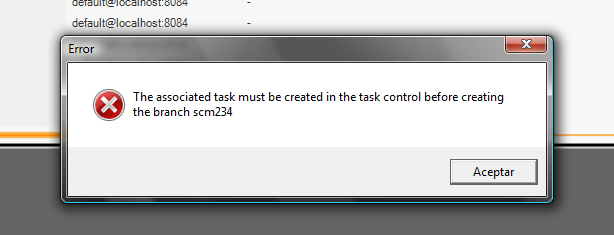


Figure 4: Trying to create a branch without its corresponding task.

### “Task on changeset” working mode

When using the “task on changeset” working mode, every time a user commits or checks in a set of changes, a new dialog window will be displayed to type the comments related to those changes and to link bugzilla tasks to the changes the check in carries with it:

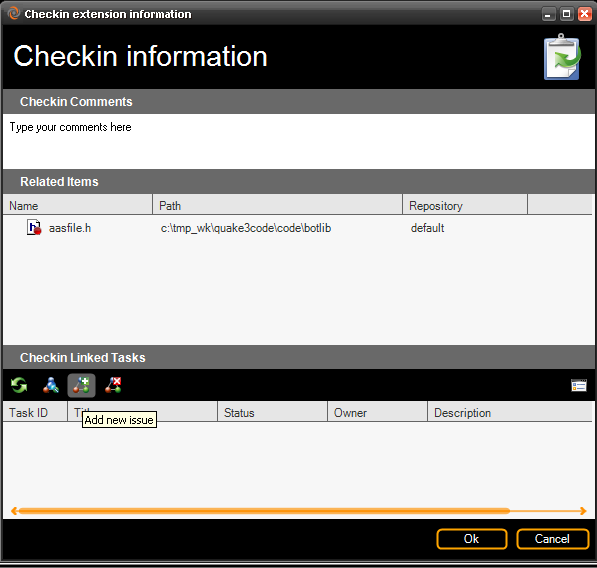


Figure 5: Checkin dialog when the task on changeset mode is configured.

The “Add new issue” option will show another dialog window. On this new dialog, as shows the image below, the user is able to choose the task to be linked to the changeset:

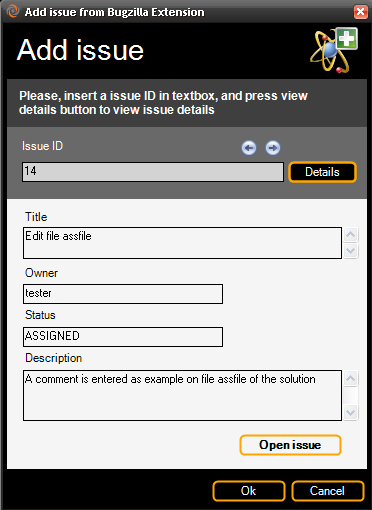


Figure 6: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

The changesets view will show information of the associated Bugzilla tasks. From this view, new tasks can be linked or unlinked to a certain changeset, and by clicking on the task a browser window will be opened displaying the full information of the task.

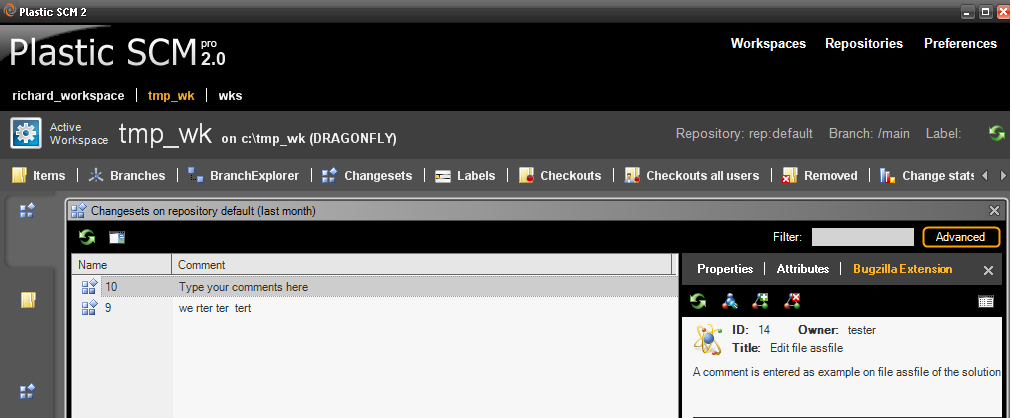


Figure 7: Changesets view, with detailed information when working in task on changeset mode.

# DevTrack integration

## Introduction

This section describes the steps to follow in order to use the DevTrack integration with Plastic SCM. It also describes its advantages.

This extension is compatible with DevTrack versions 6 and 7.

## Configuration

### Server

To setup the DevTrack extension functionality, the user has to access to the “DevTrack Admin”, select “Tool” and then the “Link Plus” option and the window that is shown on the following image will be displayed.

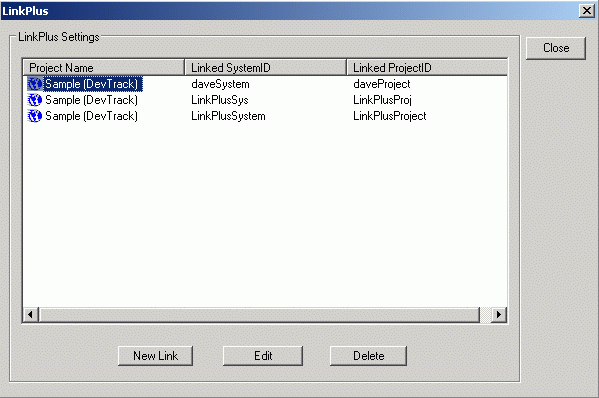


Figure 8: DevTrack server configuration.

From this window, click on the “New Link” option, and enter the following data:

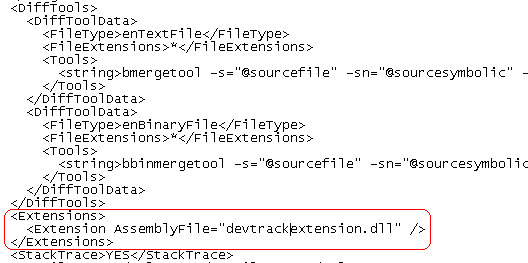
* TechExcel DevTrack Project Name: The name of the project to be created
* Linked System ID
* Linked Project ID

These last two parameters would be used from the client in order to access DevTrack.

### Client

In the first place it is necessary to copy ‘devtrackextension.dll’’ library to the folder where Plastic SCM client has been installed.

The user must add the following lines (circled in red) to the file ‘client.conf’ in order to indicate the client to use this extension.



Creating a file ‘devtrackextension.conf’ is also needed in order to specify the extension of the parameters that will interact with the server. This file will have the following appearance by default:

<DevTrackExtensionConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <DTLinkPlusUrl>http://192.168.1.237/LinkPlusWebService/WSDTIncident.asmx</DTLinkPlusUrl>

<User>terry-j</User>

<ProjectId>daveProject</ProjectId>

<SystemId>daveSystem</SystemId>

<BranchPrefix>issue</BranchPrefix> <DTBugInfoBrowserUrl>http://192.168.1.237/scripts/texcel/devtrack/buginfo.dll</DTBugInfoBrowserUrl>

</DevTrackExtensionConfiguration>

Take into account that Plastic SCM integration with DevTrack provides two different options depending on the working mode to be used; the one used by default is “task on branch” but it can be also specified as follows:

<DevTrackExtensionConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <DTLinkPlusUrl>http://192.168.1.237/LinkPlusWebService/WSDTIncident.asmx</DTLinkPlusUrl>

<User>terry-j</User>

<ProjectId>daveProject</ProjectId>

<SystemId>daveSystem</SystemId>

<BranchPrefix>issue</BranchPrefix> <DTBugInfoBrowserUrl>http://192.168.1.237/scripts/texcel/devtrack/buginfo.dll</DTBugInfoBrowserUrl>

<WorkingMode>TaskonBranch</WorkingMode>

</DevTrackExtensionConfiguration>

If the desired working mode is the “TaskOnChangeset” mode, the following “devtrackextension.conf” has to be configured its “WorkingMode” entry as follows:

<DevTrackExtensionConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <DTLinkPlusUrl>http://192.168.1.237/LinkPlusWebService/WSDTIncident.asmx</DTLinkPlusUrl>

<User>terry-j</User>

<ProjectId>daveProject</ProjectId>

<SystemId>daveSystem</SystemId>

<BranchPrefix>issue</BranchPrefix> <DTBugInfoBrowserUrl>http://192.168.1.237/scripts/texcel/devtrack/buginfo.dll</DTBugInfoBrowserUrl>

<WorkingMode>TaskonChangeset</WorkingMode>

<AttributePrefix>dev\_track</AttributePrefix>

</DevTrackExtensionConfiguration>

The meaning of the parameters on the file is as follows:

DTLinkPlusUrl -> Location of the LinkPlus web service as set up on Dev Track installation, which is also the location of the WSDTIncident.asmx file

Project ID -> The parameter set on the “Link Plus” information

System ID -> The one set on the “Link Plus” information

BranchPrefix -> Prefix will have the created branches on Plastic SCM that you want to associate with DevTrack issue. That is, if you want to create an associated branch with an issue 3 this branch should call, for example, SCM003

DTBugInfoBrowserUrl -> The BugInfo.dll URL located on DevTack path

AttributePrefix -> Only used on the “TaskonChangeset” working mode, on this mode an attribute is associated to the changesets.

Once every parameter has been set, Plastic SCM can begin to use DevTrack extension in a fully transparent way to the user.

## Characteristics

### “Task on branch” working mode

The working mode used by default is the “task on branch”, which is usually used to work with the “branch per task” pattern, on which for each task to be done, a new branch will be created. In order to use this working mode, the “devtrackextension.con” file must be set following the steps on the previous section: specifying this working mode or using the default setting which used “task on branch”.

In order to start working with Plastic SCM and DevTrack, the first step would be creating issues in DevTrack, which would be usually done by the Project Manager, by clicking on the “New” option on DevTrack the user can choose to submit a new feature or a new defect, depending on the nature of the new task to be submitted. The information to include would be title, type of task (bug, minor improvement, future enhancement or discrepancy report), description, developer assigned, etc as the following image shows:

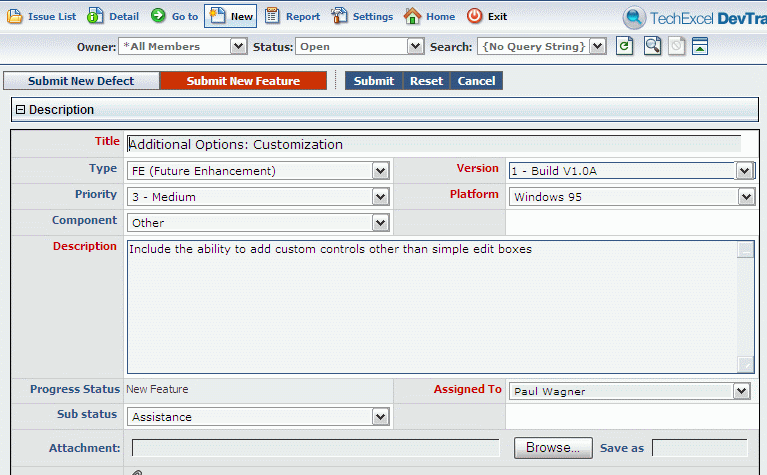


Figure 9: DevTrack user interface for creating new issues.

Once the new task is created, DevTrack assigns it a number; in this case it would be number 106 as the following image shows. Browsing the Issue List the user can find issues by developer, state, etc and they can be edited anytime.

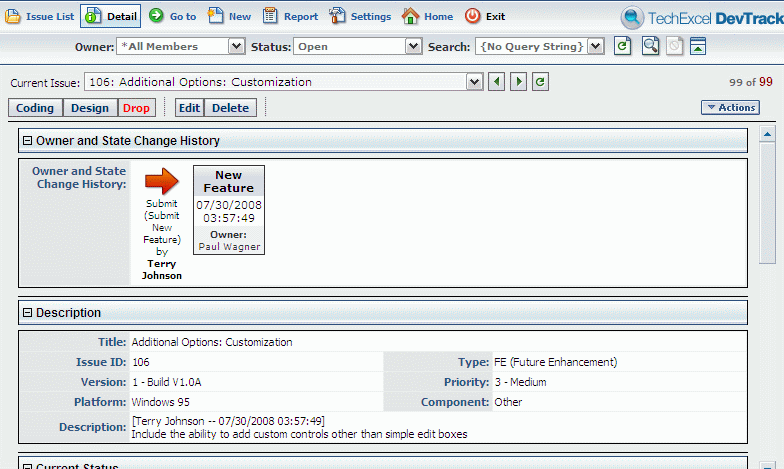


Figure 10: DevTrack user interface: detail of an issue.

The next step is that the developer who the task has been assigned to starts working on it. So, the developer creates a new branch (as simple as right clicking on the parent branch and select the “create child branch” option), and a new window will be opened, in which to include the information of the new branch: its name, in this case 106, as the DevTrack task with the prefix stated on the “devtrackextension.conf” (scm), comments related to the newly created branch and, as Plastic SCM branches are smart and remember their base, a base can be set on this dialogue, and every time a user has to change to that branch to work on it, it will automatically “remember” its base.

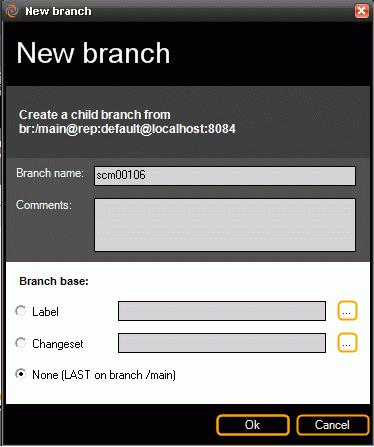


Figure 11: Creating a new branch for the task.

Now the user can always check the information of the issues just by going to the extended information on the branch view (on the left top side of the view), and select the branch: a dialog will be displayed on the right from which the DevTrack extension information is shown: ID number, owner, status, title, etc.

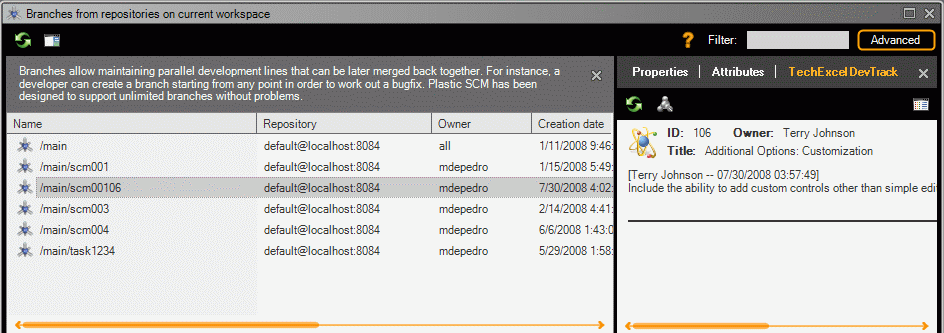


Figure 12: Branch view with extended information (task on branch mode).

Clicking on each of the issues, a new web browser window with the associated branch task will be opened. When the users change the status or any other field of the issue, the new information will appear by refreshing the DevTrack extension button.

