

1. Crucible Documentation Home	3
1.1 About Crucible	4
1.1.1 Background Reading	5
1.2 Crucible User's Guide	5
1.2.1 Using the Crucible Screens	6
1.2.1.1 Browsing Your Reviews	6
1.2.1.2 Browsing Source Files	8
1.2.1.3 Browsing Projects	9
1.2.1.4 Viewing People's Statistics	10
1.2.1.5 Searching in Crucible	14
1.2.1.6 Using RSS Feeds in Crucible	15
1.2.1.7 Using Keyboard Shortcuts in Crucible	15
1.2.2 Getting Started with Crucible	16
1.2.3 Using the Dashboard	23
1.2.3.1 Using the Project Dashboard	25
1.2.4 Conducting a Review	26
1.2.4.1 Creating a Review	26
1.2.4.1.1 Creating a Review from FishEye	26
1.2.4.1.2 Creating a Review from JIRA	28
1.2.4.1.3 Creating a Review within Crucible	29
1.2.4.1.4 Creating a Review from a URL	30
1.2.4.1.5 Creating a Patch Review	31
1.2.4.1.6 Selecting the Files for the Review	36
1.2.4.2 Adding Reviewers	42
1.2.4.2.1 Removing Reviewers From An Active Review	44
1.2.4.3 Issuing a Review	45
1.2.4.4 Performing the Review	45
1.2.4.4.1 Adding Comments	47
1.2.4.4.2 Flagging Defects	49
1.2.4.4.3 Completing your Review	49
1.2.4.4.4 Sending all of a Review's Comments via Email	51
1.2.4.5 Summarising and Closing the Review	53
1.2.4.6 Deleting an Abandoned Review	56
1.2.4.7 Moving a Review to Another Project	56
1.2.5 Changing your User Profile	57
1.2.6 Defining your Workflow	59
1.2.7 Roles and Status Classifications	63
1.2.8 Using Favourites	63
1.3 Crucible Administrator's Guide	68
1.3.1 Backing Up and Restoring Crucible Data	68
1.3.2 Creating a Permission Scheme	74
1.3.2.1 Associating a Permission Scheme with a Project	77
1.3.3 Creating a Project	79
1.3.3.1 Setting Crucible to Store all Revisions	80
1.3.4 Crucible and FishEye	81
1.3.5 Customising Email Notifications	81
1.3.5.1 Freemarker Data Model for Email Templates	83
1.3.6 Customising the Defect Classifications	83
1.3.7 Customising the Welcome Message	85
1.3.8 Deleting a Project	86
1.3.9 Editing a Project	87
1.3.9.1 Setting the Default Review Duration for a Project	88
1.3.10 JIRA Integration in Crucible	89
1.3.11 Migrating to an External Database	93
1.3.12 Trusted Applications	96
1.4 Crucible Development Hub	98
1.4.1 Documentation for Crucible Development	100
1.4.1.1 Crucible's URL Structure	100
1.4.1.2 Crucible REST API	102
1.4.1.2.1 Crucible REST API Usage Example	102
1.4.1.2.2 Conditional Get	107
1.4.1.2.3 Data Types	107
1.4.1.2.4 Project Service	114
1.4.1.2.5 Repository Service	115
1.4.1.2.6 Review Service	116
1.4.2 Crucible Plugin Types	146
1.4.2.1 Crucible Web Items	146
1.4.3 Live Code Examples for Crucible Development	152
1.4.3.1 Bundled Plugins from Crucible	152
1.4.3.1.1 Confluence SCM Plugin	152
1.4.3.1.2 File System SCM Plugin	153
1.4.3.1.3 Perforce SCM Plugin	153
1.4.3.1.4 Subversion SCM Plugin	154
1.4.3.2 SCM Plugin Examples	154
1.4.3.2.1 Crucible ClearCase plugin	155

1.4.3.2.2 Crucible Git Plugin	156
1.4.3.2.3 Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)	157
1.4.3.3 Servlet Examples	160
1.4.3.3.1 Basic Servlet Example	160
1.4.3.3.2 Crucible Reporting plugin	161
1.4.4 Developing Crucible Plugins	163
1.4.4.1 Crucible Event Listener Plugins	164
1.4.4.2 Crucible SCM Plugins	165
1.4.4.3 The Crucible API	177
1.5 Crucible FAQ	178
1.5.1 Can Crucible be run as a Windows Service?	179
1.5.2 Do I need a FishEye licence to run Crucible?	180
1.5.3 How to Automate Daily Crucible Backups	180
1.5.4 Troubleshooting	180
1.6 Crucible Installation & Upgrade Guide	180
1.6.1 Crucible Installation Guide	180
1.6.1.1 System Requirements	181
1.6.1.2 Installing Crucible	181
1.6.1.3 Configuring Crucible	182
1.6.1.4 Configuring Repositories	186
1.6.1.4.1 Enabling Reviews from the Server File System in Crucible	186
1.6.1.4.2 Setting Up a Git Repository in Stand-Alone Crucible	187
1.6.1.4.3 Setting Up a Perforce Repository in Stand-Alone Crucible	188
1.6.1.4.4 Setting Up a Repository via FishEye	188
1.6.1.4.5 Setting Up a Subversion Repository in Stand-Alone Crucible	189
1.6.1.4.6 Setting Up Reviewing of Confluence Pages in Crucible	190
1.6.1.5 Best Practices for Crucible Configuration	191
1.6.2 Crucible Upgrade Guide	191
1.6.2.1 Upgrading to a New Version of Crucible	192
1.6.2.2 Upgrading from FishEye to Crucible	193
1.7 Crucible Release Notes	194
1.7.1 Crucible Release Summary	194
1.7.2 Crucible 2.0 Release Notes	195
1.7.2.1 Crucible 2.0 Changelog	200
1.7.3 Crucible 2.0 Beta Release Notes	210
1.7.3.1 Upgrading to the Crucible 2.0 Beta	215
1.7.3.2 Crucible 2.0 Beta Reviewer's Guide	216
1.7.3.3 JIRA Integration in Crucible 2.0 Beta	217
1.7.4 Crucible 1.6 Release Notes	222
1.7.4.1 Crucible 1.6 Changelog	229
1.7.4.1.1 Crucible 1.6.3 Upgrade Guide	234
1.7.5 Crucible 1.5 Release Notes	235
1.7.5.1 Crucible 1.5 Changelog	238
1.7.6 Crucible 1.2 Release Notes	240
1.7.6.1 Crucible 1.2 Changelog	244
1.7.6.2 Crucible 1.2 Upgrade Guide	246
1.7.7 Crucible 1.1 Release Notes	246
1.7.7.1 Crucible 1.1 Changelog	246
1.7.7.2 Crucible 1.1 Upgrade Guide	247
1.8 Glossary	247
1.8.1 approve	248
1.8.2 author	248
1.8.3 code review	248
1.8.4 comment	248
1.8.5 creator	248
1.8.6 defect	248
1.8.7 moderator	249
1.8.8 participant	249
1.8.9 permission	249
1.8.10 permission scheme	250
1.8.11 project	250
1.8.12 review duration	250
1.8.13 reviewer	250
1.8.14 role	250
1.8.15 state	250
1.8.16 statement of objective	251
1.8.17 user	251

Crucible Documentation Home

Getting Started with Crucible 2.0

[Release Notes \(What's new\)](#)

[Download Crucible](#) | [Feature Tour](#) | [About Crucible](#)

[Installation Guide](#) | [Upgrade Guide](#)

Using/Administering Crucible 2.0

[User's Guide](#)

[Administrator's Guide](#)

[Quick Start: Conducting a Review](#)

Crucible IDE Integration

Use the [Atlassian Connector for Eclipse](#) or the [Atlassian Connector for IntelliJ IDEA](#) to work with your Crucible code reviews right there in your development environment. Do you use [JIRA](#), [Bamboo](#) or [FishEye](#) too? With the connector you can manage your issues and builds within your IDE, or move quickly between the IDE and a FishEye view of your source repository. **Hint:** The Atlassian IDE Connectors are free.

Resources

[Support](#) | [Forums](#) | [FAQ](#)

[Development Hub](#)

[FishEye Documentation](#)

[Crucible Evaluator Resources](#)

[Glossary](#)

Previous Versions

[Crucible 1.6 documentation](#)

[Crucible 1.5 documentation](#)

[Crucible 1.2 documentation](#)

[Crucible 1.1 documentation](#)

[Crucible 1.0 documentation](#)

Offline Documentation

You can download the Crucible documentation in PDF, HTML or XML format for use offline.

Recently Updated



[Performing the Review \(Crucible 2.0\)](#)

by Rosie Jameson [Atlassian Technical Writer] (an hour ago)



[file_outdated.png \(Crucible 2.0\)](#)

by Rosie Jameson [Atlassian Technical Writer] (an hour ago)



[file_outdated_menu.png \(Crucible 2.0\)](#)

by Rosie Jameson [Atlassian Technical Writer] (2 hours ago)

 Migrating to an External Database (Crucible 2.0)	by Partha Kamal (08 Jul)
 __newreleaseCrucible (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)
 Crucible 2.0 Changelog (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)
 Crucible SCM Plugins (Crucible 2.0)	by Sarah Maddox [Atlassian Technical Writer] (06 Jul)
 Using Keyboard Shortcuts in Crucible (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)
 Changing your User Profile (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)
 Setting the Default Review Duration for a Project (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)
 Crucible ClearCase plugin (Crucible 2.0)	by Ross Rowe [Atlassian] (06 Jul)
 crucible-clearcase-0.1.0.jar (Crucible 2.0)	by Ross Rowe [Atlassian] (06 Jul)
 Crucible Web Items (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)
 Using RSS Feeds in Crucible (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)
 Moving a Review to Another Project (Crucible 2.0)	by Rosie Jameson [Atlassian Technical Writer] (06 Jul)

About Crucible

Crucible is a powerful addition to FishEye, making it easy to review code changes, make comments, and record outcomes in an efficient, distributed, and process-neutral way.

Introduction

Crucible is a tool that facilitates code review. It can be as valuable to organisations that already have a formal inspection process as it is to teams that don't review at all.

Regular peer review is a proven process with demonstrable return on investment (ROI). The benefits vary from team to team but commonly include:

- Identifying bugs and defects early.
- Sharing expertise and encouraging knowledge transfer.
- Improving system-wide knowledge.
- Encouraging adherence to internal standards and style conventions.
- Identifying individual strengths and weaknesses.

One of the less apparent, but nonetheless important, benefits that comes from a transparent code review process is that quality improves simply from the knowledge that code may be critically reviewed. Developers take more care with style, readability, comments, and commit-messages because their peers are going to see them.

Despite these and many other clear benefits, code review is often seen as 'impractical on time sensitive projects', 'only valuable in large teams working on mission critical applications', or at worst 'a total waste of time foisted on developers by management'. Formal code review can *feel* like an expensive use of time, because the review process can:

- Be burdened by excessive paperwork and other administration.
- Interrupt your current task and make you less productive.
- Include meetings where participants fail to prepare, so that the meeting becomes a walkthrough rather than a critical review.
- Become an ego battle or point-scoring exercise dominated by a vocal minority.

These issues do not affect the immense potential value of code review. They are simply problems with some review processes.

Crucible's mission is to streamline the *process* aspects so development teams can access the benefits. Crucible achieves this by:

- Making reviews asynchronous.
- Bringing reviewing to your desk (wherever that might be).
- Eliminating most of the administration.
- Limiting the ability for individuals to dominate the dialogue.
- Providing an archival record of reviews.

Crucible increases the quality, quantity, and frequency of code reviews thereby reducing bugs, helping knowledge sharing and fundamentally

improving system quality.

Starting Points

Visit the [Crucible Feature Tour](#) to understand how Crucible can benefit you.

You can run Crucible with a FishEye-compatible source code repository set up, such as CVS, Subversion, or Perforce. For more information, please read the [FishEye documentation](#).

Read the [Installation Guide](#) to get started quickly.

For Crucible troubleshooting, see the [FAQ](#).

Background Reading

The following resources are recommended for background reading on peer code reviews:

- [White paper](#) on effective code review by Karl Wiegers.
- Book, [Peer Reviews in Software: A Practical Guide](#) by Karl Weigers.
- Software Engineering Institute web page: [Software Inspections](#).
- NASA Software Assurance Technology Center web page: [Software Formal Inspections](#).

Crucible User's Guide

This page is an index of the content in the Crucible User's Guide. Click on a link below to see the desired page.

- [Using the Crucible Screens](#)
 - [Browsing Your Reviews](#)
 - [Browsing Source Files](#)
 - [Browsing Projects](#)
 - [Viewing People's Statistics](#)
 - [Searching in Crucible](#)
 - [Using RSS Feeds in Crucible](#)
 - [Using Keyboard Shortcuts in Crucible](#)
- [Getting Started with Crucible](#)
- [Using the Dashboard](#)
 - [Using the Project Dashboard](#)
- [Conducting a Review](#)
 - [Creating a Review](#)
 - [Creating a Review from FishEye](#)
 - [Creating a Review from JIRA](#)
 - [Creating a Review within Crucible](#)
 - [Creating a Review from a URL](#)
 - [Creating a Patch Review](#)
 - [Selecting the Files for the Review](#)
 - [Adding Reviewers](#)
 - [Removing Reviewers From An Active Review](#)
 - [Issuing a Review](#)
 - [Performing the Review](#)
 - [Adding Comments](#)
 - [Flagging Defects](#)
 - [Completing your Review](#)
 - [Sending all of a Review's Comments via Email](#)
 - [Summarising and Closing the Review](#)
 - [Deleting an Abandoned Review](#)
 - [Moving a Review to Another Project](#)
- [Changing your User Profile](#)
- [Defining your Workflow](#)
- [Roles and Status Classifications](#)
- [Using Favourites](#)



Manage your Crucible reviews inside your IDE

Use the [Atlassian Connector for Eclipse](#) or the [Atlassian Connector for IntelliJ IDEA](#) to work with your Crucible code reviews right there in your development environment. Do you use JIRA, Bamboo or FishEye too? With the connector you can manage your issues and builds within your IDE, or move quickly between the IDE and a FishEye view of your source repository. **Hint:** The Atlassian IDE Connectors are free.

Using the Crucible Screens

This page contains an overview of the Crucible interface and the actions that can be carried out in the application.