### “Task on changeset” working mode

In the case that the DevTrack extension had been configured to use the “Task on changeset” working mode, the “devtrackextension.conf” must be configured as showed on the Client section. It must be indicated as the working mode by default is not “task on changeset” but “task on branch”.

In order to start working we must have created DevTrack tasks so that the developer assigned to those tasks can start working on them.

Once the developers has checked out the files he was to work with and the changes are submitted, the developer checks them in and an information dialog window will be displayed; on this dialog window users can include their comments on the check in operation as well as the issues associated from their task tracking control, DevTrack in this case. By using this working mode, one or more than one changeset in Plastic can be associated to one or more than one tasks in DevTrack. The dialog window appearance is shown on the following image:

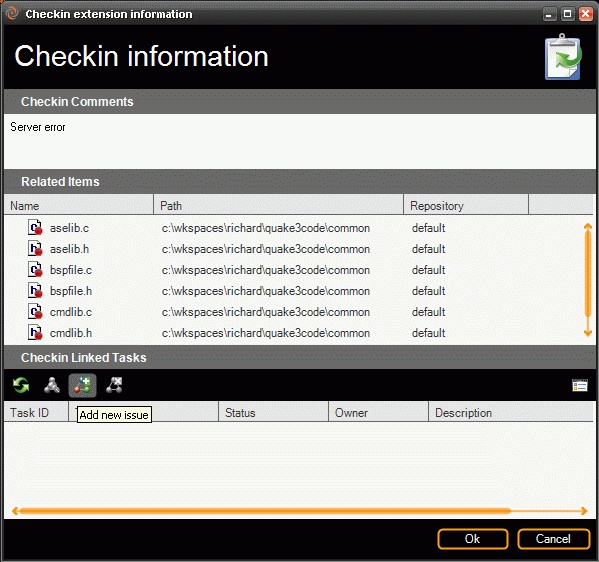


Figure 13: Checkin dialog in task on changeset mode.

By selecting the option “Add new issue” on this check in information dialogue, a new window will be displayed on which to include the ID or number of the DevTrack issue and by pressing on the “Details” button the information of the issue will be shown as the following image displays:

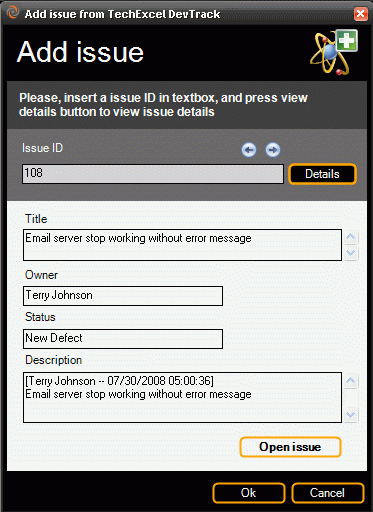


Figure 14: Checkin dialog in task on changeset mode.

Once the changesets have been linked to its corresponding issue or task in DevTrack, the extension information will be shown on the changeset view. From the DevTrack Extension information users can also add new issue or delete previous ones and by clicking on each of the issues associated to a certain changeset, a new browser window will be opened. Whenever a user modifies an issue in the task tracking tool, by refreshing the TechExcel DevTrack Extension information, it will be shown in Plastic.

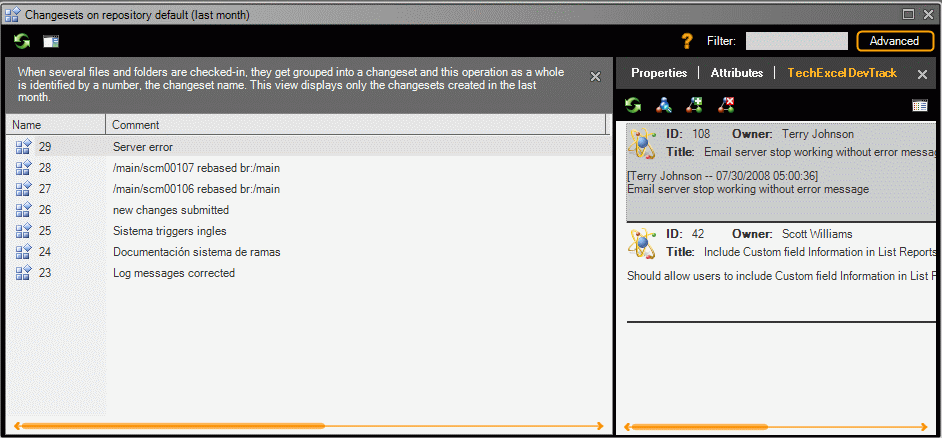


Figure 15: Changeset view in task on changeset mode, showing extended information

# Mantis integration

## Introduction

This section describes the steps to follow in order to use Mantis integration with Plastic SCM. It also describes its advantages.

This extension is compatible with Mantis versions 0.19.4, 1.0.0, 1.0.8 and 1.1.0a3.

## Configuration

### Server

To setup Mantis extension functionality, copy the script file ‘plastic.php’ (included on install\_path/client/extensions/mantis) on the Mantis installation folder.

### Client

To configure the Mantis extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select “Mantis”.

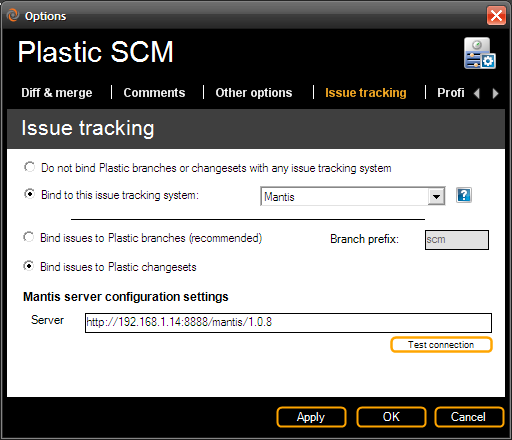


Figure 16: Plastic SCM Issue tracking configuration: Mantis

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work with this mode, a branch prefix must be specified. This branch prefix will be used to bind the Plastic branches to Mantis tasks. As the image above shows, the branches which start with prefix ‘scm’ will be linked to Mantis issues.

For instance: branch with name ‘scm022’ will be associated to Mantis task #22.

To work on “Task on Changeset mode” select the “Bind issues to Plastic changesets” radio button.

The “server” parameter specifies the URL where the Mantis server is installed and running.

## Characteristics

### “Task on branch” working mode

The working mode configured by default is the “task on branch”, which works fine with the “branch per task” pattern. On this working mode, for each task to be done a new branch will be created.

As usual, the first step would be creating issues in Mantis, this would be usually done by the Project Manager, by clicking on the “Report Issue” option a dialog on which to include the information of the new issue to be done will be opened, the information to include would be title, description, severity, etc.

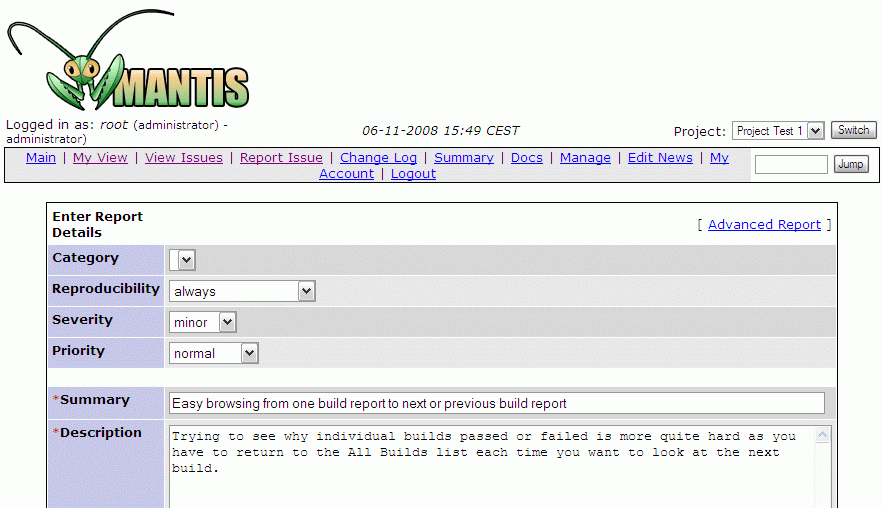


Figure 17: Mantis user interface to create a new issue

When the issue is created, Mantis assigns it a number. In this case, Mantis assigned the number 4 to the task, as shows the figure below. Users can select an issue from the list of issues -which are shown on different colors depending on their status- and change its status, so issue 4 goes from “new” to “assigned”.

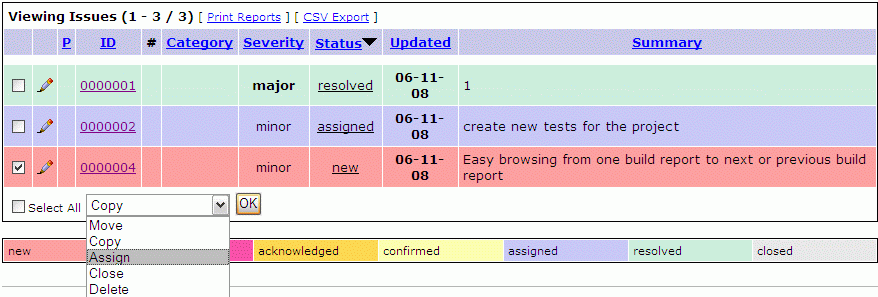


Figure 18: Mantis list of issues

The next step is that the developer who the task has been assigned to start working on it, so he creates a new branch (as simple as right clicking on the parent branch and select the “create child branch” option), as it is shown in the following image, and, in order to link it to the issue in Mantis, it must be given the same number, 4 in this case:

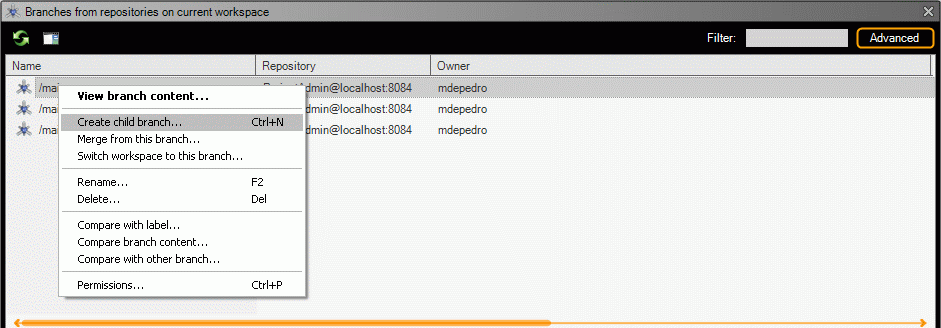


Figure 19: Creating a new branch in Plastic SCM

Now the user can check the information of the issues linked to a certain branch on the extended information on the branch view.

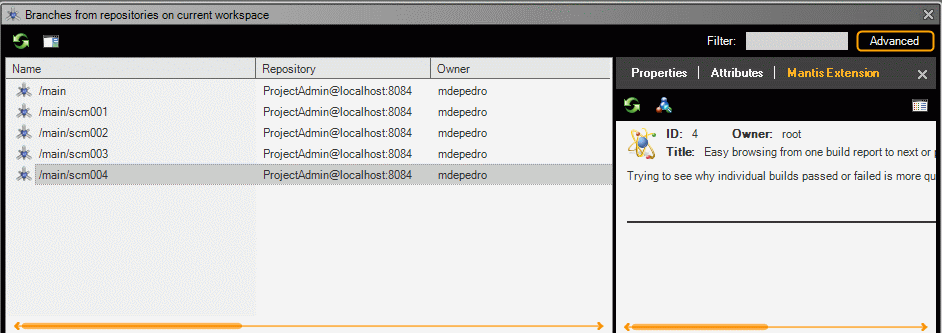


Figure 20: Plastic SCM extended information for branches when working in task on branch mode

If the user clicks on the issue pane, a browser window with the associated branch issue will be opened, when the users changes the status or any other field of the issue, the new information will appear by refreshing Mantis extension.

### “Task on changeset” working mode

To configure Plastic SCM to work on “task on changeset” working mode, the user has to configure the Mantis extension and select the “Bind issues to plastic Changesets” option, as explained before on the “Client” section.

When a developer submits some code changes and performs a check in operation, a new dialog window will be displayed. Through this dialog window, users can include their comments on the check in operation as well as the issues linked to this newly created changeset. By using this working mode, one or more than one changeset in Plastic can be associated to one or more than one issues in Mantis. The dialog window appearance is shown on the following image:

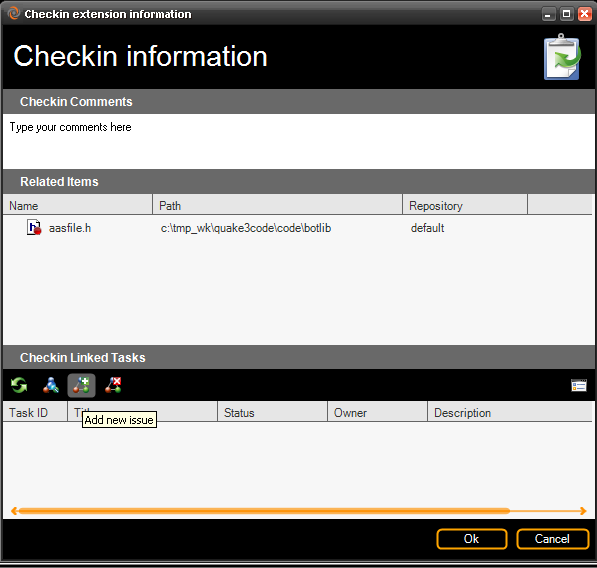


Figure 21: Checkin dialog when the task on changeset mode is configured.

The “Add new issue” option will show another dialog window. On this new dialog, as shows the image below, the user is able to choose the task or defect to be linked to the changeset:

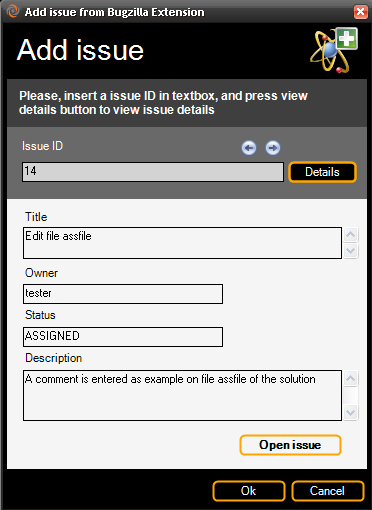


Figure 22: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

Once the changesets have been linked to the issue or task in Mantis, the extension information will be shown on the changeset view. From the Mantis Extension information pane users can also add new issue or delete previous ones and by clicking on each of the issues associated to a certain changeset, a new browser window will be opened. Whenever a user modifies an issue in Mantis, by refreshing the Mantis Extension information, it will be shown in Plastic SCM.

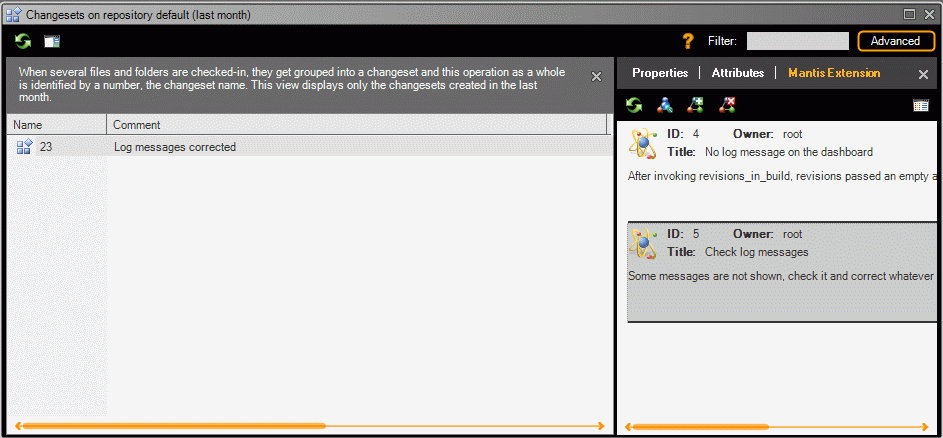


Figure 23: Changeset extended information in task on changeset working mode.

# Trac integration

## Introduction

This section describes the steps to follow in order to use Trac integration with Plastic SCM. It also describes its advantages.

This extension is compatible with Trac versions from 0.10.

## Configuration

### Server

To setup the Trac extension functionality, the XMLRPC plug-in must be installed into Trac server.

The following link: <http://trac-hacks.org/wiki/XmlRpcPlugin> provides information about how to download, install and verify that it has been properly set up.

Using Trac administration tool, enable the XML\_RPC permission on users which will use the extension. If this permission is assigned to user “anonymous” it will be available to every user.

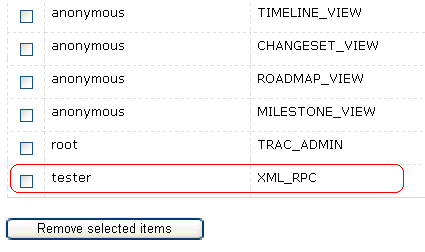


Figure 24: Trac server configuration.

When using the Trac extension, several Server configuration features must be taken into account:

* If the authentication method used is the Basic one (option --basic-auth when launching tracd), which means an authentication mode based on user/password, the URL to access XMLRPC services would be like this format: http://{trac\_server}/{trac\_project\_name}/login/xmlrpc
* If trac server is configured with any other authentication method, Plastic SCM extension cannot proceed with user authentication. In this case, user *anonymous* must have the XML-RPC permission enabled, and the URL to access to the services would be in the following format:  
  http://{trac\_server}/{trac\_project\_name}/xmlrpc   
  It will not be necessary to supply user/password authentication information to Plastic SCM Trac extension.

### Client

To configure the Trac extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select the “Trac” option.

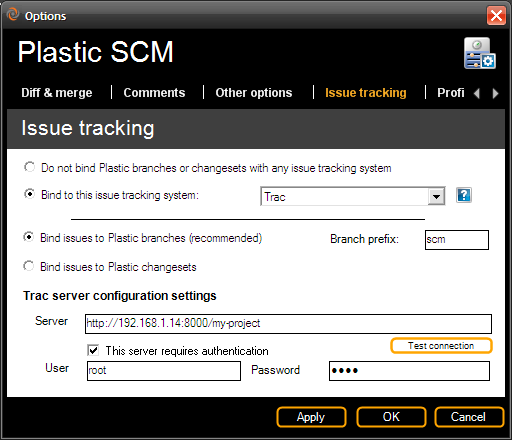


Figure 25: Plastic SCM issue tracking configuration: Trac.

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work with this mode, a branch prefix must be specified. This branch prefix will be used to bind the Plastic branches to the Trac tasks. As the above image shows, the branches that start with prefix ‘scm’ will be linked to Trac issues.

To work on “Task on Changeset mode” select the “Bind issues to Plastic changesets” radio button.

The “server” parameter specifies the URL where the Trac server is installed and running.

If the Trac server is configured on Basic authentication User/Password mode, enable the “This server requires authentication” checkbox and type the user and the password to authenticate against Trac server.

## Characteristics

### “Task on branch” working mode

The working mode configured by default is the “task on branch” mode, which works fine with the “branch per task” pattern. On this working mode, for each task to be done a new branch will be created.

As usual, the first step would be creating issues in Trac; this would be usually done by the Project Manager, by clicking on the “New Ticket” option a dialog on which to include the information of the new issue to be done will be opened, the information to include would be title, description, summary, etc.

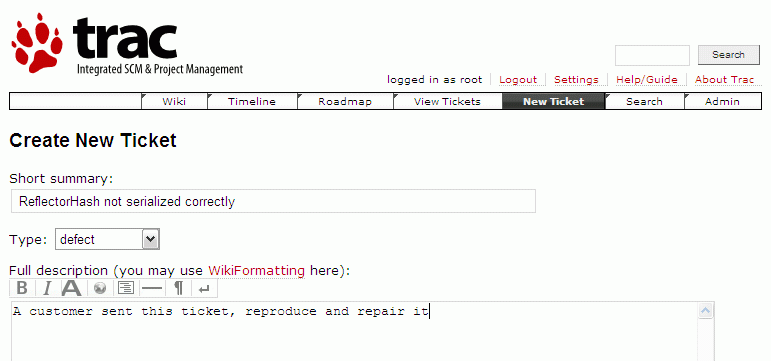


Figure 26: Creating new tickets in Trac

When the task or ticket is created, Trac assigns it a number, (number 3 on the example picture below).

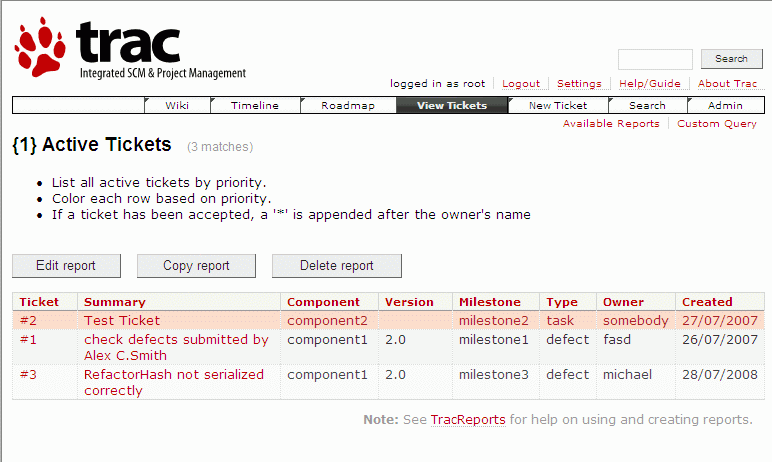


Figure 27: List of tickets in Trac.

The next step is that the developer the task has been assigned to starts working on it, so he creates a new branch (as simple as right clicking on the parent branch and select the “create child branch” option), as is shown on the image below. In order to link the new branch to the issue in Trac, its name must match the configured branch prefix for Trac extension and the Trac issue number (issue number 3 in this case). So, if branch prefix is set to ‘scm’, branch name ‘scm003’ will be linked to Trac ticket #3.

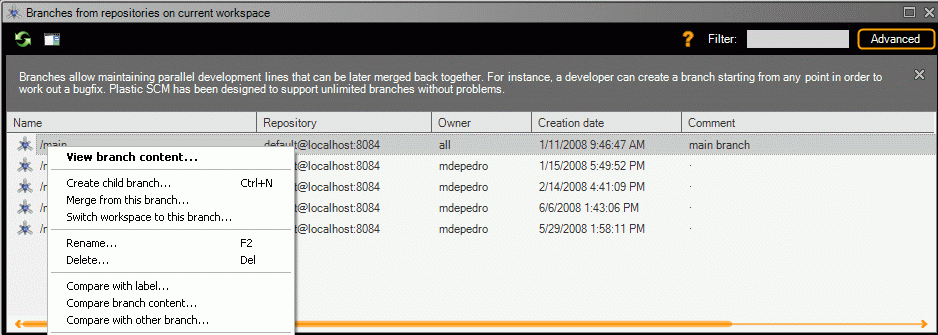


Figure 28: Creating new branches in Plastic SCM.

Extended information of the branches will be displayed by accessing on the left hand side of the branch view. When a branch is selected on the branch view, the right pane will show the information of the associated task/defect: its number, owner or developer the task has been assigned to, title and comments.

### “Task on changeset” working mode

To configure Plastic SCM to work on “task on changeset” working mode, the user has to configure the Trac extension and select the “Bind issues to Plastic Changesets” option, as explained before on the “” section.

When a developer submits some changes and performs a check in operation, a new dialog window will be displayed. Through this dialog window, users can include their comments on the check in operation as well as the issues linked to this newly created changeset. By using this working mode, one or more than one changeset in Plastic can be associated to one or more than one issue in Trac. The dialog window appearance is shown on the following image:

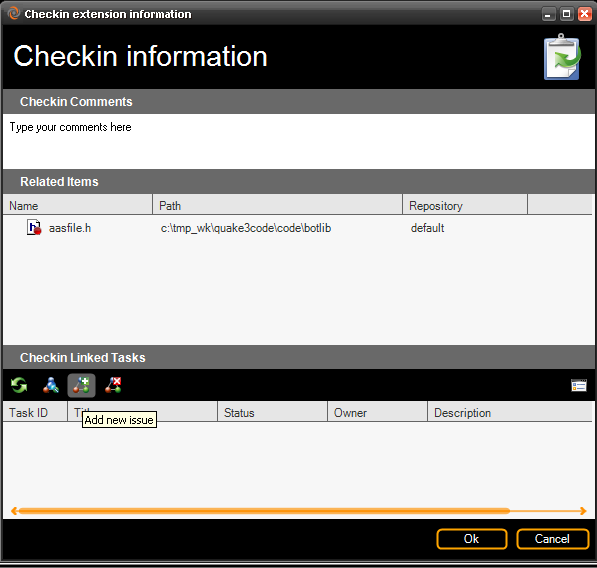


Figure 29: Checkin dialog when the task on changeset mode is configured.

The “Add new issue” option will show another dialog window. On this new dialog, as shows the image below, the user is able to choose the task or defect to be linked to the changeset:

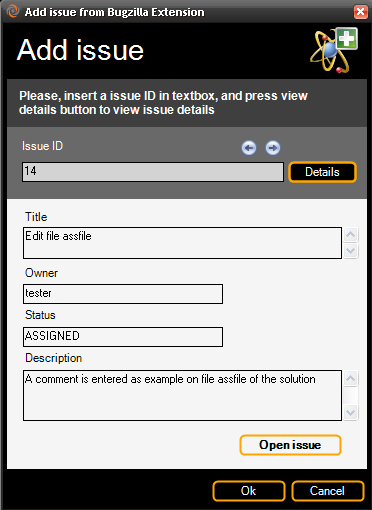


Figure 30: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

Once the changesets have been linked to the ticket or task in Trac, the extension information will be shown on the changeset view. From the Trac Extension information users can also add new tasks or delete previous ones and by clicking on each of the issues associated to a certain changeset, a new browser window will be opened. Whenever a user modifies an issue in the task tracking tool, the new data will be updated on Plastic GUI by pressing the “refresh” button on the Trac Extension pane.

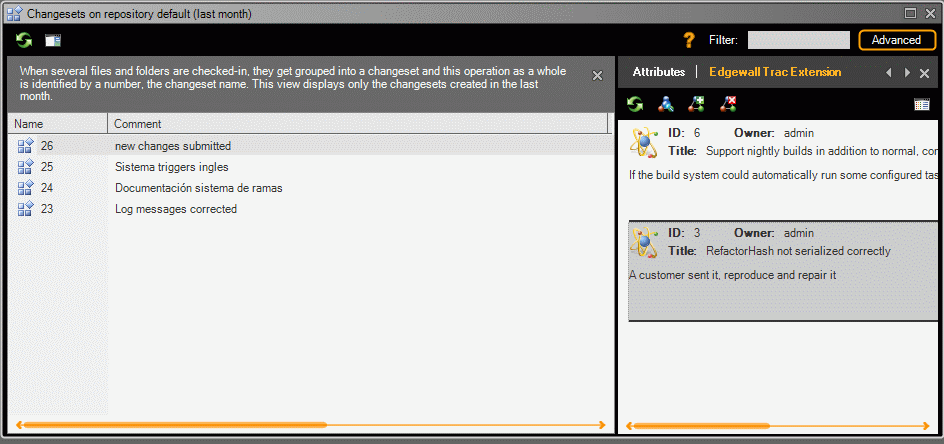


Figure 31: Changeset view with detailed information (task on changeset mode).

# JIRA Integration

## Introduction

This section describes the steps needed to setup Jira and Plastic SCM to work together.

## Configuration

### Jira Server

Plastic SCM will log the files and folders included in a changeset to their Jira issue. In order for feature to work, the Jira administrator will need to create a custom field. To create this custom field, go to the “Administration” interface and click on the “Custom fields” link inside the “Issue Fields” section on the left.



Figure 32: Issue fields options in Jira

Click on ‘Add Custom Field’.

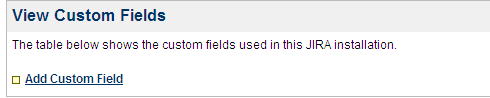


Figure 33: Adding a custom field in Jira

The first step is to setup the new field type. Select “Free text field”, allowing for unlimited text sixe. This is convenient since a task can affect many files and they need to be stored in this field. The log for check in operations on Jira is done by Plastic SCM using this Custom Field.

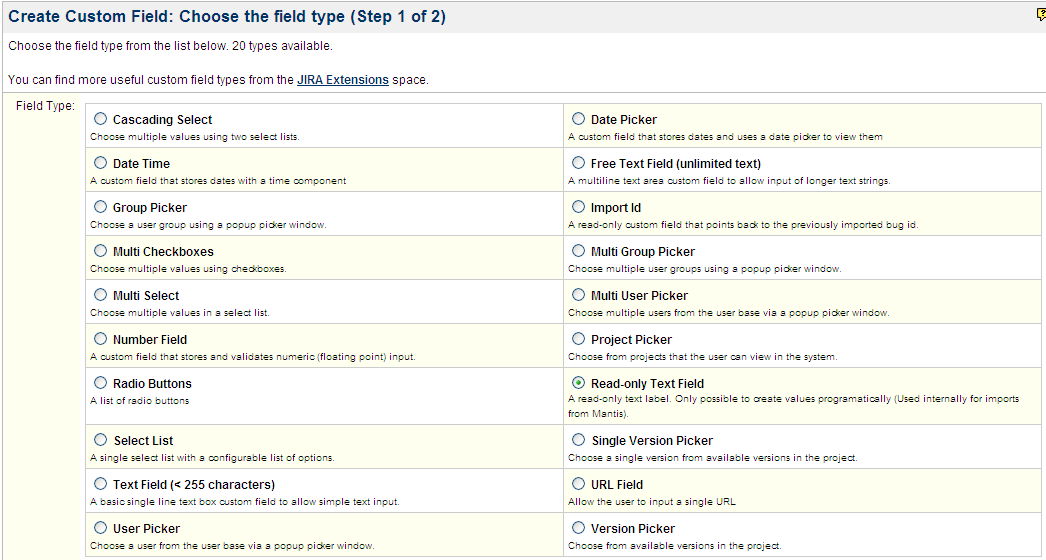


Figure 34: Types of custom fields in Jira

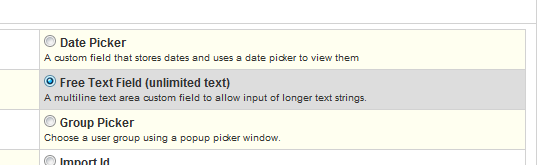


Figure 35: Specifing a free text field type

The next step is providing a name for the field. Any name will do, as Plastic will use the field Id and not the name to refer to the field. “Plastic SCM” can be a descriptive name:

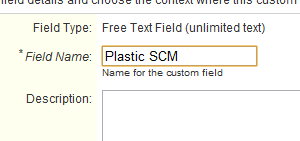


Figure 36: Name of the custom field.