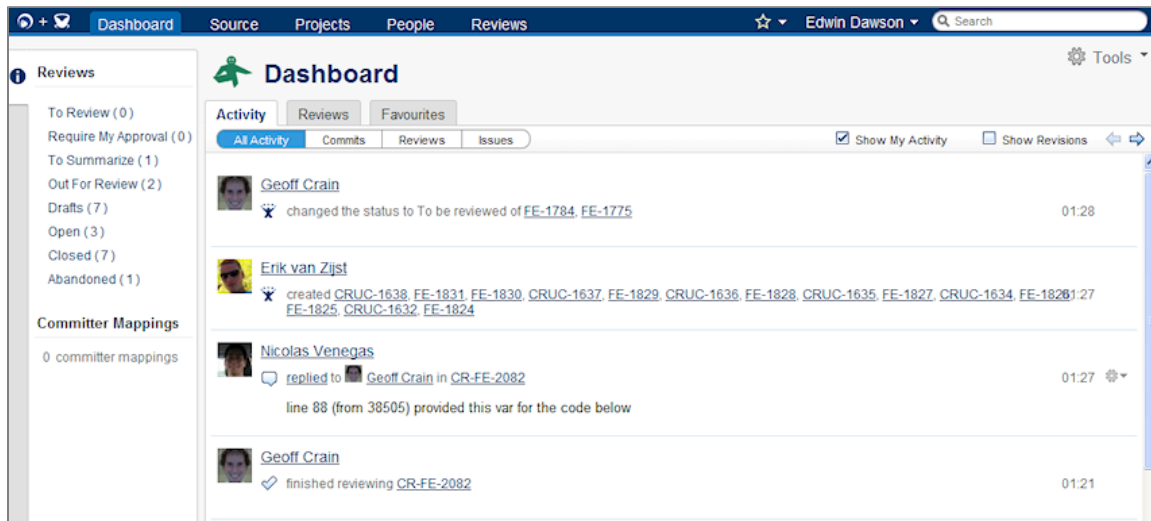
On this page:

- [Tour of the Crucible Interface](#)
- [Left Navigation Sidebar](#)
- [Related Links](#)

Tour of the Crucible Interface

When you first log in to Crucible, the Dashboard Screen opens, as shown in the screenshot below. This view shows recent general activity in Crucible.

Screenshot: *The Dashboard Screen in Crucible*



The table below explain the top-level tabs in the Crucible User Interface. Click on the name of a tab for more information.

Element name	Function	Appears
Dashboard Tab	Displays reviews and system activity related to you.	All screens.
Source Tab	Displays contents of connected source repositories.	Only when FishEye is used with Crucible.
Projects Tab	Displays reviews and content from specific projects.	All screens.
People Tab	Displays metrics on the users of the Crucible instance.	All screens.
Reviews Tab	Displays reviews.	All screens.

Left Navigation Sidebar

The navigation bar at the left of the screen applies specific filters to what is shown in the centre pane. See the page on [Browsing Your Reviews](#) for more information.

 The left navigation sidebar can be hidden or displayed by clicking the blue 'information' icon  at the top left of the sidebar.

Related Links

[Browsing Your Reviews](#)
[Browsing Source Files](#)
[Browsing Projects](#)
[Viewing People's Statistics](#)
[Searching in Crucible](#)
[Using RSS Feeds in Crucible](#)
[Changing your User Profile](#)

Browsing Your Reviews

This page contains information on browsing reviews in Crucible.

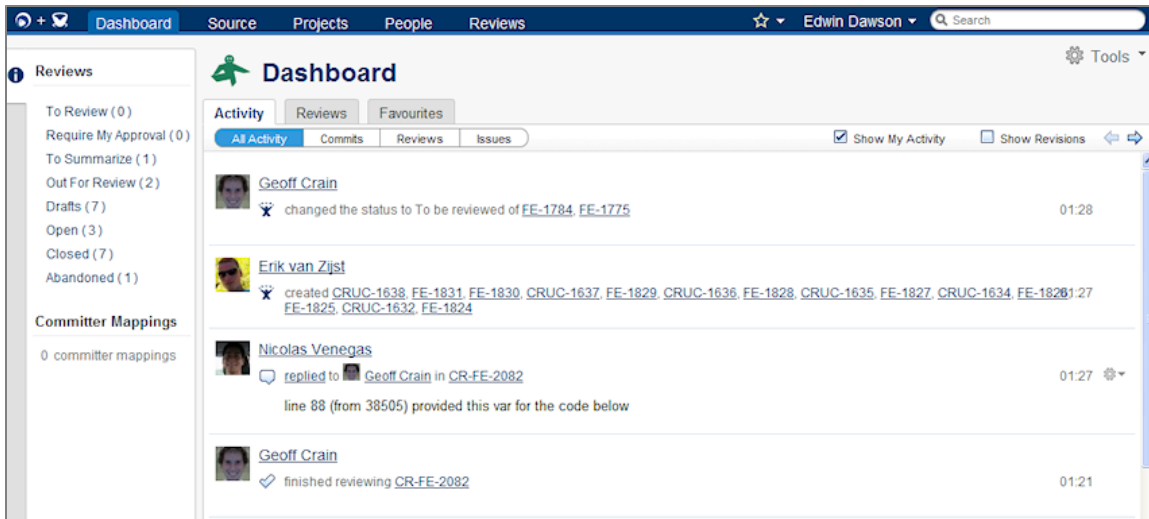
On this page:

- [Dashboard Screen Overview](#)
 - [Filtering Your View](#)

Dashboard Screen Overview

To browse all your reviews (reviews you are participating in), use the **Dashboard** tab at the top of the page. The Dashboard Screen opens, as shown in the screenshot below.

Screenshot: The Dashboard Screen in Crucible



By default, the Activity Stream is shown. This is a mix of all activity that is occurring related to Crucible, such as people making review comments, reviews opening and closing, files being committed to a linked repository, or updates to linked JIRA issues.

Filtering Your View

To filter your view, use the constraint options in the sub-nav and the side panel.

Activity Stream filters

You can also filter the items shown in the Activity Stream. To do this, click one of the options in second layer of tabs, as listed in the table below.

Tab Name	Sub-Nav Options
Activity Tab	Sub-Nav Options: <ul style="list-style-type: none">• All Activity — Shows all activity with no filtering.• Commits — Shows commits.• Reviews — Shows reviews.• Issues — Shows JIRA issues.• Show/Hide My Activity — Show/hide your own activity.• Show/Hide Revisions — Show/hide activity from other users.• Earlier / Later Activity (arrow buttons) — Pages through activity items, stepping backward or forward through the results.
Reviews Tab (Shows the result of the filter clicked in the left side panel)	Sub-Nav Options: RSS — Opens a page with the RSS feed for the current selection.
Favourites Tab (Shows all items you have marked as a 'favourite'.)	Sub-Nav Options: None.

Side Panel

The left navigation panel of the Dashboard shows the number of reviews in different states. Click on any of these states to show the list of reviews in the left-hand panel.

To Review	Reviews where the user still needs to complete their work.
Require My Approval	The user has been assigned the role of moderator for these reviews and needs to approve them.
To Summarize	The user has been assigned the role of moderator for these reviews and needs to summarise and close them.
Out For Review	Reviews created by the user that are currently in progress.
Drafts	These are reviews created by the user that have not yet been moved to the 'Approval' or the 'Require Approval' states .
Open	All open reviews that the user is participating in.
Closed	These are reviews that the user has been involved in and are now closed.
Abandoned	Reviews that are no longer relevant and can be deleted.

Browsing Source Files

When FishEye is installed with Crucible, you have the additional **'Source'** tab available in the navigation tabs at the top of the screen.

To view source files, click the **'Source'** tab. The **'Repositories'** view opens, showing summary information if you have multiple repositories set up. Click on the desired repository to view its contents.

The file explorer mode of the source view will open. Here, you can navigate through the repository by selecting files and folders on the tree in the left navigation bar. When you reach a source file, a summary page is shown, displaying recent revisions to the file.

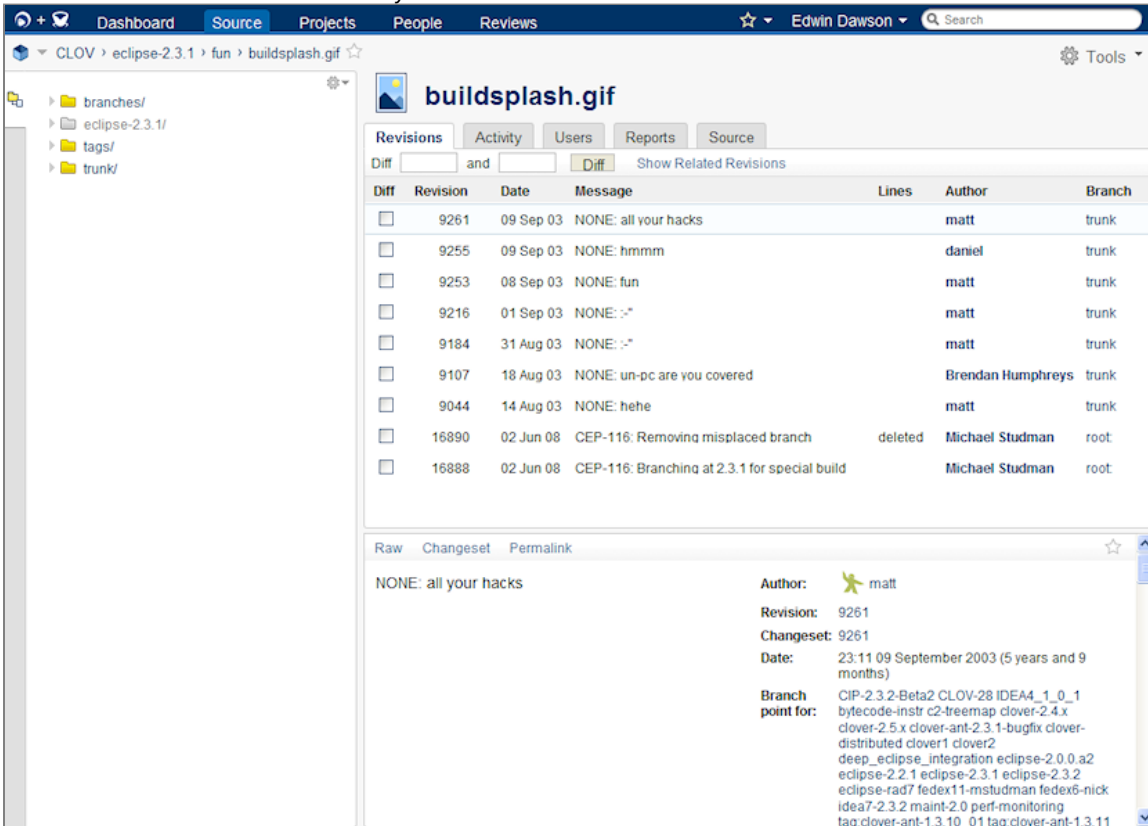
At this file level, there are five sub-tabs you can select. Their functions are outlined in the following table.

Sub-Tab Name	Description
Revisions	When viewing a file, shows the latest revisions of the file.
Files	When viewing a folder, shows the contents of the directory.
Activity	Shows recent activity on the item. There are a number of sub-options here: <ul style="list-style-type: none">• All Activity — The default view, showing commits, reviews and JIRA issues.• Commits — Shows commits in the activity stream.• Reviews — Shows review activity in the activity stream.• Scroll to Changeset — Opens the changeset ID specified in the text field (press Enter to carry out the action).• Filter — Applies constraints to the current activity stream.• Show Revisions — If this is selected, then changeset items are automatically expanded to show modified files.• Earlier Activity (Left Arrow icon) — Loads a page of earlier activity.• Later Activity (Right Arrow icon) — Loads a page of later activity.
Users	Shows the commit history of the different users that have committed changes on the item.
Reports	Shows activity charts for the item. Various chart options can be selected in the left navigation bar.
Source	Shows the contents of the file.
Query	Allows you to run an advanced search.

Screenshot: The Repositories View



Screenshot: The Source View Item Activity



Browsing Projects

To browse the content in a project, click the **Projects** tab at the top of the page. The **'Projects'** view opens.

The Projects tab is only visible in Crucible. Read more about the [definition of a project](#).

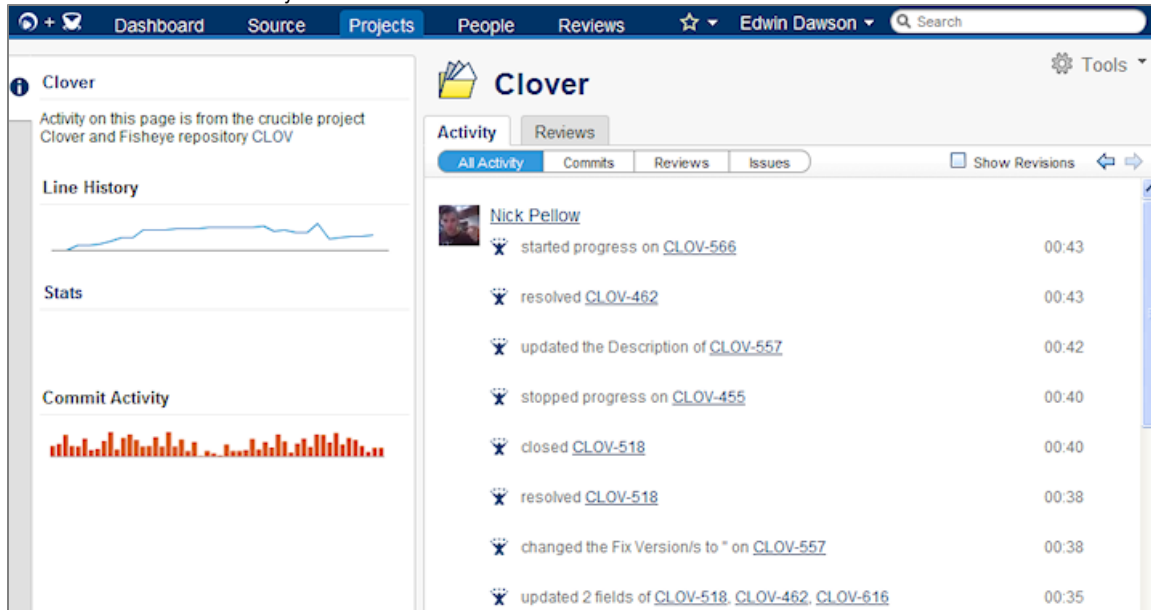
A list of projects will be shown if there is more than one. Click the name of the desired project to open it. The **'Project Activity'** page opens. In the left navigation bar, charts showing overall project statistics are displayed.

There are a number of sub-tabs on this page, listed in the table below.

Sub-Tab Name	Description
--------------	-------------

Activity	<ul style="list-style-type: none"> • All Activity — The default view. • Commits — Shows commits in the project (visible when using FishEye). • Reviews — Shows reviews in the project. • Issues — Shows JIRA issues related to this project. • Show Revisions — Shows or hides revisions in the project (visible when using FishEye). • Earlier Activity (Left Arrow icon) — Loads a page of earlier project activity. • Later Activity (Right Arrow icon) — Loads a page of later project activity.
Reviews	Shows recent reviews in the project.

Screenshot: The Crucible Projects View



Screenshot: The Crucible Projects Index

The screenshot shows the 'Projects' index page in Crucible. It features a table with two columns: 'Project name' and 'Latest Activity'. The projects listed include Default Project, Confluence Hosted, Jira Studio, FishEye, Clover, Clover IDEA plugin, Testing Project, and Clover Eclipse Plugin, each with a corresponding latest activity timestamp.