When Jira asks for the association of the field to screens, choose the “Default Screen”.

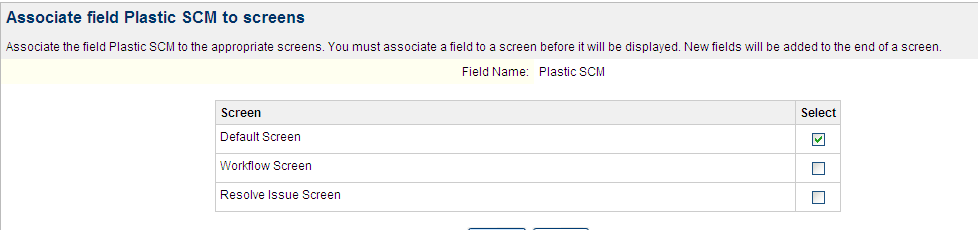


Figure 37: Specifying more options in Jira.

The new field is empty by default and will not show up in the configured screens until it gets some content.

Jira assigns an internal id to the custom field. This Id will be used to configure the link between Plastic SCM and Jira. It can be found in the URL, on the address bar of the screens page:





The field is now created and ready to be used:

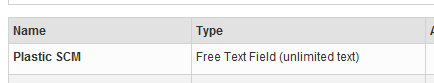


Figure 38: Plastic SCM custom text field

The last step is to ensure that Jira’s remote API is accessible by Plastic. Click on the “General Configuration” link in the “Global Settings” section:

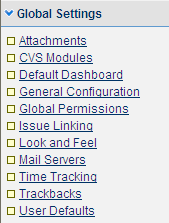


Figure 39: Global settings in Jira

Here, check that the ‘Accept remote API calls’ option is enabled:

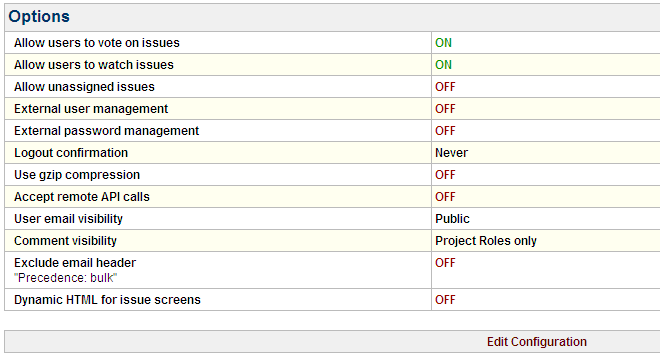


Figure 40: List of global settings in Jira.

In case it’s disabled, edit and enable it:





Once these steps have been fulfilled, the JIRA server is ready to accept Plastic SCM integration.

### Plastic SCM Client

To configure the Jira extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select “Atlassian Jira”.

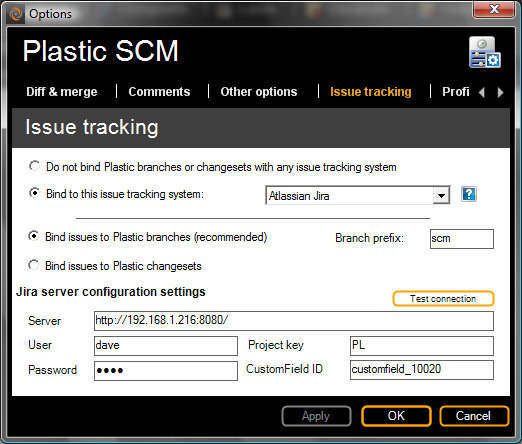


Figure 41: Issue tracking configuration in Plastic SCM: Jira.

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work in this mode, a branch prefix must be specified. This branch prefix will be used to bind the Plastic branches to the Jira tasks. As the image above shows, the branches that start with prefix ‘scm’ will be linked to Jira tasks.

To work on “Task on Changeset mode” select the “Bind issues to Plastic changesets” radio button. This mode will allow the user to select the task that the changes will be linked with at checkin time.

The “server” parameter specifies the URL where the Jira server is located. Note that the URL has to contain the port where the Jira server is running.

The “User” and “password” parameters specify the Jira credentials used to log into Jira and log the changes. Note that these fields can be the user credentials in Jira, or the credentials of a different Jira user specifically created to log Plastic SCM changes. The issue history in Jira will display the change as coming from the user configured here.

The “Project Key” parameter specifies the key of the Jira project whose issues will be related to Plastic Branches.

The “Custom Field ID” is the identifier of the custom field to be used for logging the Plastic check in information on Jira tasks. This field is built adding the literal “customfield\_” and the custom field id that Jira assigned when creating the field.



## Characteristics

### “Task on branch” working mode

The working mode configured by default is the “task on branch” mode, which works fine with the “branch per task” pattern. On this working mode, for each task to be done a new branch will be created.

To start working with Plastic SCM and Jira extension, the tasks must be created on the issue tracking system by clicking on the “Create New Issue” option and typing the data related to the new issue, such as name, associated project, summary, etc.

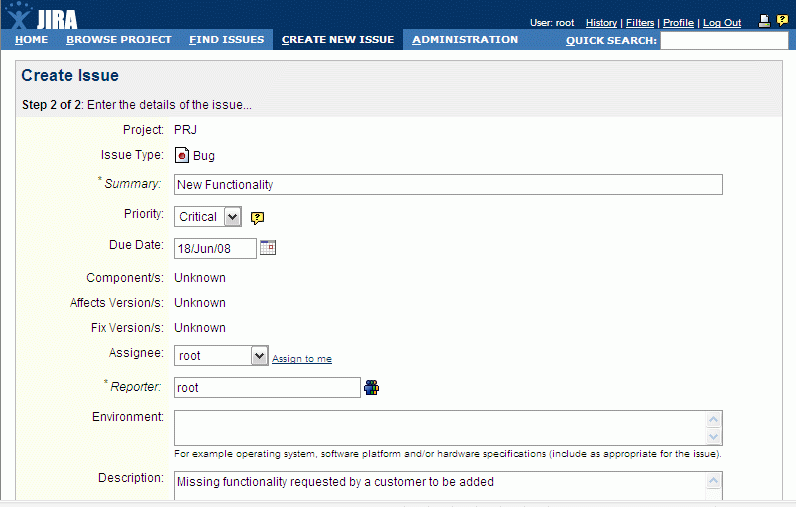


Figure 42: Creating new issues in Jira

Jira will assign a number to the newly created issue, which will be the same number given to the Plastic branch when working on that task, in this case we create issue number 3.



Figure 43: Detail of an issue in Jira

Then, the developer who has been assigned to that issue starts working on it. She will create a new branch using the Plastic SCM GUI client. Note that the name of the new branch must match the number of issue on Jira (number 3 in this case), with the scm prefix, as configured on the Plastic SCM Jira extension.

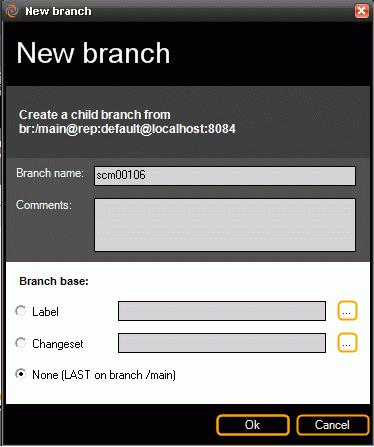


Figure 44: Creating a new branch for the task.

By clicking on the “Show extended information” button of the branches view the information associated to the branches will be displayed on the Plastic GUI´s right side, as in the figure below. By selecting a branch, The Jira issue information is displayed in Plastic SCM:

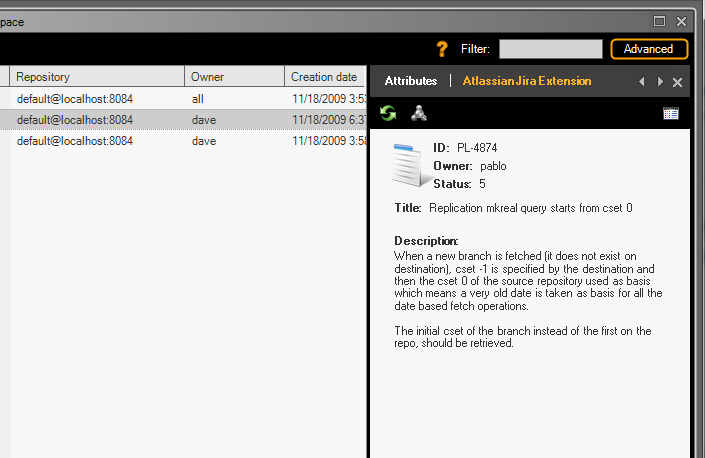


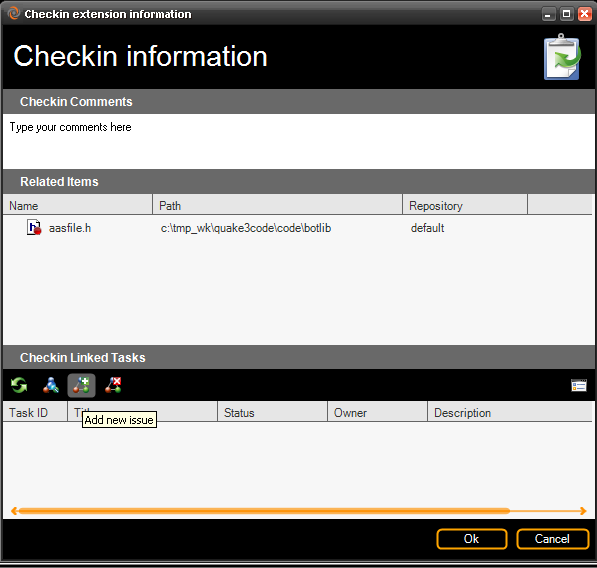
Figure 45: Detailed information of a task associated with a branch

By clicking on it, a new browser window with the associated branch issue is opened, which will display the full information of the Jira task. When the user changes the status or any other field of the issue, the new information will be shown by clicking on “Refresh” button of Jira Extension.

### “Task on changeset” working mode

The Jira setup is the same to work either in “Branch per Task” or “Task on Changeset” models. The use the later, the Plastic SCM client needs to be told so, in the Preferences dialog, under the Issue tracking tab.

The workflow will resemble something like this:

* Issues are created in Jira. When a developer starts working on one issue, it’s opened.
* Changes are made on the code, thus checked out items appear on Plastic.
* When the user decides to check in the changes, Plastic SCM will display a new dialog where the issues to be linked with those changes can be selected:
* 
* Figure 46: Checkin dialog when the task on changeset mode is configured.

This dialog allows the user to specify the check-in comments, displays a list of the items about to be checked in, and offer the option the link to the Jira issues by clicking the “Add new issue” button.

In this case, a new dialog appears, where an issue id can be entered, and the details for that issue are shown so the user can check whether this is the correct issue to link. The left and right arrow buttons navigate to the previous and next issues respectively.

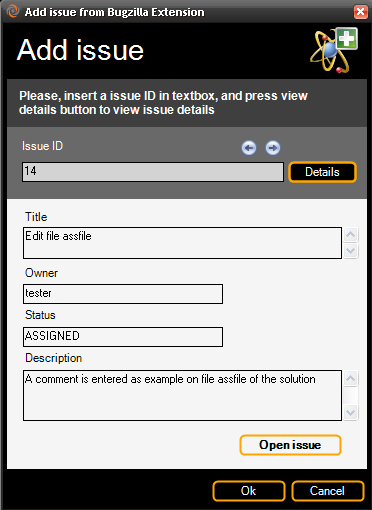


Figure 47: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

Once the changeset has been linked to the issue or task in Jira, this information can be displayed in Plastic SCM’s changesets view. When the “extended information” panel is visible (again, clicking the “Show extended information” button on the left of the toolbar), clicking in a changeset shows the associated Jira issues.

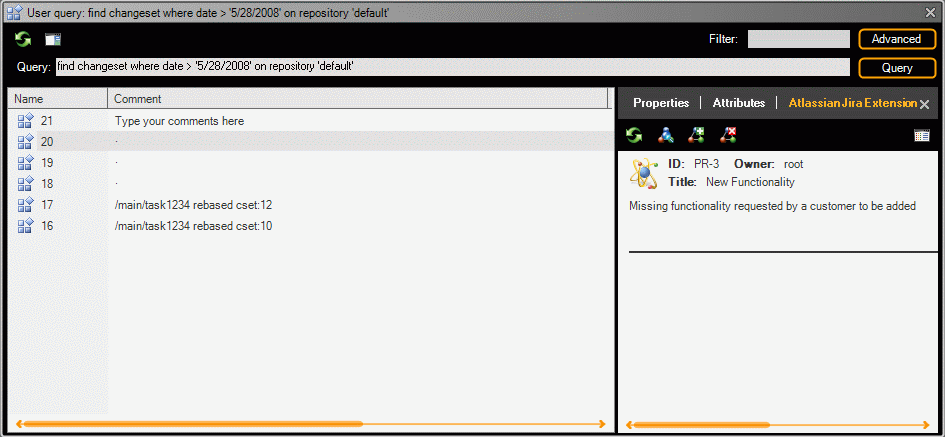


Figure 48: Changeset view with detailed information in task on changeset working mode

### Check in operations record

Besides the ability to display Jira data bound to branches or changesets inside Plastic SCM, the integration also keeps a record of the modified items inside Jira. When changes have been checked in and the integration is active, Jira will display the affected items in the “Plastic SCM” custom field:

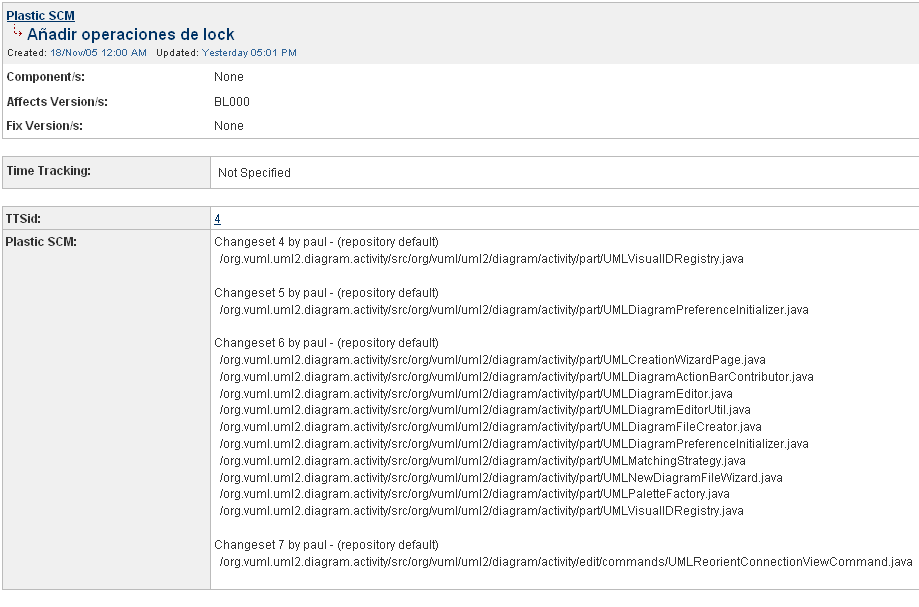


Figure 49: Detail of a issue in Jira, having checkin information generated by Plastic

## Field mapping

The Jira extension allows to map the fields from the Jira issue that will be displayed on the task information panel in Plastic SCM in the branches view or the branch explorer.

To define the field mapping a new section called FieldMappings is added in the Jiraextension.conf file:

<?xml version="1.0"?>

<JiraExtensionConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

**<FieldMappings>**

**<Map from="issue.reporter" to="Owner" />**

**<Map from="issue.resolution" to="Description" />**

**</FieldMappings>**

<BranchPrefix>scm-</BranchPrefix>

<WorkingMode>TaskOnBranch</WorkingMode>

<AttributePrefix />

...

In this sample, the field “Owner” in the Plastic GUI will be taking its value from the reporter field in the Jira issue and the Description field in Plastic takes its value from the “resolution” field in the Jira issue.

The “to” attribute in the “Map” entry describes which Plastic field the map refers to. This is the list of available Plastic fields:

|  |
| --- |
| Id |
| Owner |
| Status |
| Title |
| Description |



The “from” attribute describes the field in the Jira issue whose value will be read into the field “to” as described above. The list of available fields for the “from” attribute in the Jira extension are detailed in this table:

|  |  |
| --- | --- |
| **“from” field value** | **Field data type** |
| issue.assignee | String |
| issue.attachmentNames | String[] |
| issue.components | RemoteComponent[] |
| issue.created | DateTime |
| issue.description | String |
| issue.duedate | DateTime |
| issue.environment | String |
| issue.key | String |
| issue.priority | String |
| issue.project | String |
| issue.reporter | String |
| issue.resolution | String |
| issue.Status | String |
| issue.summary | String |
| issue.type | String |
| issue.updated | DateTime |
| issue.votes | Long |

### Mapping custom Jira fields

It’s possible to access the value of custom Jira fields with the customFieldValues modifier, like in this sample mapping:

<FieldMappings>

**<Map from="customFieldValues[10020]" to="Description" />**

</FieldMappings>

Notice that in this case there is no need to indicate “issue.” before the field name. This is because custom field values use a different mechanism to retrieve their values. The index between [ ] here is the custom field id that can be obtained from Jira as described in the previous sections. Only the custom field id number needs to be provided, in the sample 10020.

# OnTime integration

## Introduction

This section describes the steps to follow in order to use Axosoft OnTime integration with Plastic SCM as well as its advantages.

This extension is compatible with Axosoft OnTime versions 2005, 2006 and 2007.

## Configuration

### Server

To setup the OnTime extension functionality, the used version must have Ontime 200x WebServer and Ontime 200x SDK installed. The first one is necessary to open the data related to a bug through a web browser. The second one is necessary to allow third parties, like Plastic SCM to retrieve information regarding an issue.

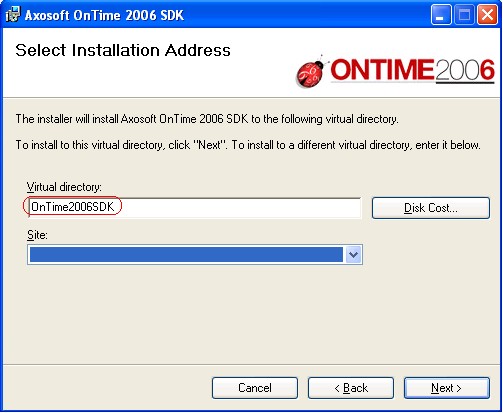
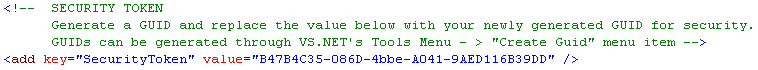


Figure 50: Axosoft OnTime server configuration

Another configuration parameter needed by the Plastic SCM client extension is the SecurityToken. This is a GUID specified on the OnTime 200x SDK configuration file ‘Web.config’ (located by default in: C:\Inetpub\wwwroot\OnTime200xSDK).





For more information about Ontime 200x WebServer and Ontime 200x SDK configuration, please refer to the manuals included on the OnTime installation.

### Client

To configure the OnTime extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select the “Axosoft OnTime” option.

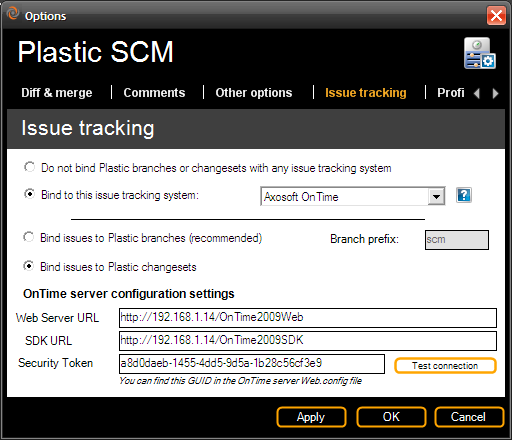


Figure 51: Issue tracking configuration: Axosoft OnTime

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work with this mode, a branch prefix must be specified. This branch prefix will be used to bind the Plastic branches to the OnTime tasks. As the image above shows, the branches that start with prefix ‘scm’ will be linked to OnTime tasks.

To work on “Task on Changeset mode” select the “Bind issues to Plastic changesets” radio button.

As explained before on the ‘Server’ topic, Plastic SCM Ontime extension needs two OnTime services to work. The first one is the “OnTimeWeb Server”, which allows the user to open a web browser to access to the full information of an OnTime task. Type in the “Web Server URL” field the URL of the ‘OnTime Web’ service where is running.

Type on the “SDK URL” the URL of the ‘OnTime SDK’ service where is running.

The “Security Token” field must be filled with the security token generated by the OnTime SDK installation. See ‘Server’ topic for more info.

## Characteristics

### “Task on branch” working mode

The working mode configured by default is the “task on branch” mode, which works fine with the “branch per task” pattern. On this working mode, for each task to be done a new branch will be created.

To start working with Plastic SCM and its OnTime extension, the defects must be created the defects on the issue tracking system: To do that, click on the “Defects” option and “New” menu, as the image below shows:

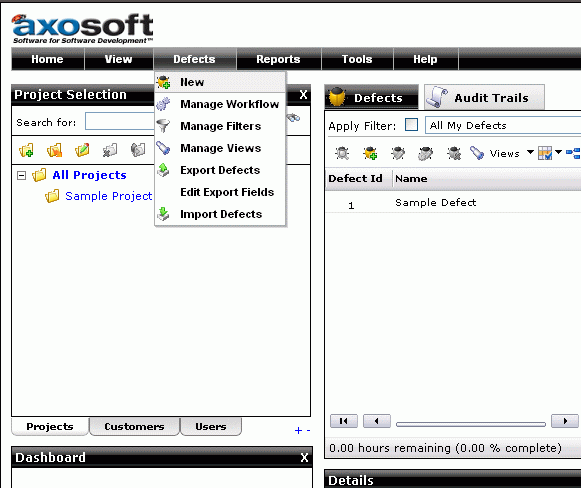


Figure 52: Creating new defects in OnTime

On the “New Defect” window, information such as name of the new task, priority, status, engineer assigned, etc must be filled. Once the new defect is saved, OnTime will give it an identification number, which will be used by the assigned developer when creating the Plastic SCM branch to work on that task.

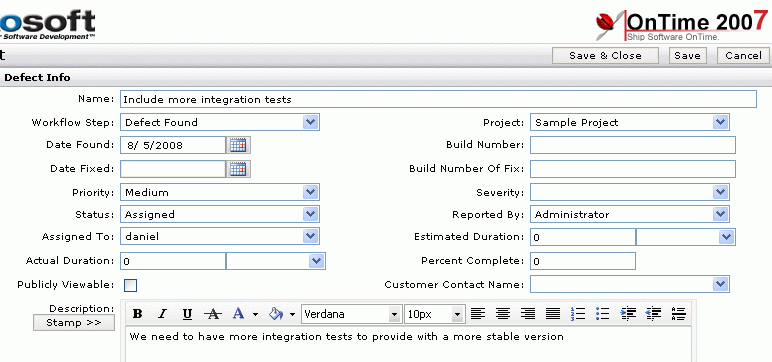


Figure 53: Defect info for a new defect in OnTime

Then, the developer who has been assigned to that issue starts working on it will create a new branch through Plastic SCM GUI client. Note that the name of the new branch must match the configured branch prefix for Plastic SCM OnTime extension plus the issue number assigned by OnTime to be linked.

By clicking on the left hand side of the branch view (“show extended information”), the information associated to the branches will be displayed on the GUI´s right hand side as the following screenshot shows. Selecting a branch the detailed information of the OnTime defect will be shown: name, title, comment and owner.

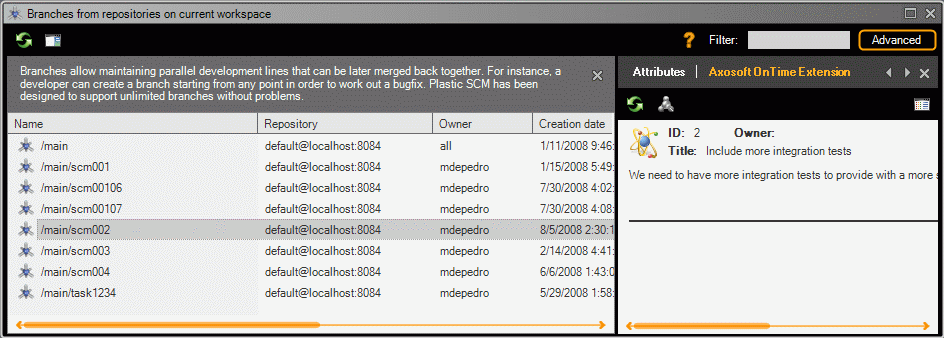


Figure 54: Detailed information of branches in task on branch mode

By clicking on it, a new web browser window with the associated branch task is opened, which will display the full information of the OnTime task. When the user changes the status or any other field of the issue, the new information will be shown by clicking on “Refresh” button of the OnTime Extension.

### “Task on changeset” working mode

To configure Plastic SCM to work on “task on changeset” working mode, the user has to configure the OnTime extension through PlasticGUI ‘Preferences’ window and select the “Bind issues to plastic Changesets” option, as explained before on the “” section.

Once the defects are created in OnTime, as showed on the previous working mode section, the developer will normally work on his changes in Plastic and when checking them in, a new dialog window will be displayed. On this dialog users can include their comments as well as the tasks or issues associated from their task control tool, OnTime in this case. Take into account that by using this working mode, one or more than one issue can be associated to one or more than one changesets:

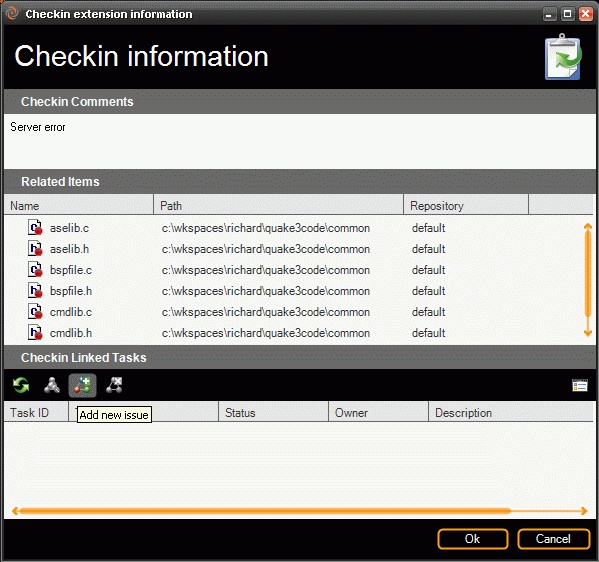


Figure 55: Checkin dialog in task on changeset mode.

On this dialog, by selecting the “Add new issue” option showed on the previous figure, a new dialog will be displayed. Enter the numeric issue identifier on this window to link the OnTime task to the new changeset:

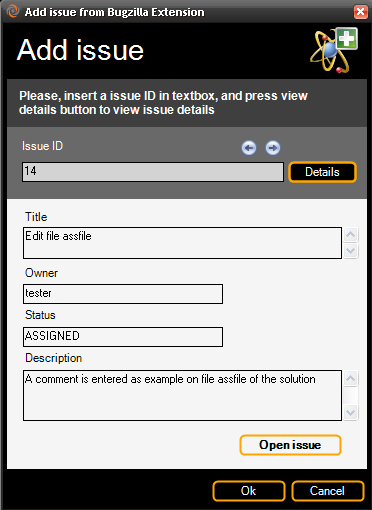


Figure 56: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

Once the changesets have been linked to the OnTime defect or task, the extension information will be shown on the changesets view. Users can also add new tasks or delete previous ones through the OnTime extension pane; and by clicking on each issue a web browser window with the associated issue in OnTime will be opened. Whenever a user modifies an issue in the task tracking tool, the new data will be updated on Plastic GUI by pressing the “refresh” button on the OnTime Extension pane.

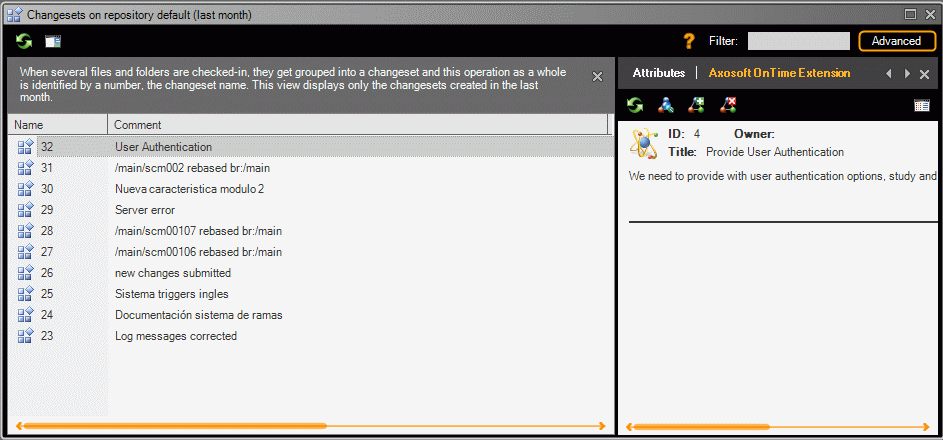


Figure 57: Changeset detailed view (task on changeset working mode).

# Version One integration

## Introduction

This section describes the steps to use the Version One integration with Plastic SCM. It also describes its advantages.

This extension is compatible with Version One version 7.2 and following ones.