Project name	Latest Activity
☆ Default Project	📅 on 25 Jun 2009
☆ Confluence Hosted	📅 on 25 Feb 2009
☆ Jira Studio	📅 on 22 Apr 2008
★ FishEye	🔍 on 25 Jun 2009
☆ Clover	🔍 on 25 Jun 2009
☆ Clover IDEA plugin	📅 on 25 Jun 2009
☆ Testing Project	💬 on 23 Jun 2009
☆ Clover Eclipse Plugin	📅 on 25 Jun 2009

Viewing People's Statistics

This page contains instructions on how to use the People tab in Crucible to see charts and activity from people with accounts on the system.

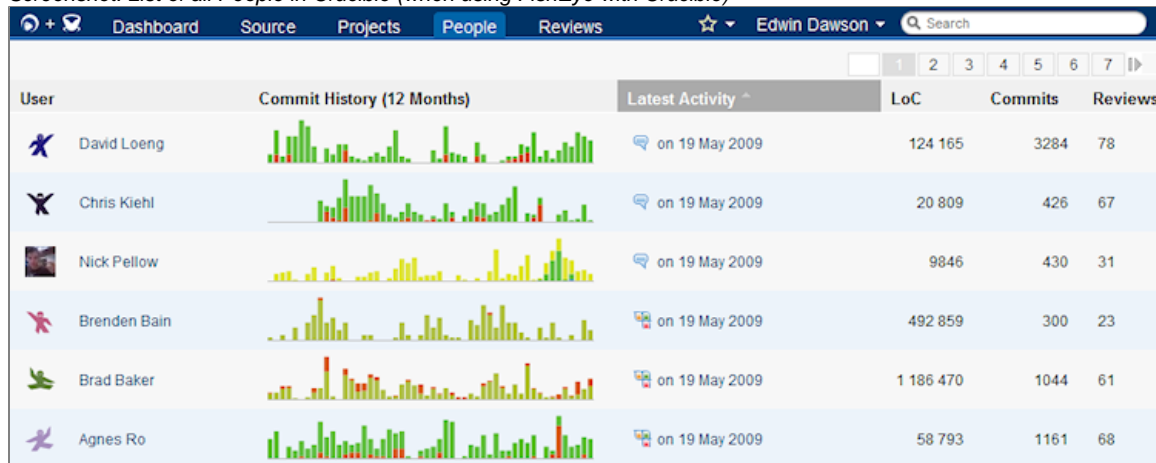
On this page:

- [Opening the List of People](#)
- [Viewing a Person's Activity Screen](#)
- [Viewing Charts on a Person's Activity](#)

Opening the List of People

To view statistics on People in Crucible, (that is, code authors, committers and reviewers) click the **People** tab at the top of the page. The list of all People appears.

Screenshot: List of all People in Crucible (when using FishEye with Crucible)



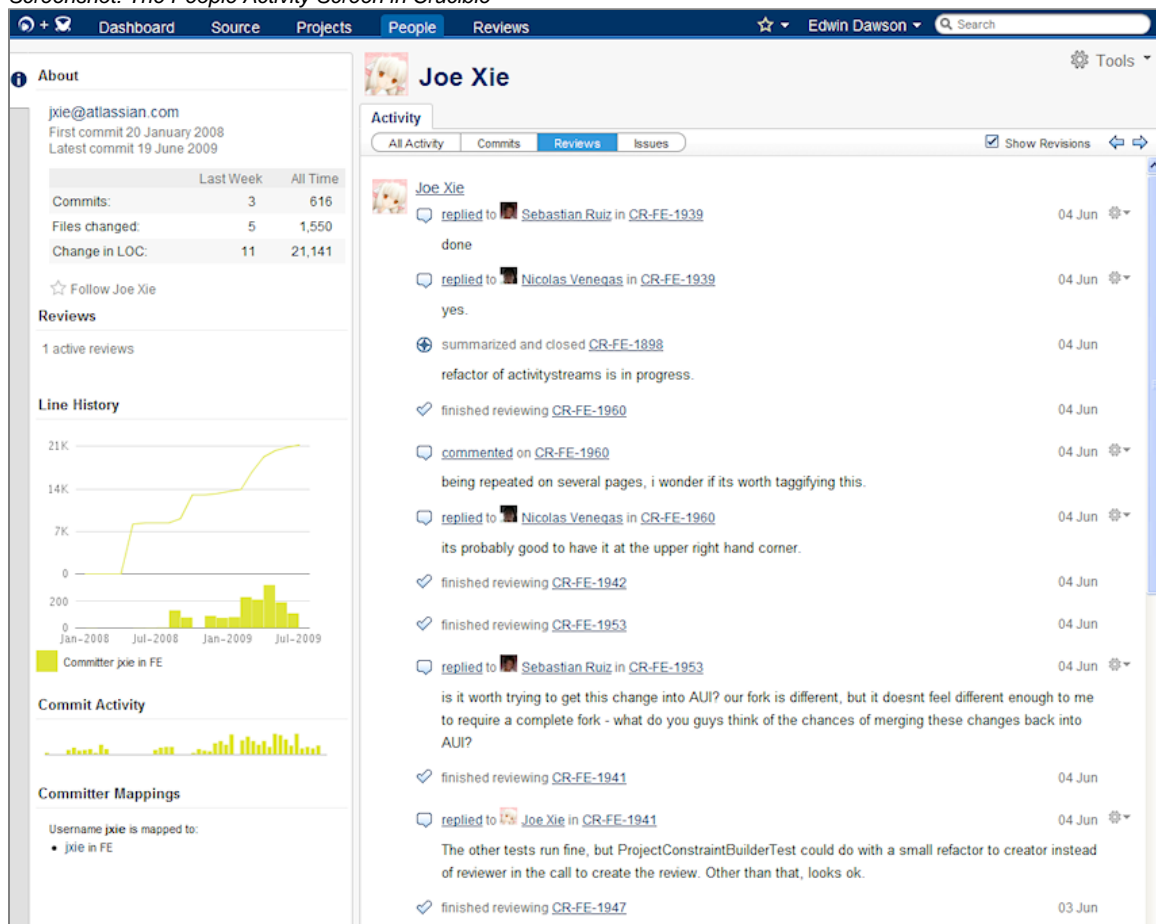
User	Commit History (12 Months)	Latest Activity ^	LoC	Commits	Reviews
David Loeng		on 19 May 2009	124 165	3284	78
Chris Kiehl		on 19 May 2009	20 809	426	67
Nick Pellow		on 19 May 2009	9846	430	31
Brenden Bain		on 19 May 2009	492 859	300	23
Brad Baker		on 19 May 2009	1 186 470	1044	61
Agnes Ro		on 19 May 2009	58 793	1161	68

The list of all people shows all users that have accounts on the system. By default, each user has a unique avatar that is randomly formed from the text in their email address. Users can choose to upload their own avatar image by uploading an image to an external service such as Gravatar, which Crucible supports. See the page on [Changing your User Profile](#).

Viewing a Person's Activity Screen

Click on a user to see a listing of activity from them as well as charts showing statistics for their activity. The People Activity screen opens.

Screenshot: The People Activity Screen in Crucible



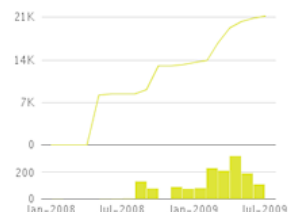
About
jxie@atlassian.com
First commit 20 January 2008
Latest commit 19 June 2009

	Last Week	All Time
Commits:	3	616
Files changed:	5	1,550
Change in LOC:	11	21,141


Follow Joe Xie

Reviews
1 active reviews

Line History



Commit Activity



Committer Mappings
Username jxie is mapped to:
• jxie in FE

Activity
All Activity | Commits | Reviews | Issues

replied to Sebastian Ruiz in [CR-FE-1939](#)
done 04 Jun

replied to Nicolas Venegas in [CR-FE-1939](#)
yes. 04 Jun

summarized and closed [CR-FE-1898](#)
refactor of activitiystreams is in progress. 04 Jun

finished reviewing [CR-FE-1960](#) 04 Jun

commented on [CR-FE-1960](#)
being repeated on several pages, i wonder if its worth taggifying this. 04 Jun

replied to Nicolas Venegas in [CR-FE-1960](#)
its probably good to have it at the upper right hand corner. 04 Jun

finished reviewing [CR-FE-1942](#) 04 Jun

finished reviewing [CR-FE-1953](#) 04 Jun

replied to Sebastian Ruiz in [CR-FE-1953](#)
is it worth trying to get this change into AUI? our fork is different, but it doesnt feel different enough to me to require a complete fork - what do you guys think of the chances of merging these changes back into AUI? 04 Jun

finished reviewing [CR-FE-1941](#) 04 Jun

replied to Joe Xie in [CR-FE-1941](#)
The other tests run fine, but ProjectConstraintBuilderTest could do with a small refactor to creator instead of reviewer in the call to create the review. Other than that, looks ok. 04 Jun

finished reviewing [CR-FE-1947](#) 03 Jun


In the right hand pane, we can see a list of all activity that relates to this user. You can click the icons to view full commit information in FishEye, click JIRA issue names to open the work ticket on an item, click the [long] button to see the list of files in context or click the [star] icon to favourite

an item.

In the left hand pane, we can see charts around this activity, such as the following: number of active reviews; charted history of lines of code; code committing activity and general statistics.

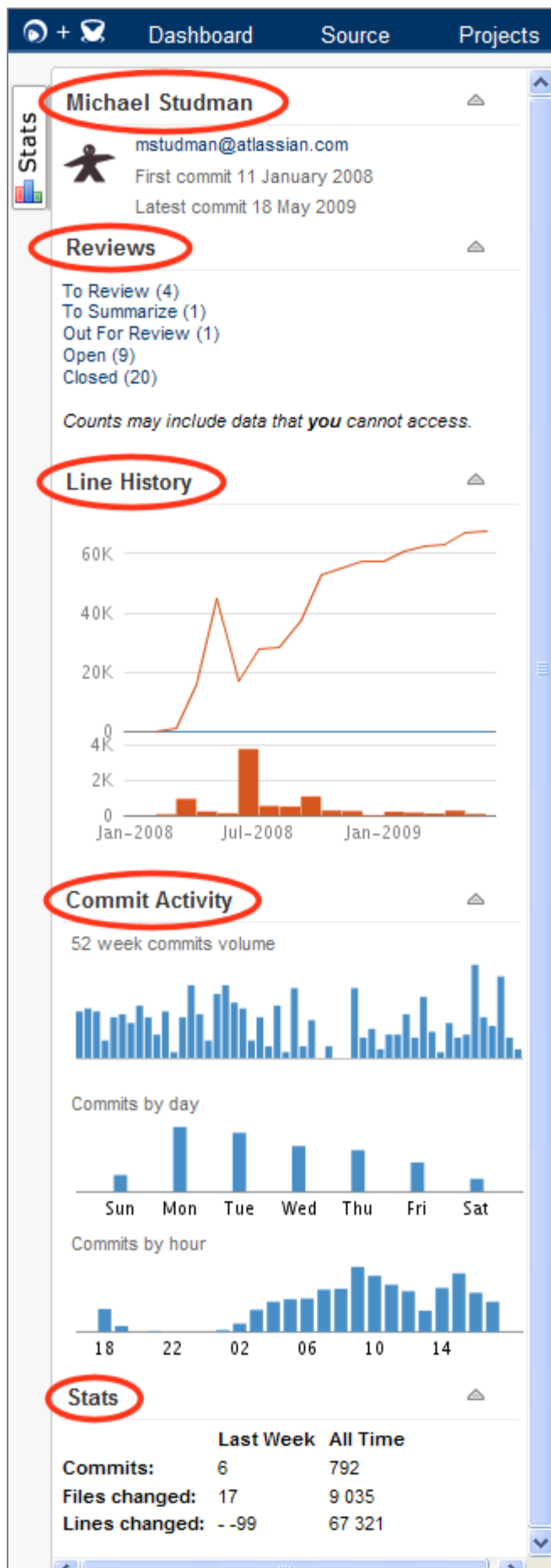
Viewing Charts on a Person's Activity

To see information on a person's activity charted in detail, click the headings in the left-hand pane. Each heading will show more information on demand, when clicked. The information available and what it means is listed below.

 The Charts in this section are only available when using FishEye.

Screenshot: People Activity Charts in Crucible

	<ul style="list-style-type: none">• Username heading: The username section shows the email address, then the first and latest commit dates for the person in context. It also shows username mappings from various systems if they have several usernames in play.• Reviews heading: The Reviews section shows several filters that you can click to constrain the review items shown in the right-hand pane. The options are To Review, To Summarize, Out For Review, Open and Closed.• Line History heading: The Line History section shows a graph with the number of lines committed to the repository, charted over time.• Commit Activity heading: The Commit Activity section shows three smaller charts; the first showing the volume of commits over a 52 week period; the second showing the relative number of commits on days of the week; the third showing the relative number of commits by the hour of the day when they were lodged.• Stats heading: The Stats section shows data points for the previous week and all-time. It shows number of commits, number of files changed and number of lines changed.
--	--



Searching in Crucible

Quick Search is the search box in the top right-hand corner of the screen.

You can find a review by searching for:

- Review IDs - must be in the format CR-xx
- Contents of Title
- Contents of Statement of Objective
- Contents of a comment

To carry out an advanced search, click the '**Reviews**' Tab.

Using the Filters Navigation Bar

'Everyones Reviews' Filters

Any reviews (even if the user has not participated in them) that have been created can be viewed, under the '**Everyone's Reviews**' section. The following options are available:

- All open reviews.
- All closed reviews.
- All reviews.

Custom Filters

To find a specific review, use the '**Custom Filter**':

Title	Find reviews by searching for words within the title.
Project	Find reviews under a particular project.
Author	Find reviews moderated by a particular authors .
Moderator	Find reviews moderated by a particular moderators .
Creator	Find reviews created by a particular creator .
Reviewer	Find reviews that are reviewed by a particular reviewer . This will default to the user logged in.
Reviewer Status	This is reliant on the above filter and is used to show reviews that have either been completed by the reviewer, not completed or all reviews.
Match Roles	To use all the above filters, choose ' all '. To use any of the filters, choose ' any '.

Review State Filter

You can use the checkboxes below with the above filters or on their own.

Draft	Reviews that are still in 'Draft' state .
Pending Approval	Reviews that have been moved out of 'Draft' state and are now waiting for the moderator to approve .
Under Review	The review is now 'Under Review'.
Summarize	The review is now in 'Summarize' state.
Closed	Reviews that are now 'Closed'.
Abandoned	Any reviews that are no longer relevant and can be deleted.
Rejected	Any reviews that a moderator has rejected.
Review needs fixing	A review will match this filter if the review enters into an undefined state because something went wrong with storing the review state . A moderator can use this filter to find the review and then change the state to something sensible.