## Configuration

### Server

It is not necessary to make any changes on the server side for Plastic SCM Version One extension.

### Client

To configure the Version One extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select the “Version One” option.

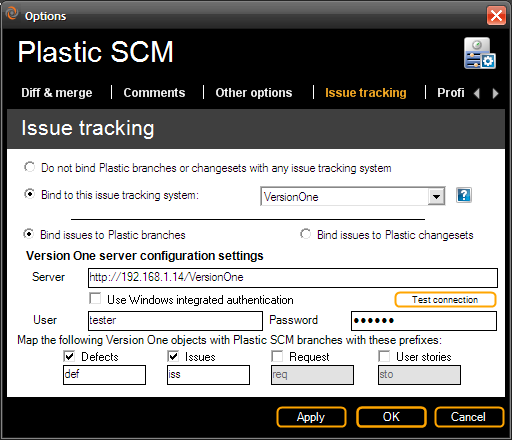


Figure 58: Issue tracking configuration in Plastic SCM: Version One

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work with this mode, the user will be able to configure several branch prefixes to bind them to several Version One objects. For instance, if the user enables the mapping between Version One Defects, branches that start with prefix ‘def’ will be linked to Version One Defects. Branch name ‘def1001’ will be linked to Version One Defect ‘D-01001’. The rest of the objects have the same behaviour. Available Version One objects to be linked are Defects, Issues, Requests and User Stories.

**Important:** Note that Version One starts enumerating their objects from number 1000.

To work on “Task on Changeset mode” select the “Bind issues to Plastic changesets” radio button. In this case, all the Version One objects will be enabled to map them to a Plastic changeset.

The “Server” parameter specifies the URL where the Version One server is installed and running.

Regarding authentication on Version One server, the “Use windows integrated authentication” checkbox tells Plastic SCM whether the authentication will be resolved with Windows domain users or not.

If this option is not selected, the “User” and “Password” fields must be filled with a valid Version One user.

## Characteristics

### “Task on branch” working mode

The working mode by default is “task on branch”, used for the “branch per task” pattern, on which for each task a new branch will be created.

As usual, the first step would be creating tasks in Version One, this would be usually done by the Project Manager, by clicking on the ”Add Defect”, “Add Issue”, “Add backlog Item” and “Add Request” shortcuts on Version One web interface. A new dialog window will be opened to type the data of the new task. Version One assigns numbers to tasks starting on 1000 and up. This is very important to remark, because branch names have to contain the configured prefix for that type of task and the assigned number by Version One.

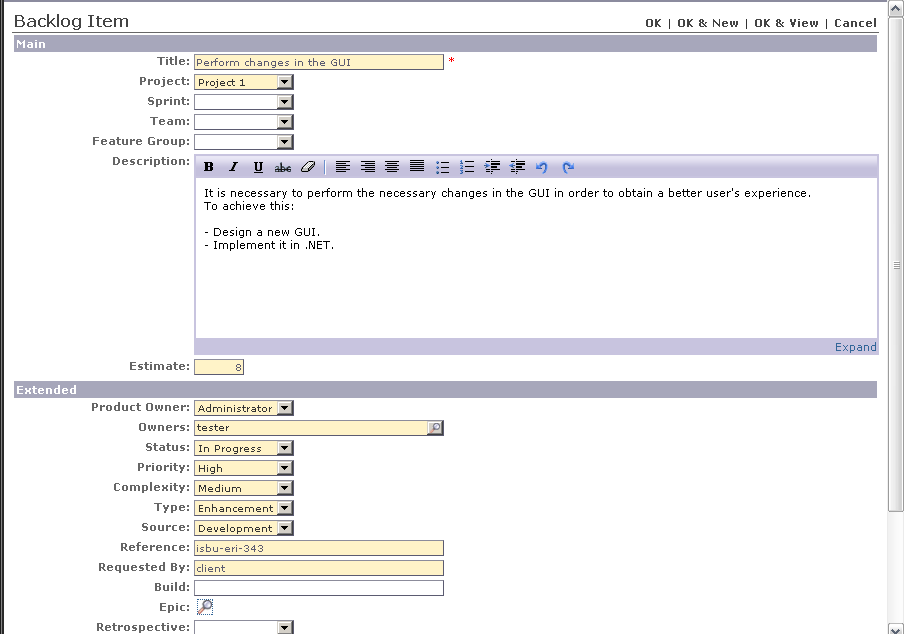


Figure 59: Creating a new backlog item in Version One

When the issue is created, Version One assigns it a number. In the following figure we can see a list of items created:

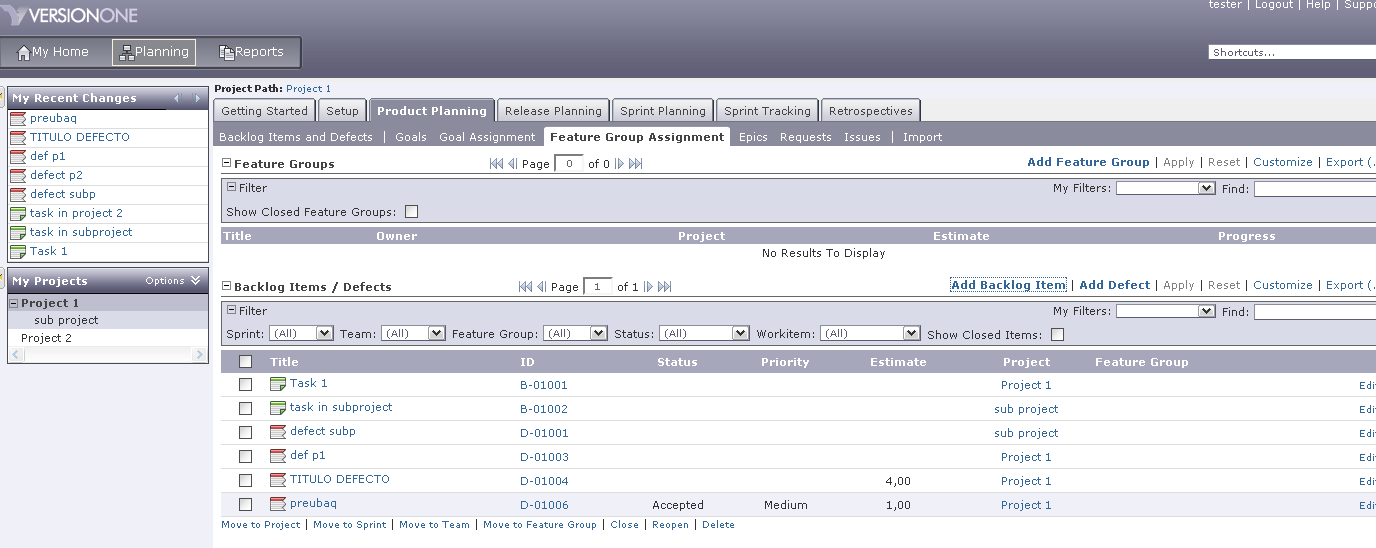


Figure 60: List of defects in Version One

The next step is: when a developer has been assigned to a task, the developer has to create a new branch (as simple as right clicking on the parent branch and select the “create child branch” option), as the following image shows, and, in order to link it to the task in Version One, the branch must be given the same number (remember: this number always would be upper than 1000):

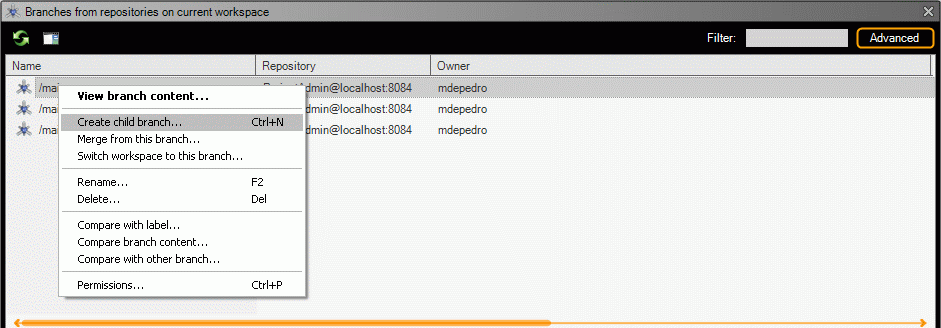


Figure 61: Creating a new branch in Plastic SCM to resolve the new issue

Now the user can check the information of the issues just by going to the extended information on the branch view and selecting the branch: a new pane will be displayed on the right side of the view from which the Version One extension information is shown: task number, owner, status, etc.

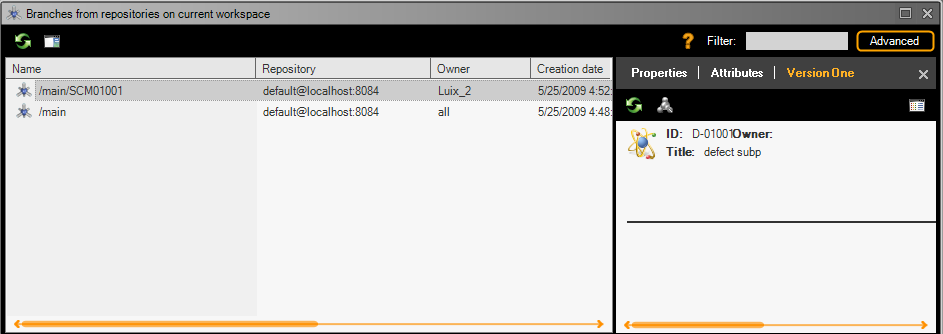


Figure 62: Detailed branch information in task on branch mode

Performing a double click on the task associated with a branch, a new web browser window will be opened. When a user changes the status or any other field of the issue, the new information will appear by refreshing Version One extension.

### “Task on changeset” working mode

To configure Plastic SCM to work on “task on changeset” working mode, the user has to configure the Version One extension through Plastic GUI ‘Preferences’ window and select the “Bind issues to plastic Changesets” option, as explained before on the “” section.

In order to start working we must have created Version One tasks so that the developer assigned to those tasks can start working on them. Once the changes are submitted and the developer checks them in, a new dialog window will be displayed; on this window users can include their comments on the check in operation as well as the issues associated from their task tracking control, Version One. By using this working mode, one or more than one changeset in Plastic SCM can be associated to one or more than one defects, issues, requisites or user stories in Version One. The dialog window appearance is shown on the following image:

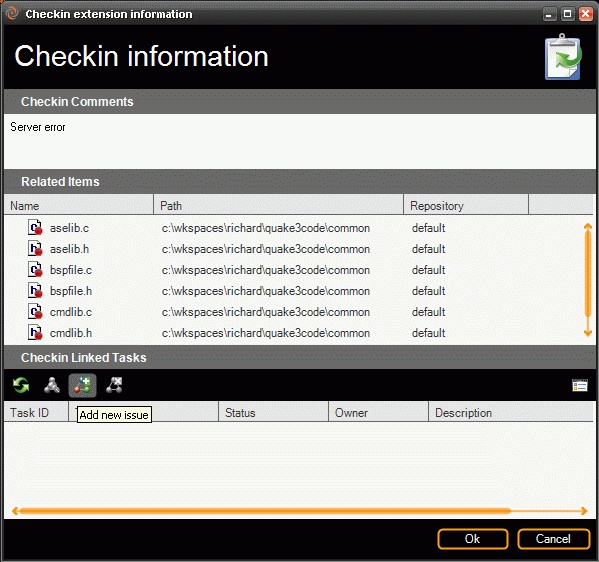


Figure 63: Checkin dialog in task on changeset mode.

By selecting the option “Add new issue” on this check in information dialogue, a new window will be displayed on which to include the full ID of the Version One task and by pressing on the “Details” button the information of the issue will be shown as the following image displays:

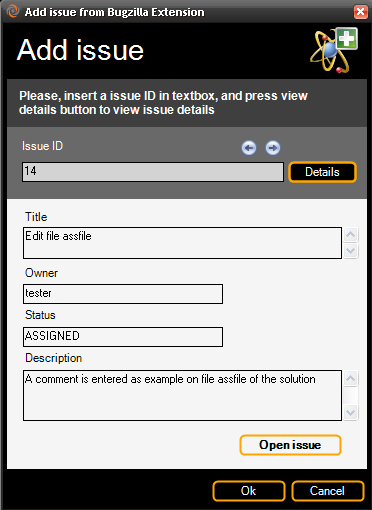


Figure 64: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

Once the changeset(s) have been linked to Version One tasks, the extension information will be shown on the changeset view. From the Version One extension information users can also add new tasks or delete previous ones, and by clicking on each of the issues associated to a certain changeset, a browser with Version One will be opened. Whenever a user modifies an issue in Version One, by refreshing the Version One Extension information, it will be shown in Plastic SCM.

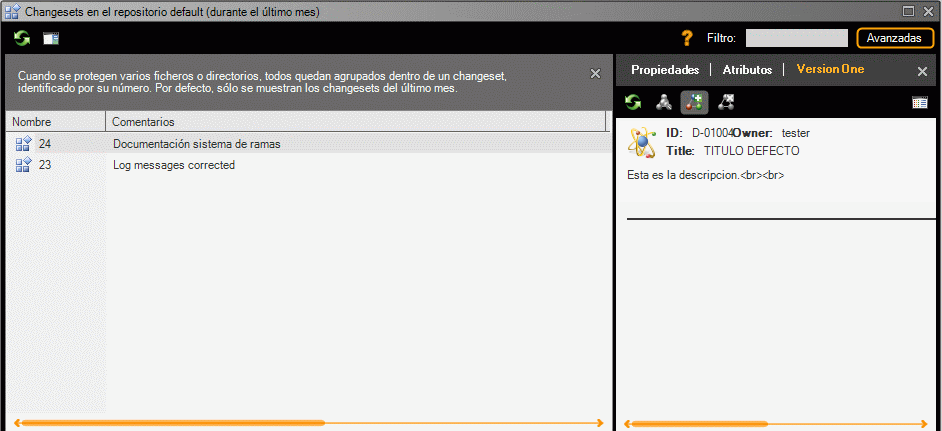


Figure 65: Changeset view with detailed information (task on changeset mode).

# FogBugZ integration

## Introduction

This section describes the steps to follow in order to use FogBugz integration with Plastic SCM. It also describes its advantages.

This extension is compatible with VersionOne version 7.2 and following.

## Configuration

### Server

It is not necessary to make any changes on the server side for Plastic SCM FogBugZ extension.

### Client

To configure the FogBugZ extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select the “FogBugZ” option.

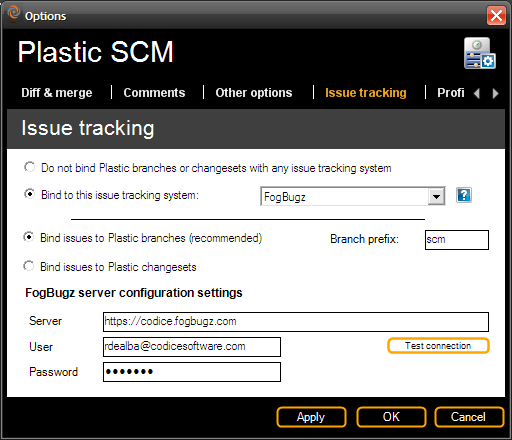


Figure 66: Issue tracking configuration in Plastic SCM: FogBugZ

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work with this mode, a branch prefix must be specified. This branch prefix will be used to bind the plastic branches to the FogBugZ tasks. As the image above shows, the branches that start with prefix ‘scm’ will be linked to FogBugZ issues.

To work on “Task on Changeset mode” select the “Bind issues to Plastic changesets” radio button.

The “Server” parameter specifies the URL where the FogBugZ server is installed and running.

The “User” and “Password” fields must be filled with a valid FogBugZ user.

## Characteristics

### “Task on branch” working mode

The working mode by default is “task on branch”, used for the “branch per task” pattern, on which for each task a new branch will be created.

As usual, the first step would be creating tasks in FogBugZ; this would be usually done by the Project Manager, by clicking on the “New Case” option. A new dialog window will be opened to type the data of the new task.

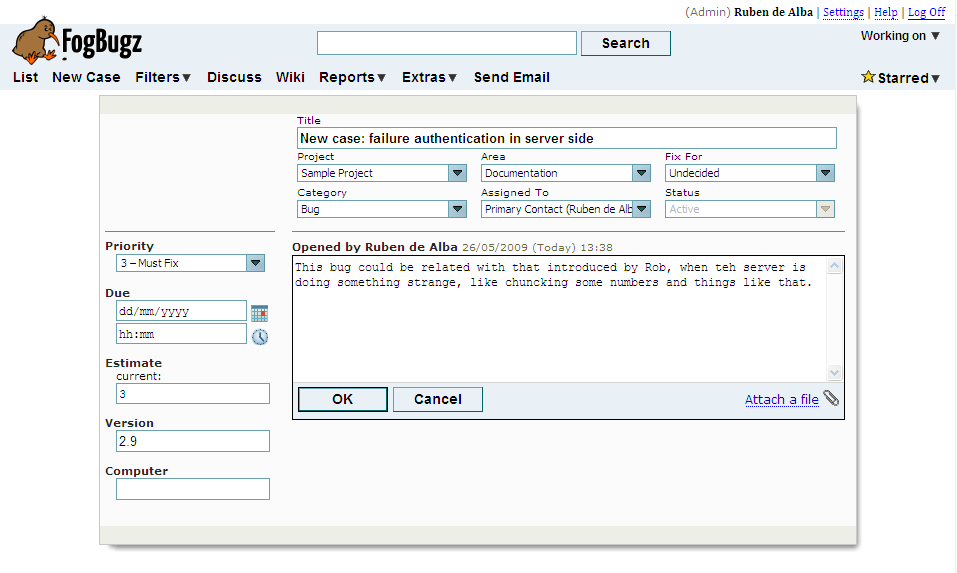


Figure 67: Creating a new issue in FogBugZ

When the issue is created, FogBugZ assigns it a number. The next step is: when a task has been assigned to a developer, the developer has to create a new branch (as simple as right clicking on the parent branch and select the “create child branch” option), as the image below shows. In order to link the branch to the task in FogBugZ, the branch name must match the configured branch prefix and the same task number.

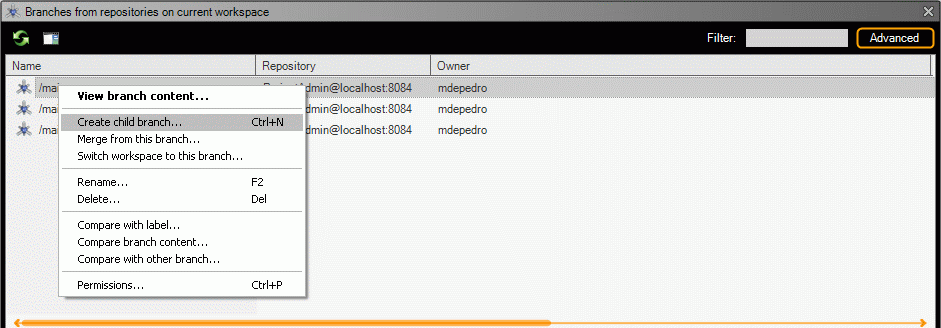


Figure 68: Creating a new branch in Plastic SCM

Now the user can check the information of the issues just by going to the extended information on the branch view and selecting the branch: a new pane will be displayed on the right side of the view from which the FogBugZ extension information is shown: task number, owner, etc.

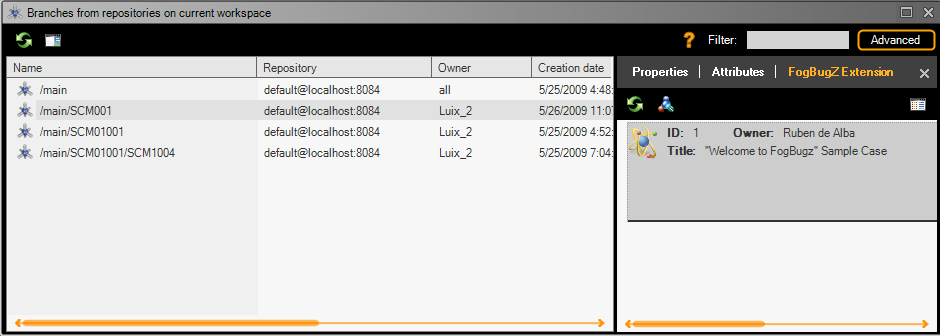


Figure 69: Branch view showing extended information (task on branch mode)

By clicking on it, a new web browser window is opened, which will display the full information of the FogBugZ task. When the user changes the status or any other field of the issue, the new information will be shown by clicking on “Refresh” button of the FogBugZ Extension.

### “Task on changeset” working mode

To configure Plastic SCM to work on “task on changeset” working mode, the user has to configure the FogBugZ extension through PlasticGUI ‘Preferences’ window and select the “Bind issues to plastic Changesets” option, as explained before on the “” section.

In order to start working we must have created FogBugZ tasks so that the developer assigned to those tasks can start working on them. Once the changes are submitted and the developer checks them in, new dialog window will be displayed; on this window users can include their comments on the check in operation as well as the issues associated from their task tracking control, FogBugZ. By using this working mode, one or more than one changeset in Plastic can be associated to one or more than one tasks or defects in FogBugZ. The dialog window appearance is shown on the following image:

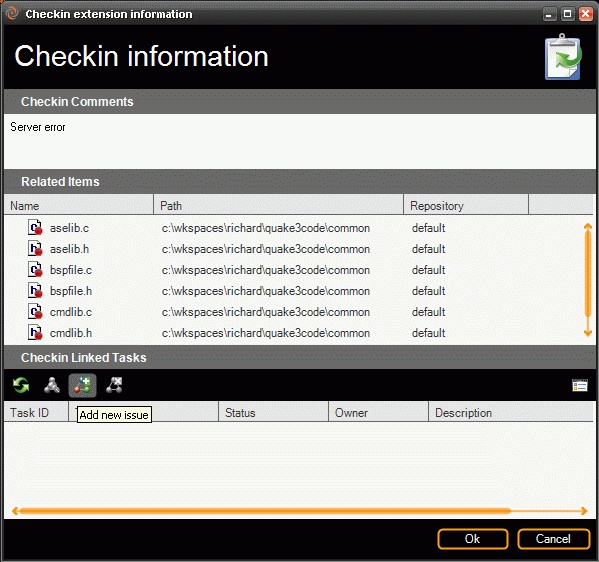


Figure 70: Checkin dialog in task on changeset mode.

By selecting the option “Add new issue” on this check in information dialog, a new window will be displayed on which to include the ID of the FogBugZ issue and by pressing on the “Details” button the information of the issue will be shown as the following image displays:

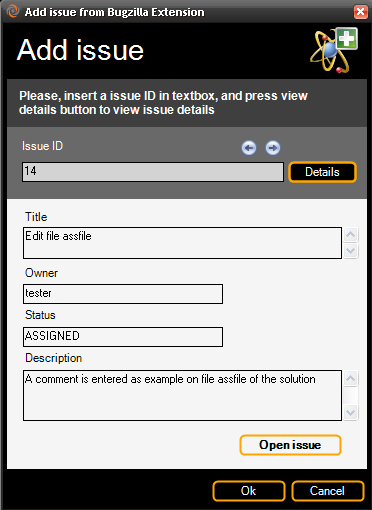


Figure 71: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

Once the changesets have been linked to the issue or task in FogBugZ, the extension information will be shown on the changeset view. From the FogBugZ extension information users can also add new tasks or delete previous ones, and by clicking on each of the issues associated to a certain changeset, a browser with FogBugZ will be opened. Whenever a user modifies an issue in FogBugZ, by refreshing the FogBugZ Extension information, it will be shown in Plastic.

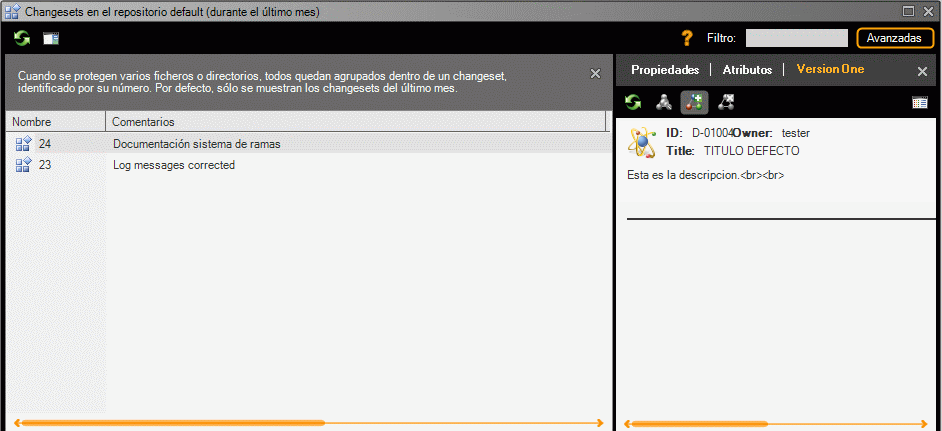


Figure 72: Changeset view showing detailed information (task on changeset mode)

### Checked in operations record

Apart from showing information on the issues associated to the branches integrated with Plastic SCM, this extension logs in FogBugZ every check in done with associated issues. So, after performing a check in operation, linked FogBugZ tasks will have the following appearance:

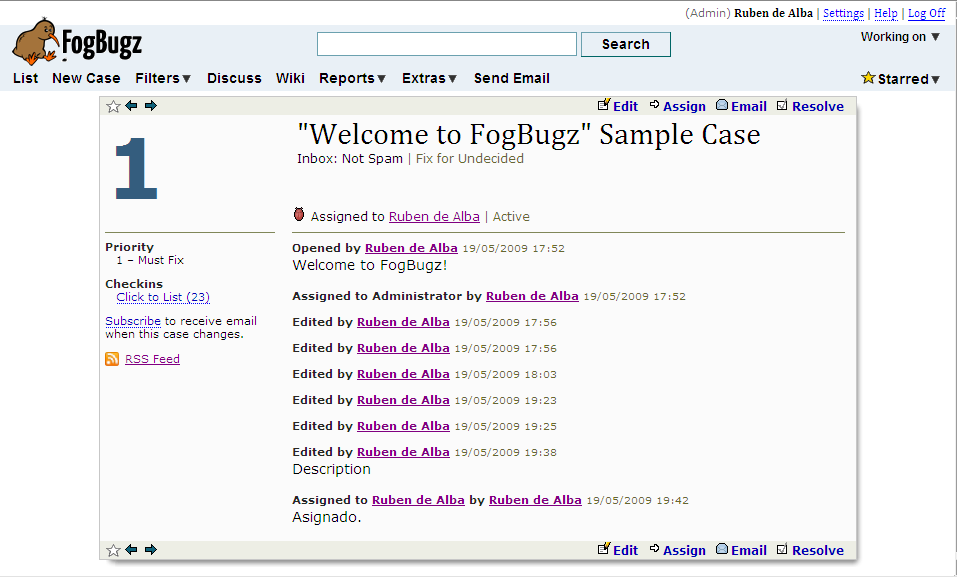


Figure 73: Detail of a FogBugZ issue.

Users can see how each check in operation has been registered on the FogBugZ task by clicking on the “Checkins – Click to list” link:

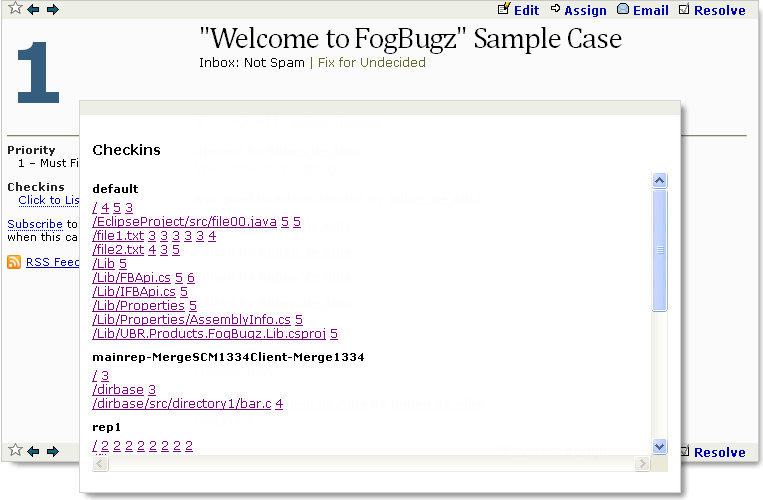


Figure 74: Checkin information generated by Plastic SCM.

# Rally Integration

## Introduction

This section describes the steps to follow in order to use the Plastic SCM Rally Software Development integration.

## Configuration

### Server

To setup the Rally extension functionality, a valid Rally account is needed. There is neither to be done on the server side nor local setup because Rally is a Web based service, all the actions against Rally are done via Internet.

### Client

To configure the Rally extension on the Plastic SCM client, click on “Preferences” tab of the main window of the Plastic SCM GUI. A new dialog window will be opened. Click on the “Issue tracking” tab and select the radio button “Bind to this issue tracking system:” and select the “Rally” option.

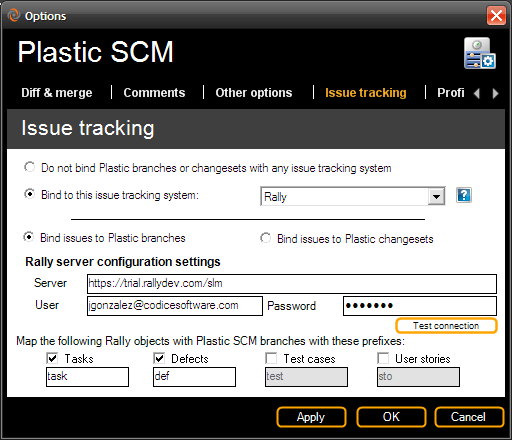


Figure 75: Issue tracking configuration Rally.

By default, the “Bind issues to Plastic branches” option is activated. If Plastic SCM is configured to work with this mode, the user will be able to configure several branch prefixes to bind them to several Rally objects. For instance, if the user enables the mapping between Rally Defects and Rally Tasks, branches that start with prefix ‘def’ will be linked to Rally Defects, and branches that start with prefix ‘task’ will be linked to Rally Tasks. For instance: Branch name ‘def021’ will be linked to Rally Defect ‘DE21’and branch name ‘task021’ will be linked to Rally Task ‘TA21’. The rest of the objects have the same behaviour. Available Rally objects to be linked are Task, Defects, Test cases and User Stories.