Screenshot: Advanced Search Filter Options

Everyone's Reviews

All Open Reviews (64)

All Closed Reviews (1948)

All Reviews (2126)

Custom Filter

Title

Project

any

Author

any

Moderator

any

Creator

any

Reviewer

any

Reviewer Status

any

Match

all

Roles

☐ Draft

☐ Pending Approval

☐ Under Review

☐ Summarize

☐ Closed

☐ Abandoned

☐ Rejected

☐ Review needs fixing

Cancel


Apply Filter

Search Reviews

Using RSS Feeds in Crucible

Subscribing to an RSS Feed

In Crucible, all pages with an activity stream and any page which has a list of reviews will have an RSS option.

To access the RSS feed for a page, open the **'Tools'** drop-down menu at the top right corner of the screen, then click the **'RSS'**  option.

This will open a page with the RSS feed displayed; you can also paste the URL from that page into your RSS reader of choice.

Using Keyboard Shortcuts in Crucible

Keyboard shortcuts are available for most of the commonly-used functions in Crucible.

To see a list of available shortcuts, open the '**Tools**' drop-down menu at the top right corner of the screen, and select the '**Keyboard Shortcuts**' option.

Getting Started with Crucible

This page contains a basic overview of Crucible workflows, followed by a simple example showing a code review between two people.

Crucible is a flexible application that caters for a wide range team sizes and work styles. You will need to know about the basic roles used in Crucible.

Roles:

There are several roles that review participants can take up:

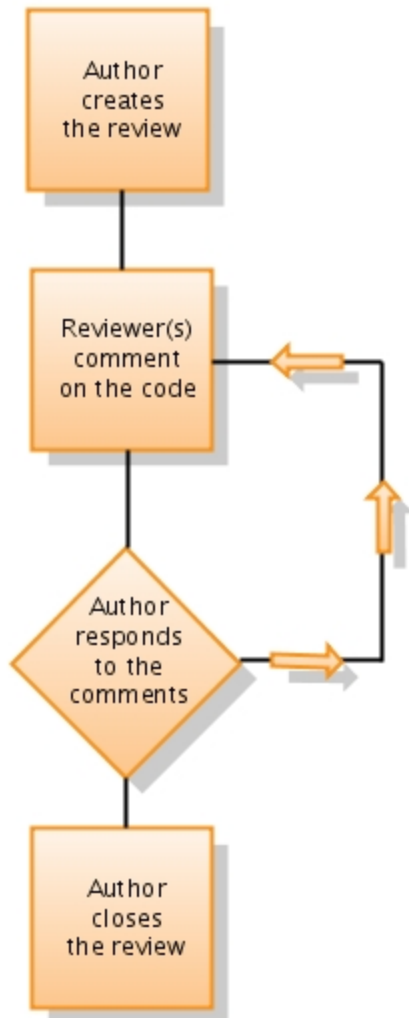
- **Author:** Usually the creator of the code; the person who will act on the review's outcome.
- **Reviewer:** A participant that will comment on the source files in the review, raising points and discussion on the work that was done.
- **Moderator:** Usually the person who starts the review and is responsible for deciding the outcomes and closing it.

You will also need to understand how workflow is conducted in Crucible. This is configurable, but the most basic example follows.

Crucible Workflow:

There are a number of different ways in which you can use Crucible for code reviews. The following diagram shows the basic workflow that applies to most Crucible code reviews.

Diagram: Workflow for One-to-One Reviews




Need more information? Read more about the different forms of [workflow in Crucible](#).

Next, we explore the workflow in a two-person code review in Crucible.

Example Workflow: Two Participant Code Review

This is a simplified set of instructions for executing a one-to-one review involving two people. In this example, the code author wears "three hats", acting as [review creator](#), [moderator](#) and [code author](#), managing the review process as well as taking final responsibility for closing the review. The second person is the reviewer.

- [Example Workflow: Two Participant Code Review](#)
 - 1. The Author Starts the Review
 - 2. The Reviewer Comments on the Code
 - 3. The Author Responds to the Comments
 - 4. The Author Closes the Review

 For instructions on Crucible workflow with more than two people, see [this page](#).

1. The Author Starts the Review

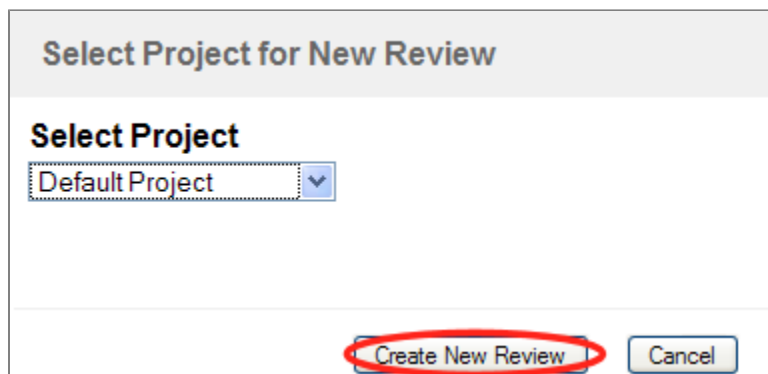
To begin, the code author sets up the review. There are a [number of ways to do this](#), but for this example, the author starts from the FishEye Source view of the file he wants to review:

Screenshot: Opening a review from the FishEye Source view



From the FishEye Source view, the author clicks the '**Reviews**' drop-down menu above the source view, then selects '**Create New Review**'. If there are multiple projects, the Project Selection dialog opens.

Screenshot: The Project Selection dialog



In the Project Selection dialog, you are prompted to choose a project for this review from the drop-down list. Once the selection is made, the author clicks the '**Create New Review**' button. The Manage Files dialog opens.

Screenshot: The Crucible Manage Files dialog

Manage Files

Review Content

Change Sets

Files

Search

Suggestions

Patches

Uploads

Repository: CLOV Author: (any) Branch: (any) Tag:

Add to Review as: Diff to Previous Version Remove all revisions from review

Go to changeset: End of Changesets

Earlier Changesets

Files in CLOV/

☒ 35536 created by mstudman on 12 May 2009, 15:47:15 -0500 (6 hours ago)
CLOV-378: Making back refs transient in lower-level model objects and pushing dataLength field to MethodInfo out of ElementInfo. -5% model serialization/deserialization time, -8% db file size, +5 Str...

- ☒ /trunk/core/src/com/cenqua/lover/registry/BranchInfo.java 35536 (+10 -3) diffs
- ☒ /trunk/core/src/com/cenqua/lover/registry/MethodInfo.java 35536 (+28 -2) diffs
- ☒ /trunk/core/src/com/cenqua/lover/registry/StatementInfo.java 35536 (+10 -2) diffs
- ☒ /trunk/core/src/com/cenqua/lover/registry/ElementInfo.java 35536 (+0 -9) diffs
- ☒ /trunk/core/src/com/cenqua/lover/registry/ClassInfo.java 35536 (+10 -0) diffs

☐ 35388 created by mstudman on 11 May 2009, 09:00:34 -0500 (37 hours ago)
CEP-325: Making coverage cloud work properly on windows after clicking a cloud link.

- ☐ /trunk/eclipse/com.cenqua.lover.core/src/com/cenqua/lover/eclipse/editors/cloud/CloudHtmlReporter.java 35388 (+7 -1) diffs
- ☐ /trunk/eclipse/com.cenqua.lover.core/src/com/cenqua/lover/eclipse/editors/cloud/EditorLinkingHtmlRenderingSupport.java 35388 (+5 -5) diffs

☐ 35381 created by mstudman on 11 May 2009, 04:43:43 -0500 (41 hours ago)
CEP-326: Ignoring CCE problem when plugin is upgraded but Eclipse is not yet restarted.

- ☐ /trunk/eclipse/com.cenqua.lover.core/src/com/cenqua/lover/eclipse/projects/model/CoverageModelsMonitor.java 35381 (+4 -0) diffs

☐ 35367 created by npellow on 11 May 2009, 01:37:00 -0500 (45 hours ago)
NONE: creating branch for 2.5.x

Note: only showing 100 of 2997 files in this changeset.

- ☐ /branches/lover-2.5.x 35367 (+0) new
- ☐ /branches/lover-2.5.x/build.xml 35367 (+69) new
- ☐ /branches/lover-2.5.x/lib 35367 (+0) new
- ☐ /branches/lover-2.5.x/lib/junit-3.8.1.jar 35367 (+0) new

Abandon Review Done

In the Manage Files dialog, the author selects the source files they want to include in the review, by clicking the checkboxes next to the desired files. Once finished, the author clicks 'Done'. The Edit Review dialog appears, where the author can create and issue the review.

Screenshot: Creating a review in the Edit Review dialog

Edit Review

Title: Test Review for Documentation

Project: Clover Moderator: Edwin Dawson Author: Edwin Dawson

Reviewers: ☐ Allow anyone to join

☒ Geoff Crain

Start typing the reviewer's name then press enter to select.

Close Suggestions

User	Most Contribution (%)	Total LoCOpen Reviews
brendan(brendan)	ClassInfo.java (84%)	526 0
Nick Pellow(npellow)	ClassInfo.java (1%)	3 0
Michael Studman(mstudman)	BranchInfo.java (41%)	64 3
Brendan Humphreys(bhumphreys)	MethodInfo.java (1%)	3 1

Due date: 2009-05-14T00:00:00

Link to Jira Issue:

Issue key: Verify

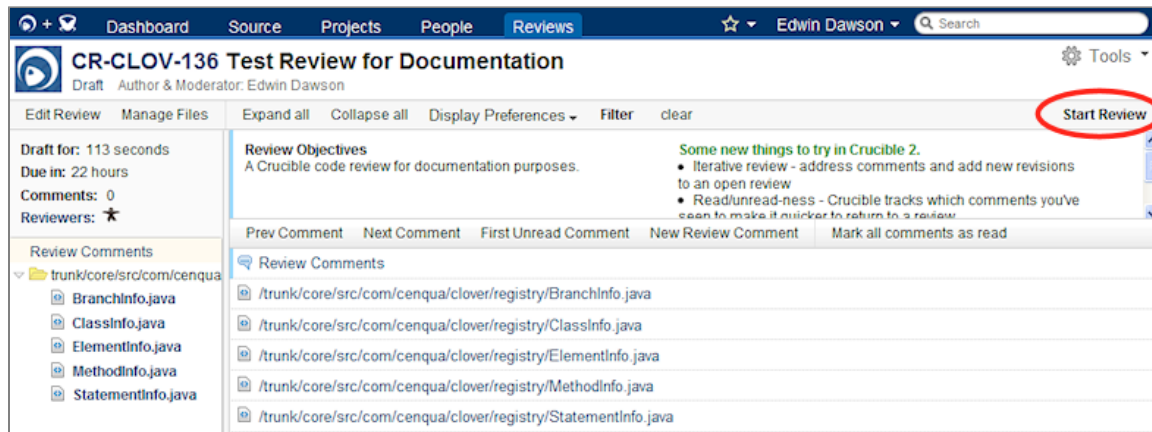
Statement of Objectives: A Crucible code review for documentation purposes.

Abandon Review Save

In the Edit Review dialog, the author enters information needed for the review. This includes entering a title and description for the review,

selecting reviewers, a due date and the key for a related JIRA issue (if any). The project, moderator and author are pre-selected (for this example, the author should select himself as a moderator. When finished, the author clicks **'Save'**. The review will now be created in a draft form.

Screenshot: A newly created Crucible review



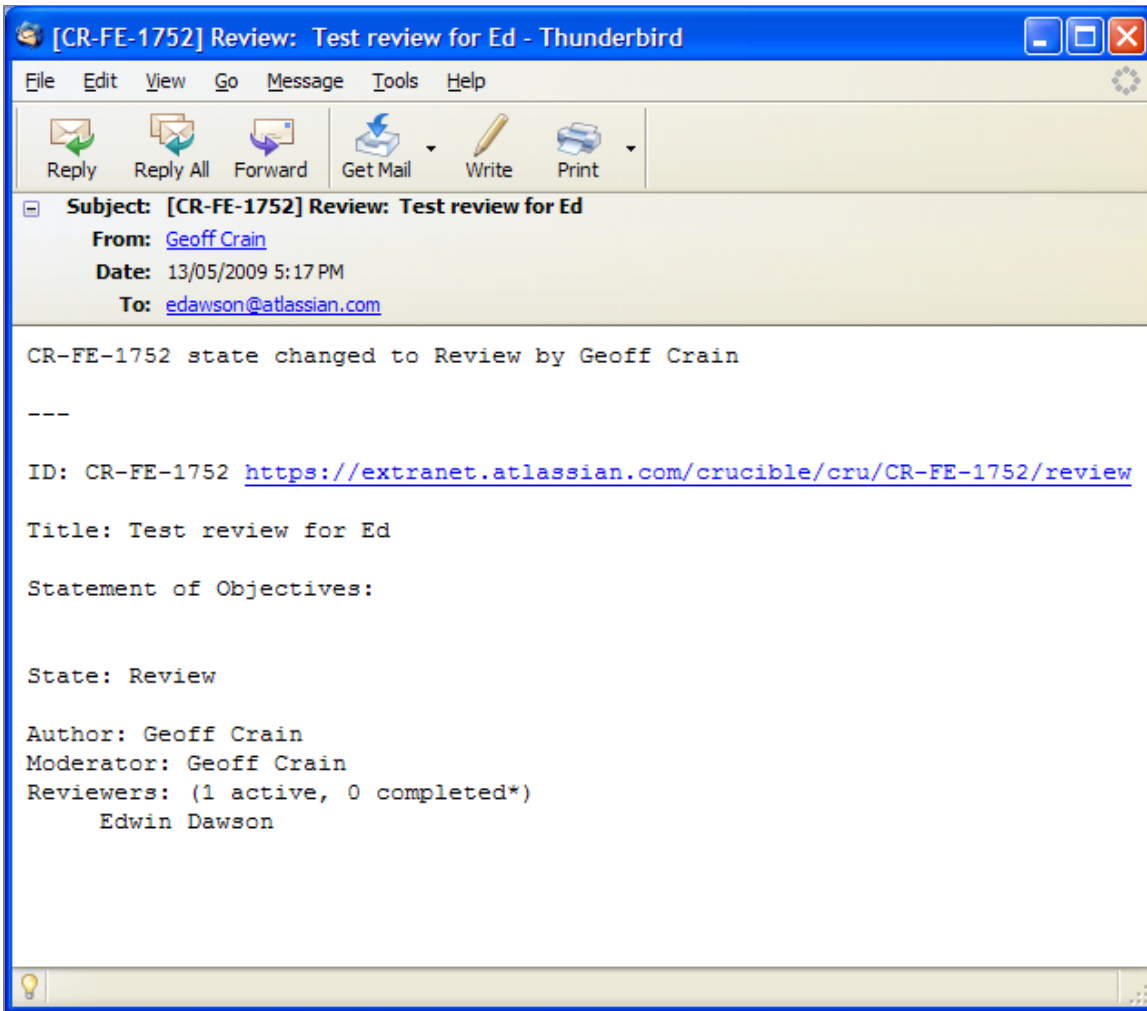
The draft review opens. In the draft stage, the author can check the contents of the review files to ensure they are correct and put in any notes for reviewers as comments. During the draft phase, no notification email is sent out to reviewers. Once the author is finished with the draft phase, he clicks **'Start Review'**.

The review will now be started and notification email will go out to all participants. Crucible will now send out an email notification to all the participants. This lets them know that the review is under way and prompts them to take action, providing a URL for direct access to the review. (You can also [subscribe to an RSS feed](#).)

2. The Reviewer Comments on the Code

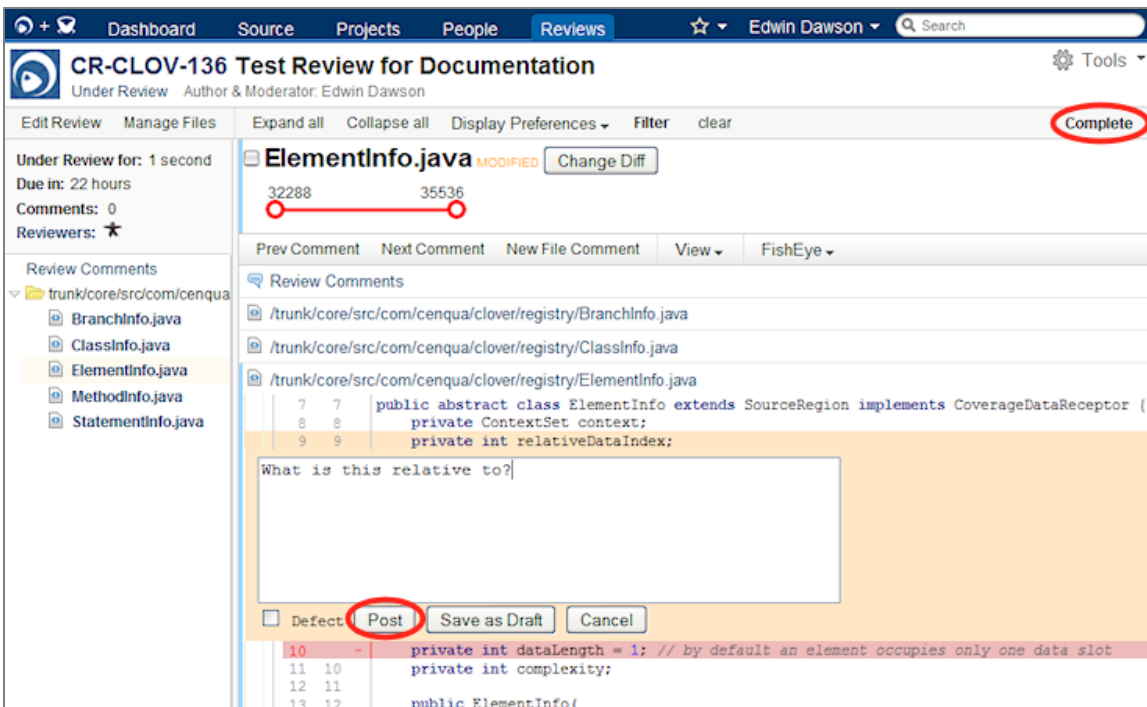
The reviewer will receive an email from Crucible (or an [RSS feed](#) update) with a link that they can follow to the review.

Screenshot: A Crucible review notification email



When the reviewer clicks the link in the notification email, the Crucible Review screen opens.

Screenshot: The Crucible Review screen




On the Crucible Review screen, the code changes under review are displayed. The reviewer clicks filenames to expand the code for in-line reviewing. As the reviewer reads the changes, they can simply click on any line to enter a comment there (multiple lines can be selected by clicking and dragging).

The reviewer clicks the **'Post'** button when each comment is finished.

The reviewer repeats this process for all files in the review. Reviewers can leave the session and resume it later; their work is automatically saved.

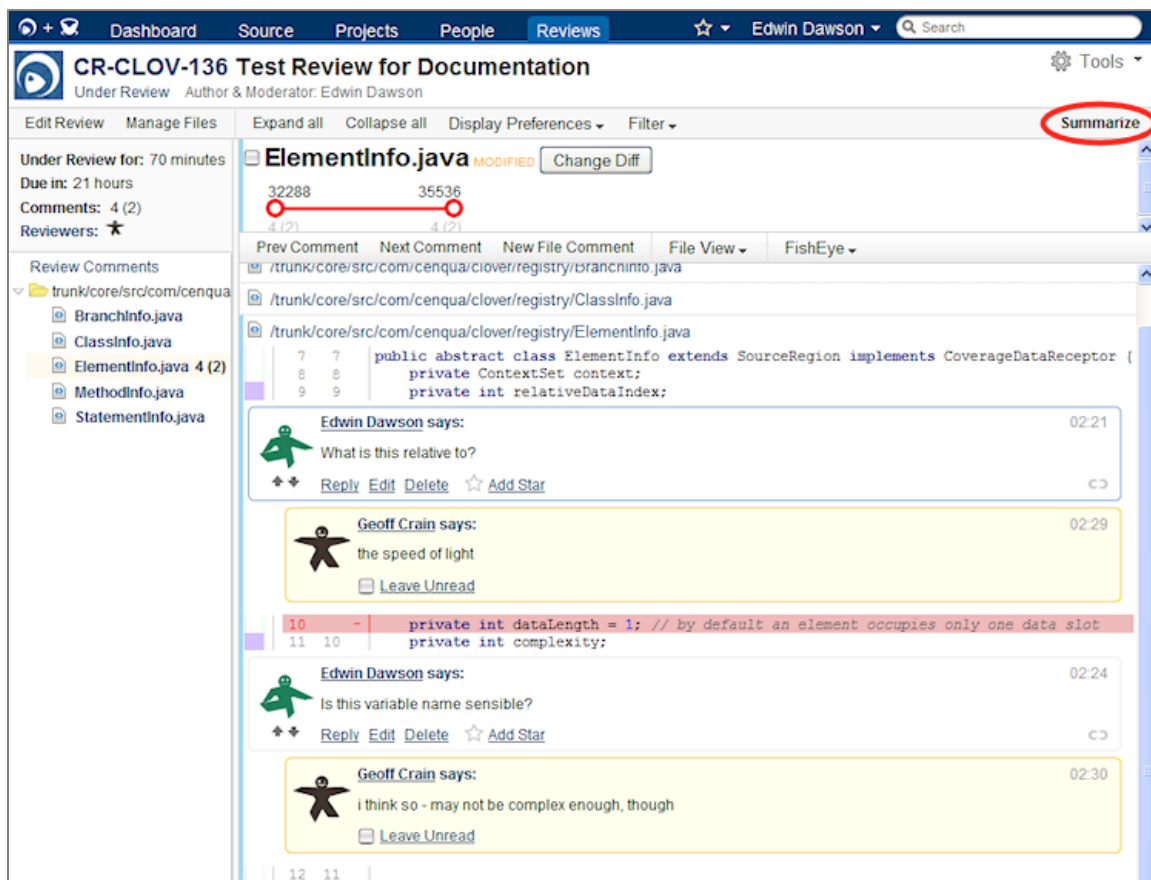
When the reviewer has finished their code review work, they click the  button.

 By default, an email is sent to participants every time a comment is posted. This is an individual setting. Each reviewer can configure their own profiles to adjust the list of events that will trigger email notifications.

3. The Author Responds to the Comments

During the review process, the author/moderator can also make contributions, responding to reviewer comments and making corrections.


Screenshot: Comment threads in Crucible



The screenshot shows the Crucible interface for a review session titled "CR-CLOV-136 Test Review for Documentation". The reviewer is Edwin Dawson. The review is for 70 minutes, with 21 hours remaining. There are 4 comments and 2 reviewers. The file being reviewed is `ElementInfo.java`, which has been modified. The code diff shows changes to the `ElementInfo` class, which extends `SourceRegion` and implements `CoverageDataReceptor`. The code includes fields for `ContextSet` and `relativeDataIndex`. The comment threads show Edwin Dawson asking "What is this relative to?" and "Is this variable name sensible?". Geoff Crain responds with "the speed of light" and "I think so - may not be complex enough, though". The `Summarize` button is circled in red in the top right corner.

4. The Author Closes the Review

When all reviewers have *Completed* their reviews, the author/moderator is notified via email. The author/moderator clicks the link in the notification email, returning to the Review screen.

The author/moderator will then add any final comments, then click the  button when finished. The Crucible Summarize Review screen opens.

Screenshot: Summarizing a review in Crucible

Summarize Review

Summarize the review outcomes (optional)

The review was a great success.

Continue Without Closing
Close Review

On the Crucible Summarize Review screen, the author/moderator enters an optional summary of the review's results, then clicks [Close Review](#). This closes the review, signalling the end of work. A final email notification will be sent to the review participants, informing them that the review is now closed. The closed review screen will load, displaying the summary and archiving the completed review as read-only.

Screenshot: Viewing a closed review

Dashboard
Source
Projects
People
Reviews
Edwin Dawson
Search

CR-CLOV-136 Test Review for Documentation

Closed Author & Moderator: Edwin Dawson

Edit Review
Manage Files
Expand all
Collapse all
Display Preferences
Filter
Reopen

Closed on: 13 May 2009
Due in: 3 hours
Comments: 4
Reviewers: ☆

Review Comments
trunk/core/src/com/cenqua/
BranchInfo.java
ClassInfo.java
ElementInfo.java 4
MethodInfo.java
StatementInfo.java

ElementInfo.java MODIFIED Change Diff
32288 35536
Prev Comment Next Comment File View FishEye

Review Comments
/trunk/core/src/com/cenqua/registry/BranchInfo.java
/trunk/core/src/com/cenqua/registry/ClassInfo.java
/trunk/core/src/com/cenqua/registry/ElementInfo.java
7 7 public abstract class ElementInfo extends SourceRegion implements CoverageDataReceptor {
8 8 private ContextSet context;
9 9 private int relativeDataIndex;

Edwin Dawson says: 02:21
What is this relative to?
Add Star

Geoff Crain says: 02:29
the speed of light

If the author/moderator ever needs to resume work on the closed review, they can simply click [Reopen](#) when viewing this screen. Doing this will return the review's status to "open".

For more information on workflow in Crucible and best practices for code reviews, see [Requesting and Conducting a Review](#).

Using the Dashboard

This page contains information on browsing reviews in Crucible.

On this page:

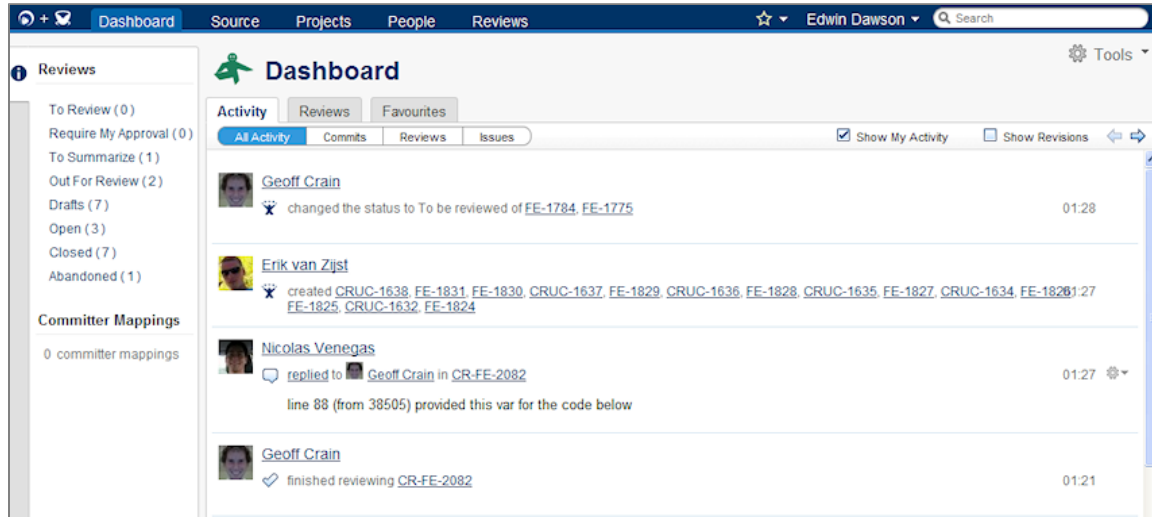
- Dashboard Screen Overview

- [Filtering Your View](#)

Dashboard Screen Overview

To browse all your reviews (reviews you are participating in), use the **Dashboard** tab at the top of the page. The Dashboard Screen opens, as shown in the screenshot below.

Screenshot: The Dashboard Screen in Crucible



By default, the Activity Stream is shown. This is a mix of all activity that is occurring related to Crucible, such as people making review comments, reviews opening and closing, files being committed to a linked repository, or updates to linked JIRA issues.

Filtering Your View

To filter your view, use the constraint options in the sub-nav and the side panel.

Activity Stream filters

You can also filter the items shown in the Activity Stream. To do this, click one of the options in second layer of tabs, as listed in the table below.

Tab Name	Sub-Nav Options
Activity Tab	Sub-Nav Options: <ul style="list-style-type: none"> • All Activity — Shows all activity with no filtering. • Commits — Shows commits. • Reviews — Shows reviews. • Issues — Shows JIRA issues. • Show/Hide My Activity — Show/hide your own activity. • Show/Hide Revisions — Show/hide activity from other users. • Earlier / Later Activity (arrow buttons) — Pages through activity items, stepping backward or forward through the results.
Reviews Tab (Shows the result of the filter clicked in the left side panel)	Sub-Nav Options: <ul style="list-style-type: none"> • RSS — Opens a page with the RSS feed for the current selection.
Favourites Tab (Shows all items you have marked as a 'favourite'.)	Sub-Nav Options: None.

Side Panel


The left navigation panel of the Dashboard shows the number of reviews in different states. Click on any of these states to show the list of reviews in the left-hand panel.

To Review	Reviews where the user still needs to complete their work.
------------------	--

Require My Approval	The user has been assigned the role of moderator for these reviews and needs to approve them.
To Summarize	The user has been assigned the role of moderator for these reviews and needs to summarise and close them.
Out For Review	Reviews created by the user that are currently in progress.
Drafts	These are reviews created by the user that have not yet been moved to the 'Approval' or the 'Require Approval' states .
Open	All open reviews that the user is participating in.
Closed	These are reviews that the user has been involved in and are now closed.
Abandoned	Reviews that are no longer relevant and can be deleted.

Using the Project Dashboard

To browse the content in a project, click the **Projects** tab at the top of the page. The '**Projects**' view opens.