To work on “Task on Changeset” mode select the “Bind issues to Plastic changesets” radio button. In this case, all the Rally objects will be enabled to map them to a Plastic changeset.

The “Server” parameter specifies the URL where the Rally server is running.

The “User” and “Password” fields must be filled with a valid Rally user for the configured Rally Server.

## Characteristics

### “Task on branch” working mode

In order to use this working mode, the first step would be setting it up on Plastic SCM Rally extension, following the indications from the previous section, or leaving the configuration by default which applies to this mode.

To start working with Plastic SCM and its Rally extension the objects on Rally must be created. Users can create new tasks, defects, test cases and user stories through the “My Home” tab:

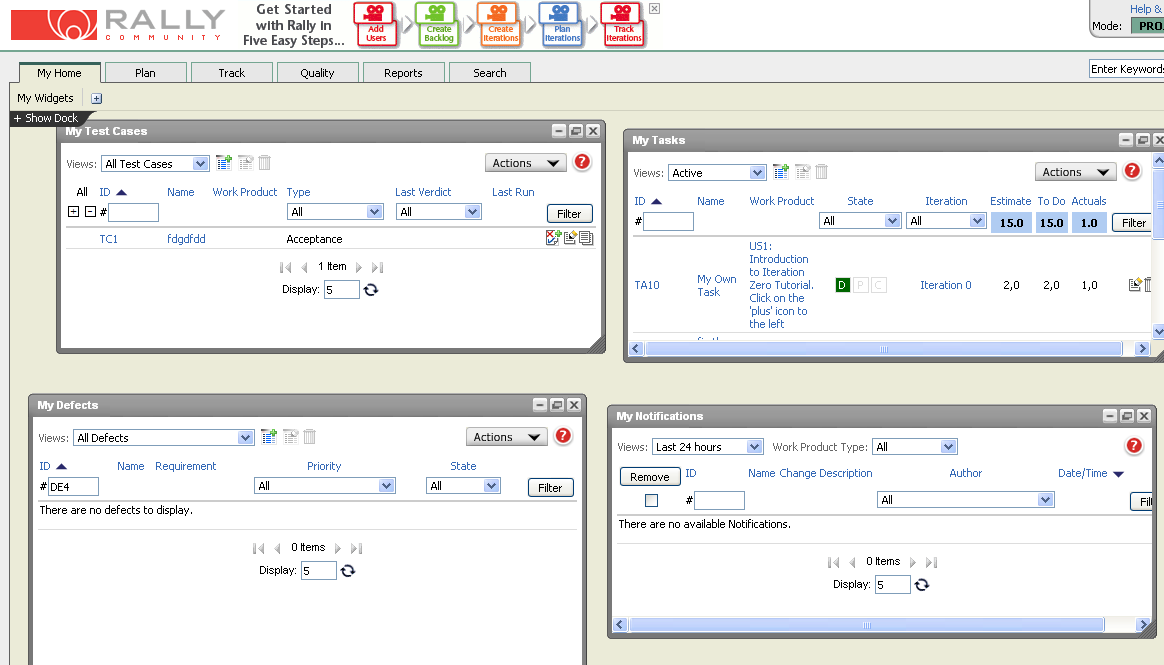


Figure 76: Rally user interface general view.

Defects can be created through its own tab (“Defects”) by clicking on the “Add new defect” action. The same for the rest of the Rally objects, such as “Tasks”, “Test Cases” and “User Stories”.

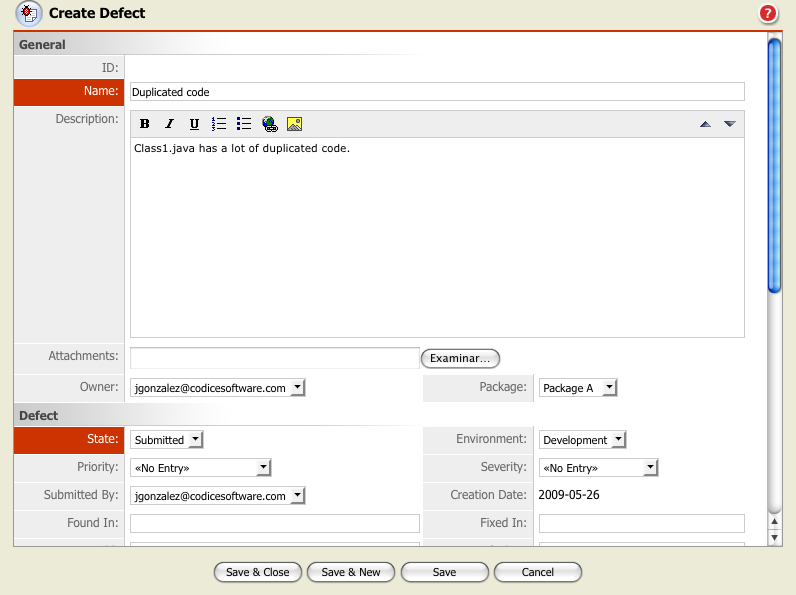


Figure 77: Creating a new defect in Rally.

On the “New Defect” window, information such as name of the new task, priority, status, etc must be included. Once the new defect is saved, Rally will give it an identification string, which will be later used by the assigned developer when creating the Plastic branch to work on that task. The prefix ID will depend on the object type. For instance if user creates a new task, the ID could be ‘TA14’ or ‘DE14’ in case of a defect.

By clicking on the left hand side of the branch view (“show extended information”), the information associated to the branches will be displayed on the GUI´s right hand side as the following screenshot shows. Selecting a branch the detailed information of the Rally object will be shown: name, title, comment and owner.

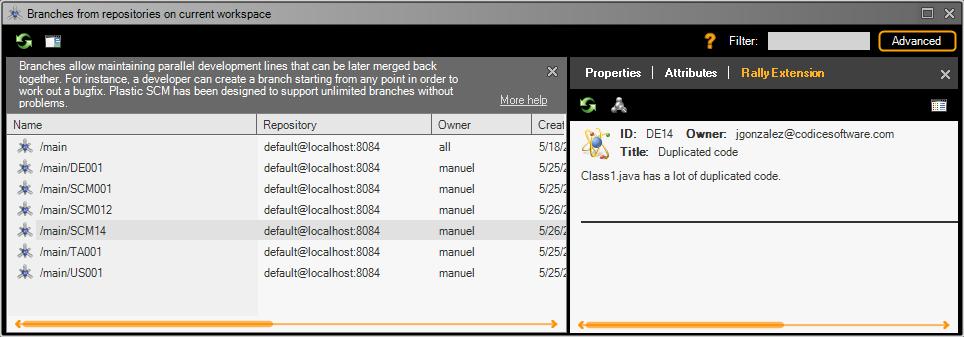


Figure 78: Branch view showing detailed information, in task on branch working mode.

By clicking on it, a web browser window with the associated branch task is opened. Whenever the user finishes the task and changes its status, in order to view the new status, the user will only have to refresh it on the Plastic branch extended information.

### “Task on changeset” working mode

To configure Plastic SCM to work on “task on changeset” working mode, the user has to configure the Rally extension through PlasticGUI ‘Preferences’ window and select the “Bind issues to plastic Changesets” option, as explained before on the “” section.

Once the objects are created in Rally, as showed on the previous working mode section, the developer will normally work on his changes in Plastic and when checking them in, a check in information dialog will be displayed; on this dialog users can include their comments as well as the tasks or issues associated from their task control tool, Rally in this case. Take into account that by using this working mode, one or more than one issue can be associated to one or more changesets:

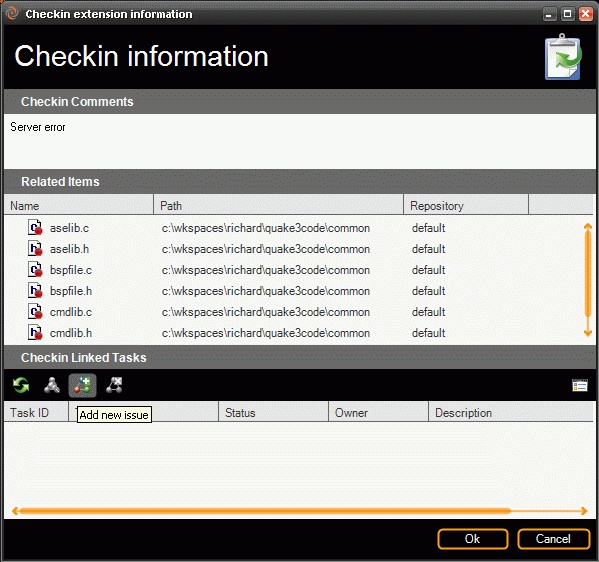


Figure 79: Checkin dialog in task on changeset mode.

On this dialog, by selecting the “Add new issue” option showed on the previous figure, a new dialog window will be displayed. On this window, the user has to type the full ID of the Rally object to be linked to that changeset:

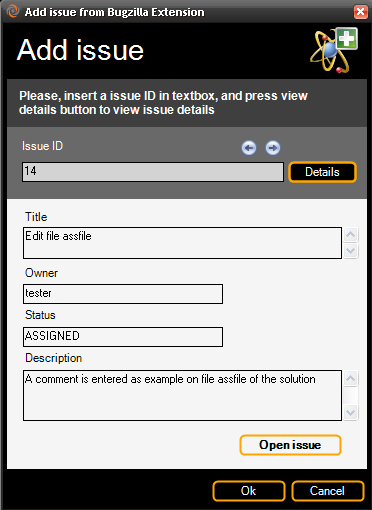


Figure 80: Add issue dialog, to associate a task with a changeset when performing a checkin operation.

Once the changeset has been linked to the Rally object, its information will be shown from the changeset view, and users can also add new links to Rally objects or delete previous ones; and by clicking on each issue a web browser window with the associated object in Rally will be opened. Whenever a user modifies a defect, task, test case or user story from the issue tracking tool, their changes will be loaded by clicking the “refresh” button of Rally Extension.

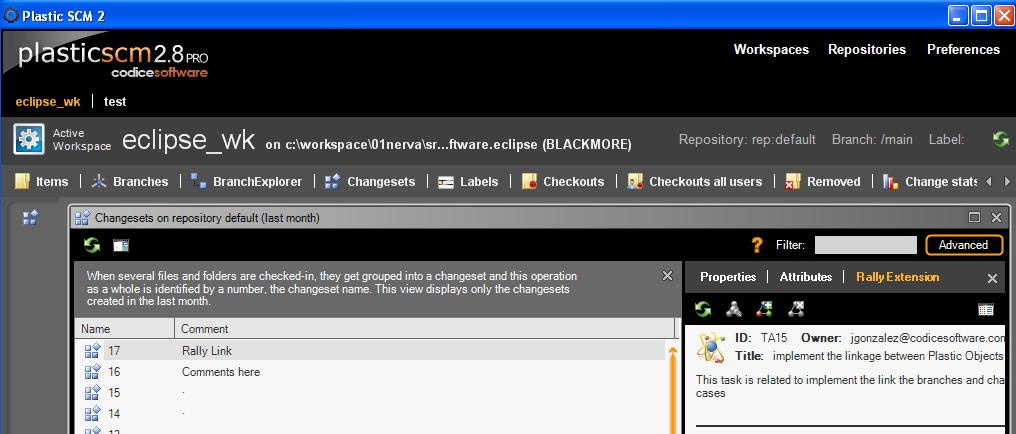


Figure 81: Changeset view showing extended information, in task on changeset working mode.

# Global extension configuration

It’s possible to set a global configuration in the server so that all clients have the same settings by default. They can override the global settings by changing the values in their configuration. This is useful to set default global parameters like connection strings and let the users specify their particular credentials.

The steps to setup the global configuration are summarized in this list:

* Create a configuration file using the Plastic GUI.
* Create the global configuration files and folder in the Plastic server.
* Move the configuration file created in the client to the server, making it global.
* Customize the global configuration file so that it contains only the needed configuration elements.
* Restart the Plastic SCM server.
* On the clients, choose the task management system to use. Global settings will appear by default.

The following sections explain every step.

## Creating a configuration file

Use the Plastic SCM GUI client to set up the extension settings. This will produce a configuration file in the user’s local settings folder:

**Windows XP and previous versions:**

C:\Documents and Settings\<user>\Local Configuration\Program Data\plastic\

**Windows Vista / 7**

C:\Users\<user>\AppData\local\plastic\

**Linux**

/home/<user>/.plastic/

The extension configuration file will have a name composed of your Task Management system (i.e. jira) and the suffix extension.conf, for instance:

**jiraextension.conf**

That file contains all the configuration parameters that Plastic uses to work with the task control system. For example, in the case of Jira it will be something like this:

<?xml version="1.0"?>

<JiraExtensionConfiguration xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<BranchPrefix>scm</BranchPrefix>

<WorkingMode>TaskOnBranch</WorkingMode>

<AttributePrefix />

<SoapUrl>/rpc/soap/jirasoapservice-v2</SoapUrl>

<TaskUrl>/browse/</TaskUrl>

<HttpUrl>http://192.168.1.232:8080</HttpUrl>

<User>admin</User> <Password>|SoC|9UPFGBct4onv5NZJ+zVp+B1pW8sUESx8rvJYVP8Pj9nOSU9MM1Ah6w==</Password>

<ProjectKey>PRJ</ProjectKey>

<CustomFieldId>10020</CustomFieldId>

</JiraExtensionConfiguration>

## Create the global configuration folders on the server

In the server installation folder, the administrator needs to create several files and folders if they not already there:

Plastic SCM\server\

**config\**

**config\extensions**

Inside the “config” folder, a file named configurationfiles.conf contains the files that a client can download from this server. It contains an “extensions” section and the list of files that are available. This file is provided for security reasons, ensuring that only the files listed there can be downloaded from the server. This is a sample file:

[extensions]  
server\config\extensions\jiraextension.conf

In the sample, a Jira configuration file is made available for clients. Notice that the directory separator char is the backslash (“\”), even if the clients or server are running on a Unix platform.

In order to load the new configuration, restart the Plastic SCM service. This is only needed when modifications are made on the configurationfiles.conf file.

## Move the configuration file from the client to the server

Move the extension configuration file created in step 1 to the “config\extensions\” folder just created on the server.

## Customize the global configuration

Now the extension.conf file can be customized with the common values that every client will use. This way, when a user opens the ‘Issue Tracking’ tab in the ‘Preferences’ dialog of Plastic SCM client, those common values will be loaded automatically, and the rest of values which the global configuration does not state must be filled by the user.

Notice that the client where the configuration was initially set up contains all the values in its extension.conf file, located in the local settings folder, so if you want to test whether the values of the global configuration are correctly loaded you should delete previously this file.

## Restart the Plastic SCM server

It is necessary to restart the Plastic SCM server so that it can use the new configuration and load it in the clients when a user opens the ‘Issue tracking’ tab in the ‘Preferences’ dialog of the Plastic client.

## Setup clients

On the clients, choose the task management system to use. Global settings will appear by default. It is important that every user sets the task control system using the Plastic client, so that a extension.conf file can be saved in his local settings folder. This file will contain the information provided by the user; the information obtained by the server and not modified by the client will be saved empty. It is important to annotate that the password is saved encrypted in the extension.conf file.