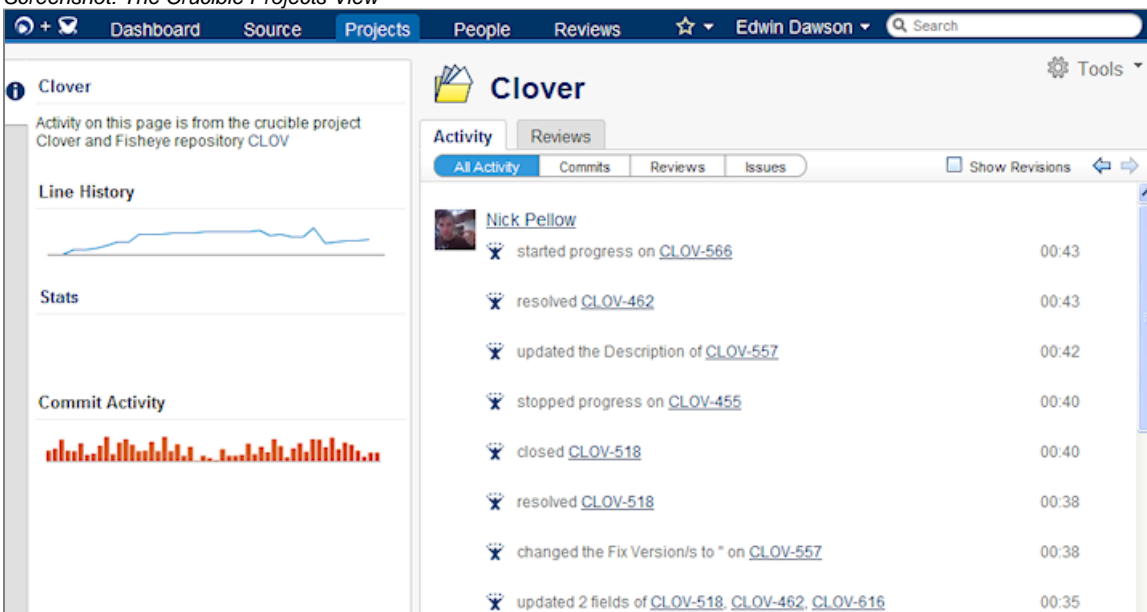
 The Projects tab is only visible in Crucible. Read more about the [definition of a project](#).

A list of projects will be shown if there is more than one. Click the name of the desired project to open it. The '**Project Activity**' page opens. In the left navigation bar, charts showing overall project statistics are displayed.

There are a number of sub-tabs on this page, listed in the table below.

Sub-Tab Name	Description
Activity	<ul style="list-style-type: none"> All Activity — The default view. Commits — Shows commits in the project (visible when using FishEye). Reviews — Shows reviews in the project. Issues — Shows JIRA issues related to this project. Show Revisions — Shows or hides revisions in the project (visible when using FishEye). Earlier Activity (Left Arrow icon) — Loads a page of earlier project activity. Later Activity (Right Arrow icon) — Loads a page of later project activity.
Reviews	Shows recent reviews in the project.

Screenshot: The Crucible Projects View



Screenshot: The Crucible Projects Index

Dashboard Source Projects People Reviews ☆ Edwin Dawson Search	
Projects	
Project name	Latest Activity
☆ Default Project	📅 on 25 Jun 2009
☆ Confluence Hosted	🌐 on 25 Feb 2009
☆ Jira Studio	🌐 on 22 Apr 2008
★ FishEye	🔍 on 25 Jun 2009
☆ Clover	🔍 on 25 Jun 2009
☆ Clover IDEA plugin	📅 on 25 Jun 2009
☆ Testing Project	💬 on 23 Jun 2009
☆ Clover Eclipse Plugin	📅 on 25 Jun 2009

Conducting a Review

This page contains links to instructions how to create a review and manage the workflow through its various states to completion. Click on the desired topic to see more information.

- [Creating a Review](#)
- [Adding Reviewers](#)
- [Issuing a Review](#)
- [Performing the Review](#)
- [Summarising and Closing the Review](#)
- [Deleting an Abandoned Review](#)
- [Moving a Review to Another Project](#)

For an overview of how to apply a workflow to Crucible, see [Defining your Workflow](#).

For an explanation of the different roles that people play in a review, see [Roles and Status Classifications](#).

Creating a Review

This page explains how to create a Crucible review.

There are a number of ways to create a review. Choose from the list below:

- [Creating a Review from FishEye](#)
- [Creating a Review from JIRA](#)
- [Creating a Review within Crucible](#)
- [Creating a Review from a URL](#)
- [Creating a Patch Review](#)
- [Selecting the Files for the Review](#)

 Note that only people with the '**Create**' permission can create a review.

Creating a Review from FishEye

This page explains how to create a Crucible review from FishEye.

On this page:

- [Opening the FishEye Source View](#)
- [Starting the Review](#)
- [Choosing a Project](#)
- [Selecting Files for Review](#)

Opening the FishEye Source View

To create a review from within FishEye,

To begin, the code author sets up the review. There are a number of ways to do this, but for this example, the author starts from the FishEye Source view of the file he wants to review:

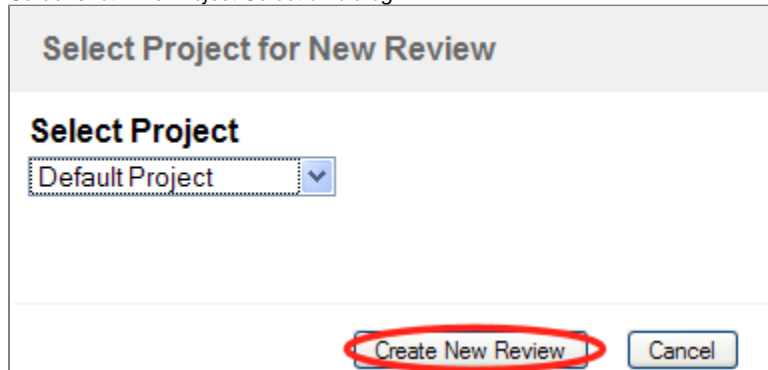
Screenshot: Opening a review from the FishEye Source view



Starting the Review

From the FishEye Source view, the author clicks the '**Reviews**' drop-down menu above the source view, then selects '**Create New Review**'. If there are multiple projects, the Project Selection dialog opens.

Screenshot: The Project Selection dialog



Choosing a Project

In the Project Selection dialog, you are prompted to choose a project for this review from the drop-down list. Once the selection is made, the author clicks the '**Create New Review**' button. The Manage Files dialog opens.

Screenshot: The Crucible Manage Files dialog

Manage Files

Review Content

Change Sets

Files

Search

Suggestions

Patches

Uploads

Repository: CLOV Author: (any) Branch: (any) Tag:

Add to Review as: Diff to Previous Version Remove all revisions from review

Go to changeset: End of Changesets

Earlier Changesets

Files in CLOV/

☒ 35536 created by mstudman on 12 May 2009, 15:47:15 -0500 (6 hours ago)
CLOV-378: Making back refs transient in lower-level model objects and pushing dataLength field to MethodInfo out of ElementInfo. -5% model serialization/deserialization time, -8% db file size, +5 Str...

☒ /trunk/core/src/com/cenqua/lover/registry/BranchInfo.java 35536 (+10 -3) diffs

☒ /trunk/core/src/com/cenqua/lover/registry/MethodInfo.java 35536 (+28 -2) diffs

☒ /trunk/core/src/com/cenqua/lover/registry/StatementInfo.java 35536 (+10 -2) diffs

☒ /trunk/core/src/com/cenqua/lover/registry/ElementInfo.java 35536 (+0 -9) diffs

☒ /trunk/core/src/com/cenqua/lover/registry/ClassInfo.java 35536 (+10 -0) diffs

☐ 35388 created by mstudman on 11 May 2009, 09:00:34 -0500 (37 hours ago)
CEP-325: Making coverage cloud work properly on windows after clicking a cloud link.

☐ /trunk/eclipse/com.cenqua.lover.core/src/com/cenqua/lover/eclipse/editors/cloud/CloudHtmlReporter.java 35388 (+7 -1) diffs

☐ /trunk/eclipse/com.cenqua.lover.core/src/com/cenqua/lover/eclipse/editors/cloud/EditorLinkingHtmlRenderingSupport.java 35388 (+5 -5) diffs

☐ 35381 created by mstudman on 11 May 2009, 04:43:43 -0500 (41 hours ago)
CEP-326: Ignoring CCE problem when plugin is upgraded but Eclipse is not yet restarted.

☐ /trunk/eclipse/com.cenqua.lover.core/src/com/cenqua/lover/eclipse/projects/model/CoverageModelsMonitor.java 35381 (+4 -0) diffs

☐ 35367 created by npellow on 11 May 2009, 01:37:00 -0500 (45 hours ago)
NONE: creating branch for 2.5.x

Note: only showing 100 of 2997 files in this changeset.

☐ /branches/lover-2.5.x 35367 (+0) new

☐ /branches/lover-2.5.x/build.xml 35367 (+69) new

☐ /branches/lover-2.5.x/lib 35367 (+0) new

☐ /branches/lover-2.5.x/lib/junit-3.8.1.jar 35367 (+0) new

Abandon Review Done

Selecting Files for Review

In the Manage Files dialog, the author selects the source files they want to include in the review, by clicking the checkboxes next to the desired files. Once finished, the author clicks 'Done'. The Edit Review dialog appears, where the author can create and issue the review.

The next step is to add reviewers.

Creating a Review from JIRA

This page explains how to create a Crucible review directly from JIRA, the Atlassian issue-tracker application.

To create a review from within JIRA, click the small Crucible icon next to the required changeset on the 'FishEye' tab.

Screenshot: Adding a Review from within JIRA

All Comments Work Log Change History FishEye Crucible Sort Order:

Create crucible review for all 2 changesets in Default Project

22879 by Craig Sharkie [Atlassian] (2 files) 30/Oct/08 11:03 PM (1 month, 7 days ago)

CRUC-771: Responses to issues raised in comments.

- Reverted date format in simpleform and
- Removed unused code from create_selectPatch and updated grammar

branches/1.6.3/src/content/WEB-INF/jsp/crucible/create/create_selectPatch.jspf 22879 (+2 -8) diffs

branches/1.6.3/src/content/WEB-INF/jsp/crucible/create/search/simpleform.jspf 22879 (+2 -2) diffs

When you click the icon, you will be prompted to select the relevant project(if more than one project exists) in which to create your review. A new draft review will then be created, including the following information pre-filled:

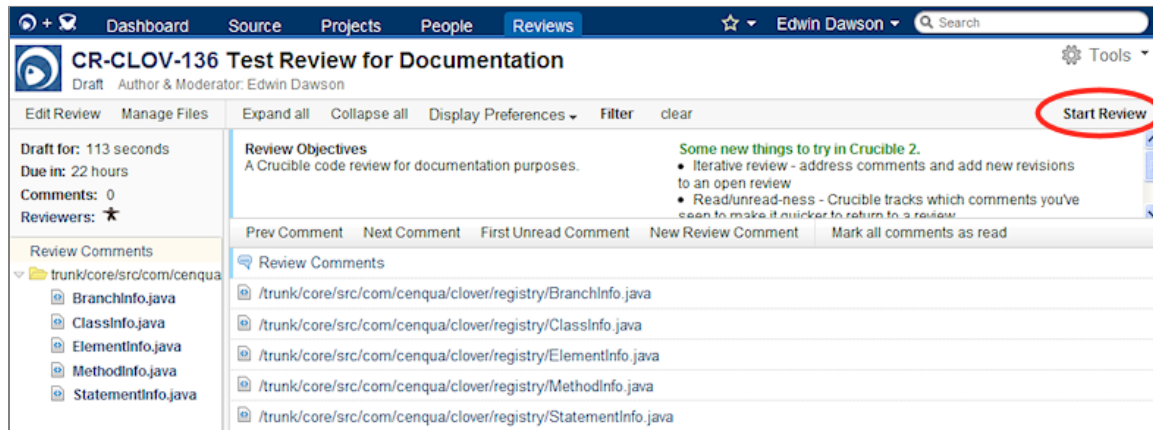
- The content of the changeset becomes the content (i.e. files) to be reviewed.
- The author of the changeset becomes the author of the review, if Crucible is aware of this user. Otherwise the creator of the review becomes the author.
- The creator of the review becomes the moderator.

- The commit log message is used as both the Title and [Statement of Objective](#).

All aspects of the review can be changed. To edit any of the above settings, click the title to see the '**Edit details**' screen. Or you can click the [Manage Files](#) tab.

If you click the Crucible icon, you will see the '**Review**' screen below:

Screenshot: Review Screen in Crucible



The next step is to [add reviewers](#).

Creating a Review within Crucible

This page explains how to create a review from the Crucible interface.

On this page:

- [Open the Reviews Screen in Crucible](#)
- [Next Steps](#)

Open the Reviews Screen in Crucible

Within Crucible, open the Reviews Screen by clicking the **Reviews** Tab. You can now create a review by opening the **Tools** menu at the top right of the Reviews screen, then selecting **Create Blank Review**. The **Edit details** screen appears, as shown below.

Screenshot: Adding a review in Crucible

- The [creator](#) of the review becomes the [moderator](#).
- The commit log message is used as both the Title (unless you have explicitly defined a title in your URL) and [Statement of Objective](#).

All aspects of the review can be changed. To edit any of the above settings, click the title to see the '**Edit details**' screen. Or you can click the [Manage Files](#) tab.

The next step is to [add reviewers](#).

Creating a Patch Review

This page includes instructions on generating patch files from your repository, and how to load them into Crucible to be reviewed.

On this page:

- [Loading a Patch Into Crucible](#)
- [Creating a Patch File From Your IDE](#)
 - [Creating a Patch File in IntelliJ IDEA 7.0](#)
 - [Creating a Patch File in Eclipse 3.3.1.1](#)
- [Creating a Basic Patch File From The Repository Command Line](#)
 - [CVS Patch Creation On The Command Line](#)
 - [Subversion Patch Creation Via The Command Line](#)
 - [Perforce Patch Creation Via The Command Line](#)
- [Creating Patches That Include All Lines of Code](#)
 - [Creating a Patch in CVS With All Lines of code](#)
 - [Creating a Patch in Subversion With All Lines of Code](#)
 - [Creating a Patch in Perforce With All Lines of Code](#)

Crucible allows you to review a change before it has been committed. To do this, you upload a patch file to the '**Patch**' tab (or paste it in as text) when [creating a review](#). You must first generate this patch file from your repository, using either commands built into your IDE, or via the repository command-line tools.

By default, patch files will only show a few lines of code surrounding each change, rather than the entire file and its changes. However, you can create a patch that includes all of the original files' code by using specific parameters ([listed on this page](#)) when creating a patch with the command-line tools for your repository.

Crucible can accept a patch file created from any version control system, but instructions on this page cover only Subversion, CVS and Perforce repositories.

Loading a Patch Into Crucible

Once you have generated the patch file, loading patch files into Crucible is easy.

Screenshot: The Crucible Patch tab

If you already have the patch file you want to review, open the Patches View by clicking '**Manage Files**' on the review toolbar, then click '**Patches**' on the left navigation bar of the Manage Files view. From there, simply click '**Browse**' to select the file and click '**Upload**' to add it to your review.

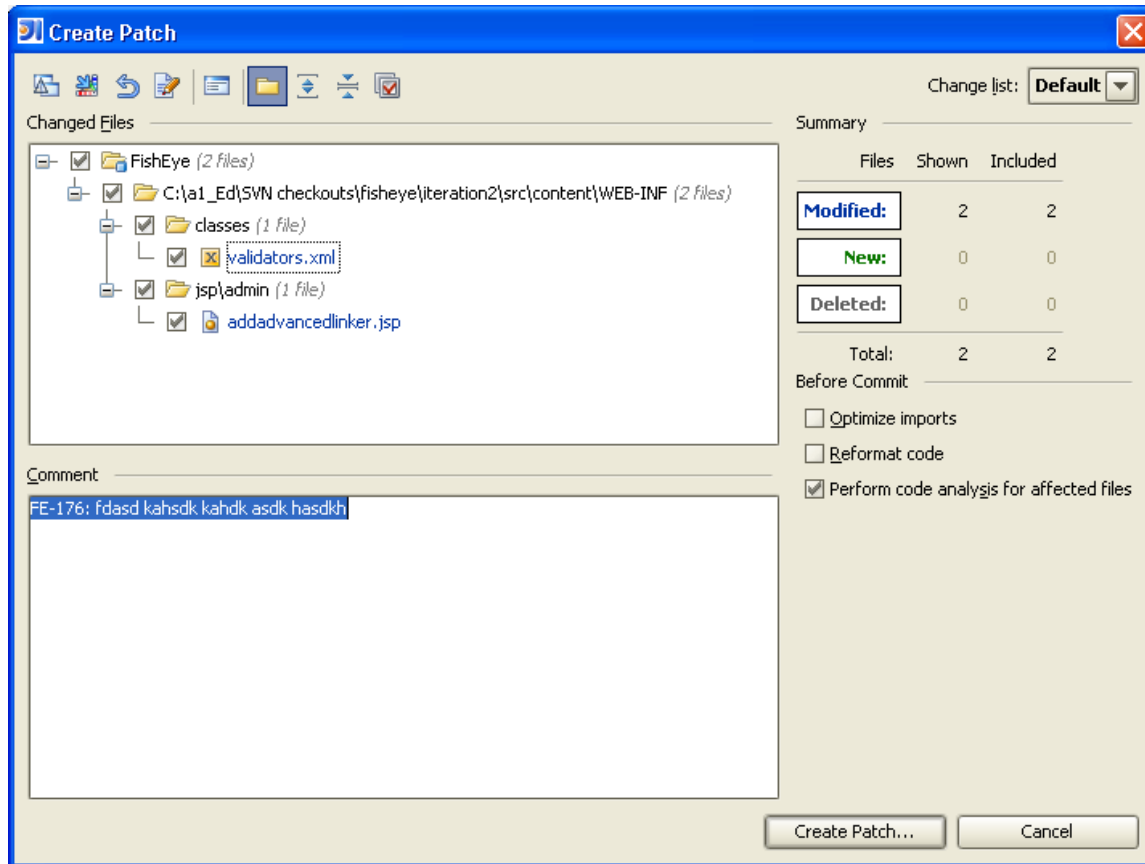
Creating a Patch File From Your IDE

Creating a Patch File in IntelliJ IDEA 7.0

To create a Patch File under IntelliJ IDEA, do the following:

Select a parent folder, sub-folder or file that you have altered in the Project tool window. Select **'Version Control' > 'Create Patch'**. The following window appears:

Screenshot: The IDEA Create Patch window



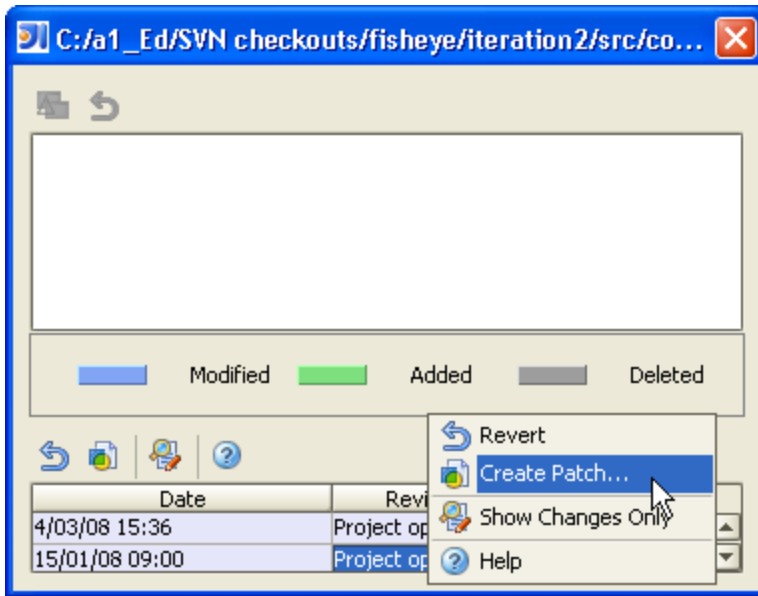
Click **'Create Patch'**. Choose a location to save the patch file and click **'Ok'**.

If You Do Not Have the Create Patch Command Available Under IDEA

If you have not configured version control in IDEA, you may not have the **'Create Patch'** option available. If so, use the following steps to create a patch file in IDEA:

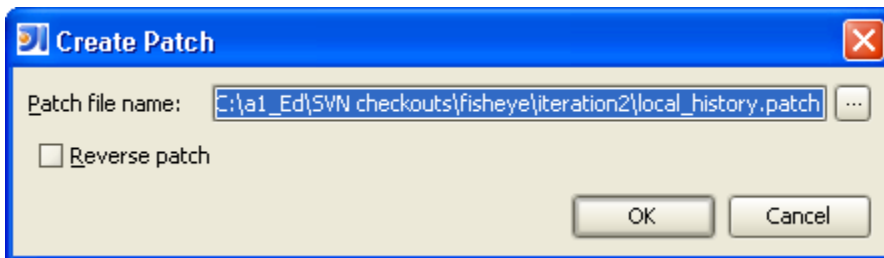
1. Select a parent folder, sub-folder or file that you have altered in the Project tool window, right-click it and choose **'Local History' > 'Show History'**.

Screenshot: The IDEA Show History dialog



2. In the Local History view, right-click the revision number, and choose '**Create Patch**'.

Screenshot: The IDEA Create Patch dialog



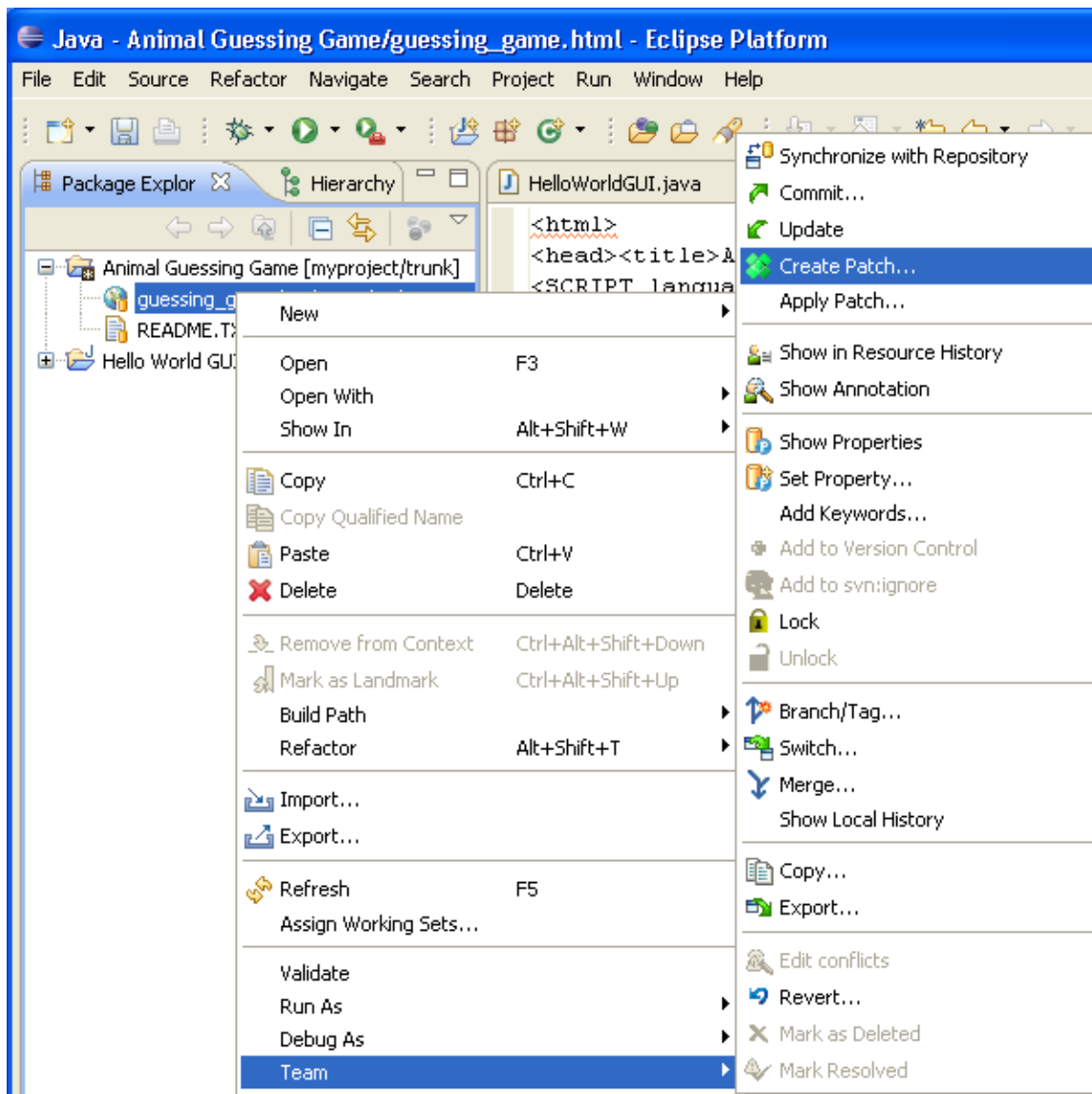
3. In the Create Patch dialog, choose a location for the patch file and a file name, then click '**OK**'.

Creating a Patch File in Eclipse 3.3.1.1

To create a patch file under Eclipse, do the following:

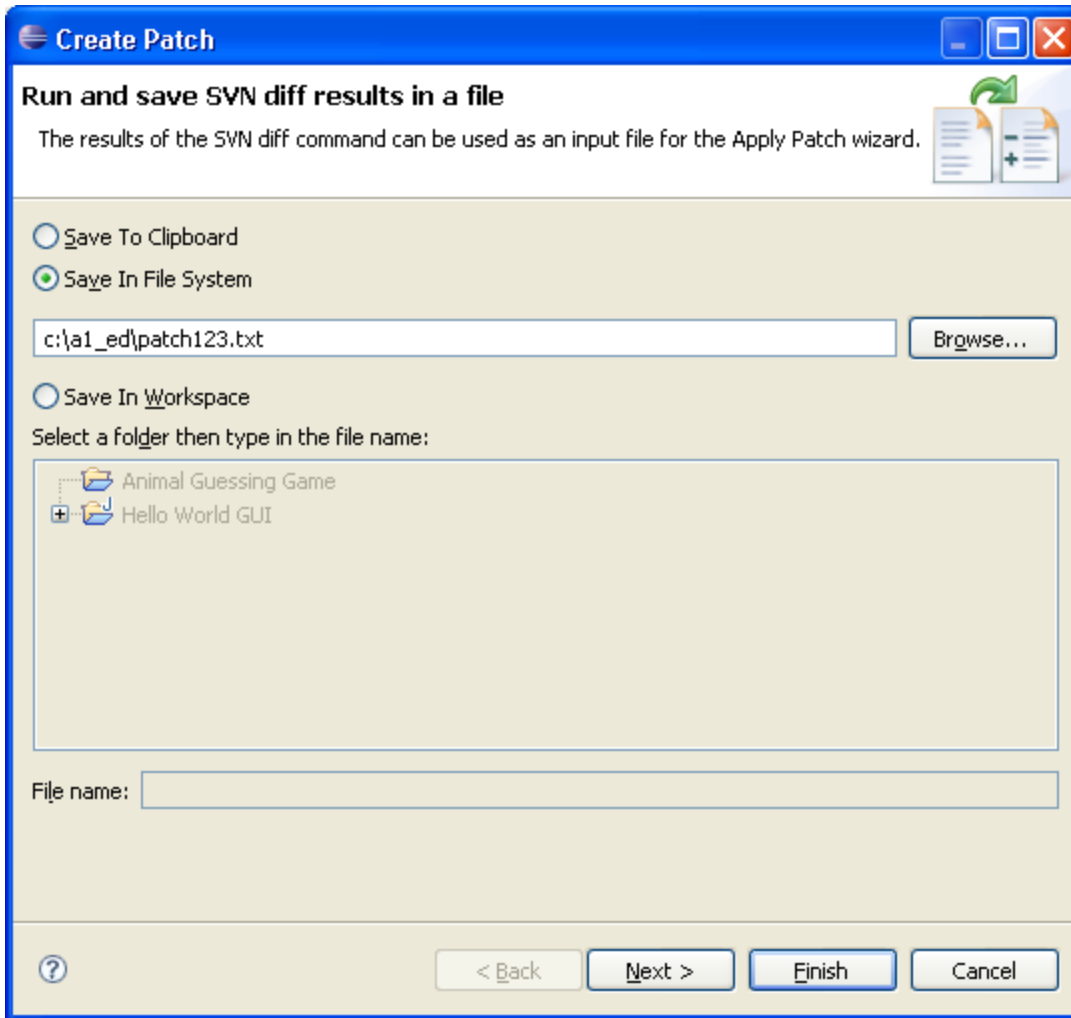
Find the parent folder, sub-folder or file that you have altered, right-click it and choose '**Team**' > '**Create Patch**'.

Screenshot: Instigating a Patch in Eclipse



In the Create Patch window, choose a location on your computer and type an appropriate file name (the file format is plain text).

Screenshot: The Eclipse Create Patch dialog



Creating a Basic Patch File From The Repository Command Line

CVS Patch Creation On The Command Line

To create a patch in CVS, use the `cv diff -Nu` command from your workspace. For example:

```
cv diff -Nu > patch.txt
```

Note that patch files created with this command will only include around three lines of code, before and after each change.

Subversion Patch Creation Via The Command Line

To create a patch in Subversion, use the `svn diff` command from your workspace. For example:

```
svn diff > patch.txt
```

 `svn diff` does not print any information about files copied in the workspace.

Note that patch files created with this command will only include around three lines of code, before and after each change.

Perforce Patch Creation Via The Command Line

To create a patch in Perforce, you must ensure you have set `P4DIFF` to point to a GNU-compatible diff program.

Then use `p4 diff -du` to generate a patch for changed files. For example:

```
p4 diff -du > patch.txt
```

Since Perforce diffs do not include added and deleted files so you should then do a `p4 opened` to find such files. For added files, append them individually to the patch using GNU diff:

```
diff -u path_to_added_file /dev/null >> patch.txt
```

In the example above, replace `path_to_added_file` with the actual path of your added file. You can follow a similar procedure with deleted files using `p4 print` to extract the previous version of the file.

Note that patch files created with this command will only include around three lines of code, before and after each change.


Creating Patches That Include All Lines of Code

To create a patch file that shows all lines of code as well as the changes, use the following commands from your repository command-line tools.

Creating a Patch in CVS With All Lines of code

To create a CVS patch that shows all code (not just the changes and surrounding code), use this command:

```
cvs diff -N -U 10000 > patch.txt
```

 The '10000' number refers to the number of code lines included in the patch, before and after each change. '**Patch.txt**' represents your desired name for the new patch file.


Creating a Patch in Subversion With All Lines of Code

To create a patch in Subversion that shows all code (not just the changes and surrounding code), use this command:

```
svn diff --diff-cmd diff -x "-U 10000" > patch.txt
```

- The in-built diff feature in `svn diff` does not support specifying lines of context, so you must tell Subversion to use an external diff command.
- The second "diff" in the command above needs to be the name of your external diff command. You might need to specify the full path to that command, such as `/usr/bin/diff`.
- On the Windows platform, you may need a Unix-like emulator such as [Cygwin](#), and install the optional diff command for that.

Creating a Patch in Perforce With All Lines of Code

 Unfortunately, Perforce does not directly support creating patches that include all lines of code. A workaround is to checkout 'before' and 'after' versions of the file, and use GNU Diff to create a patch between the two files. That file could then be loaded into a Crucible review.

 A future version of Crucible will include helper tools to assist in this process.

Selecting the Files for the Review

This page explains how to select files that will be included in a Crucible review.

The **Manage Files** dialog allows you to select and modify which files make up the review. In the Manage Files dialog, the author selects the source files they want to include in the review, by clicking the checkboxes next to the desired files. Once finished, the author clicks **Done**.

On this page:

- [Overview of the Manage Files Dialog](#)
 - [Left Navigation Bar Links In The Manage Files Dialog](#)
- [Selecting Changesets for Review](#)
 - [Options for Adding Changesets](#)
 - [Understanding the Icons Shown with Changesets](#)
- [Selecting Files for Review](#)
- [Searching in the Manage Files View](#)
- [Viewing the Content Selected for Review](#)
- [Using the Suggestions Feature When Adding Files to a Review](#)
- [Adding Patch Files to a Review](#)

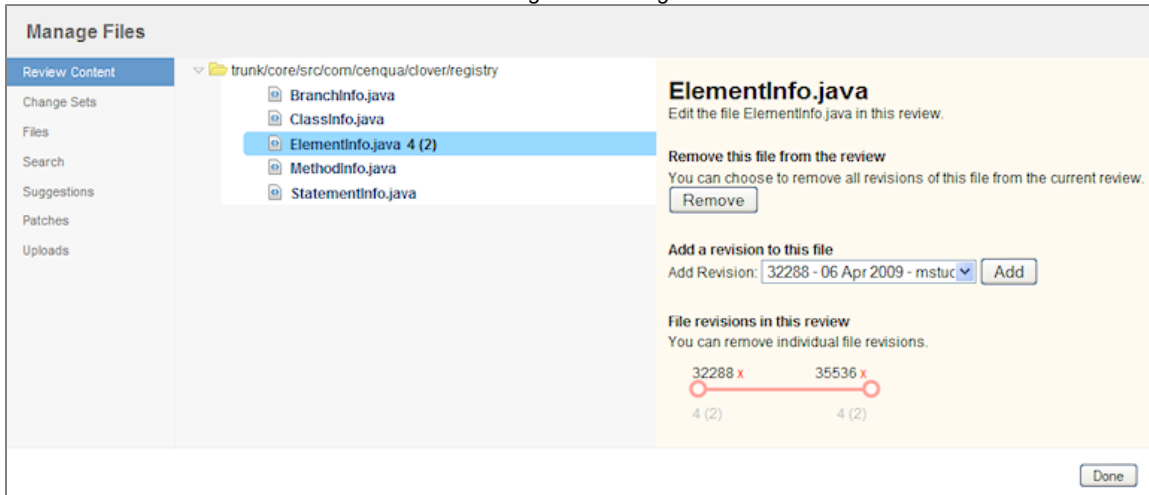
- [Uploading Files to a Review](#)
- [Choosing the Form of the Review](#)

Overview of the Manage Files Dialog

The **'Manage Files'** dialog is first displayed when you are [creating a review](#). Once a review has started, (as the [creator](#) or [moderator](#)) you can reopen the dialog by clicking the **Manage Files** button on the Review toolbar.

The left navigation bar allows you to select and locate different files for the review including files from repositories, patch files and uploaded files. The screenshot below shows the view shown when checking the list of currently selected files; the **'Review Content'** view.


Screenshot: The Review Content View Under the Manage Files Dialog



See the table below for detailed instructions on the various links on the left navigation bar of the Manage Files dialog.

Left Navigation Bar Links In The Manage Files Dialog

Left Nav Title	Explanation	Special Conditions
Review Content	Shows the list of files selected for review and allows you to manage them.	
Change Sets	Allows you to choose change sets from a Source Code Management (SCM) repository and add them to a review.	Only appears when FishEye is installed.
Files	Allows you to choose files from a Source Code Management (SCM) repository and add them to a review.	
Search	Allows you to search a Source Code Management (SCM) repository for files or change sets and add them to a review.	Only appears when FishEye is installed.
Suggestions	Analyses the list of files in the current review and makes suggestions based on certain logic (for example, suggesting a newer version of a file if one exists).	
Patches	Allows you to upload patch files to a review.	
Uploads	Allows you to upload any file to a review, including binary files and files outside of a Source Code Management (SCM) repository.	

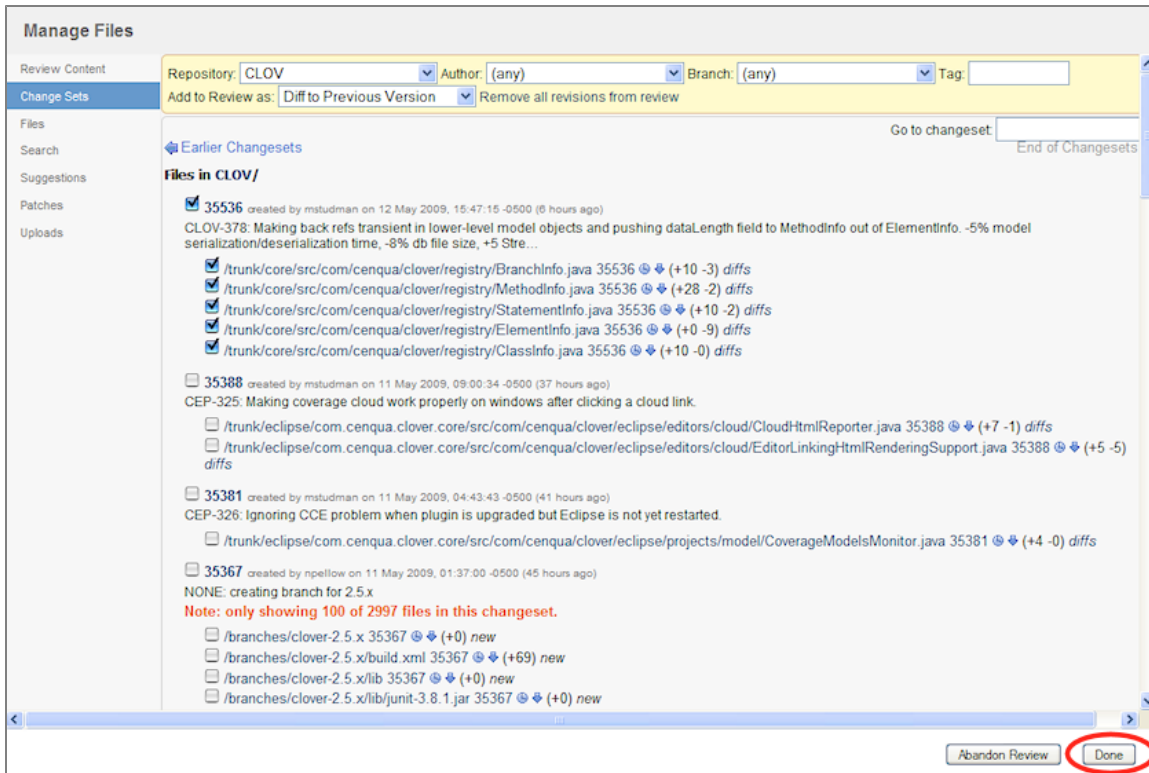
 When a red dot is shown next to a file name or changeset name, it indicates that for the file you have selected, a newer version exists. When a number is shown next to the file or changeset name, it indicates the number of comments that have been made. Numbers in brackets show unread comments.



Selecting Changesets for Review

To add changesets to a review, open the **'Changesets'** view by clicking the "Changesets" option on the left navigation bar in the Manage Files dialog. The Changesets view opens.

Screenshot: The Changesets View in the Manage Files Dialog



By default, Crucible presents a list of the author's changesets in reverse chronological order. You can see other changesets by changing the options at the top of the screen.

Click the checkbox next to a changeset ID to add the entire changeset, or the checkboxes next to file names to add or remove individual files. Clicking the checkbox of a parent item will select or de-select all of its children. Click **'Remove all revisions from review'** to remove all.

Options for Adding Changesets

Repository	This is a list of the repositories that contain the files that can be reviewed. If the repository you require is not in the list then it has not been added to FishEye. Please contact your Crucible/FishEye administrator.
Author	This contains a list of all the authors who have made changes within the repository. When creating a review, this will default if possible to the username of the user authoring this review and will therefore show their changesets.
Branch	This will only show files and recent changes on that branch from the repository set above.
Tag	This will only show files and recent changes tagged.
Add to Review As	Choose the form of the review; see full details .
Go to Changeset	Allows you to jump to a particular change set by entering its title and pressing Enter.

Understanding the Icons Shown with Changesets

To help decide what files are to be placed under review, you can click the icons next to the files to gain further information about them before they go out for review:

Clock icon or the file URL	A view detailing the history of this particular file.
Changeset ID next to the file URL	A view of the complete file. Amended lines are highlighted on the left in yellow.
Down arrow icon	Option to download the file.
Change Indicator (+n -n)	This shows how many lines have been amended (e.g. +3 -2) and also what type of change has been made. If it says 'diffs' then you can click this to see the differences in the file between the revisions.

After you have chosen the files, click the **'Review Content'** option on the left navigation bar to view the files that are included in the review.

Selecting Files for Review

To add files to a review, click the **'Files'** option on the left navigation bar of the Manage Files dialog. The Files view opens.

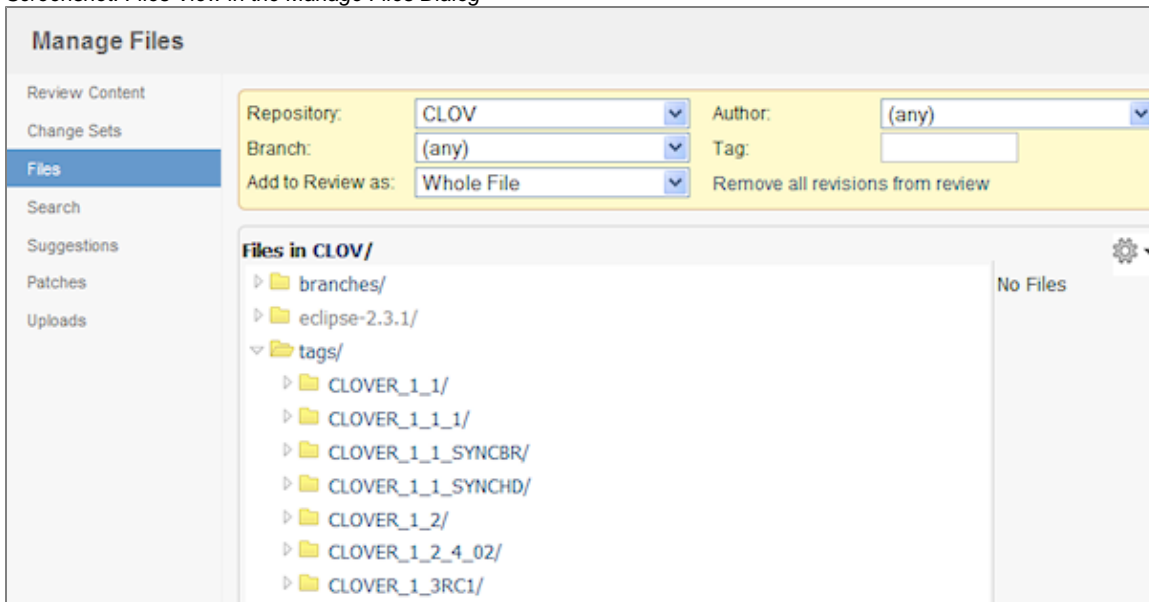
To find a file, browse the folders by clicking the relevant folder. The folders by default are sorted by path name but can be sorted by last-commit or first-commit. To choose a file for reviewing, click the checkbox to the left of the filename and if required the revision number.

Caveats:


- Empty folders will be greyed out.
- If the folders contain empty folders then a toggle option called **'Hide Empty'** will appear under the 'Sort' options.
- To see or ignore **deleted** files, you can click the **'Hide'** and **'Show'** options located above the file names on the left.

After you have chosen the files, click the **'Review Content'** view to see the files that are included in the review.

Screenshot: Files View in the Manage Files Dialog



Searching in the Manage Files View

 The 'Search' view is available when using FishEye with Crucible.

Screenshot: The Search View in the Manage Files Dialog

If you are not certain about which changesets/revisions/files to include in a review, use the Search view to find them.

Adjust the search filters to find the files you need. If the simple filters are not enough, read about [EyeQL queries](#) in the FishEye documentation.

After you have chosen the files, click the **'Review Content'** view to see the files that are included in the review.

Read the FishEye documentation for more information about the [searching your repository](#).

Viewing the Content Selected for Review

To look at the list of content selected for the review, click the **'Review Content'** option on the left navigation bar. The Review Content view opens.

In this view, you can look over the list of items selected for review. For each item, you can click its name to see which revisions are included in the review. In that view, you can also remove individual files or content items completely (by clicking the small red 'X' icons over each item), as well as adding or removing individual revisions.

Note that you cannot removed items when they already have comments added to them.

Screenshot: The Review Content View in the Manage Files Dialog

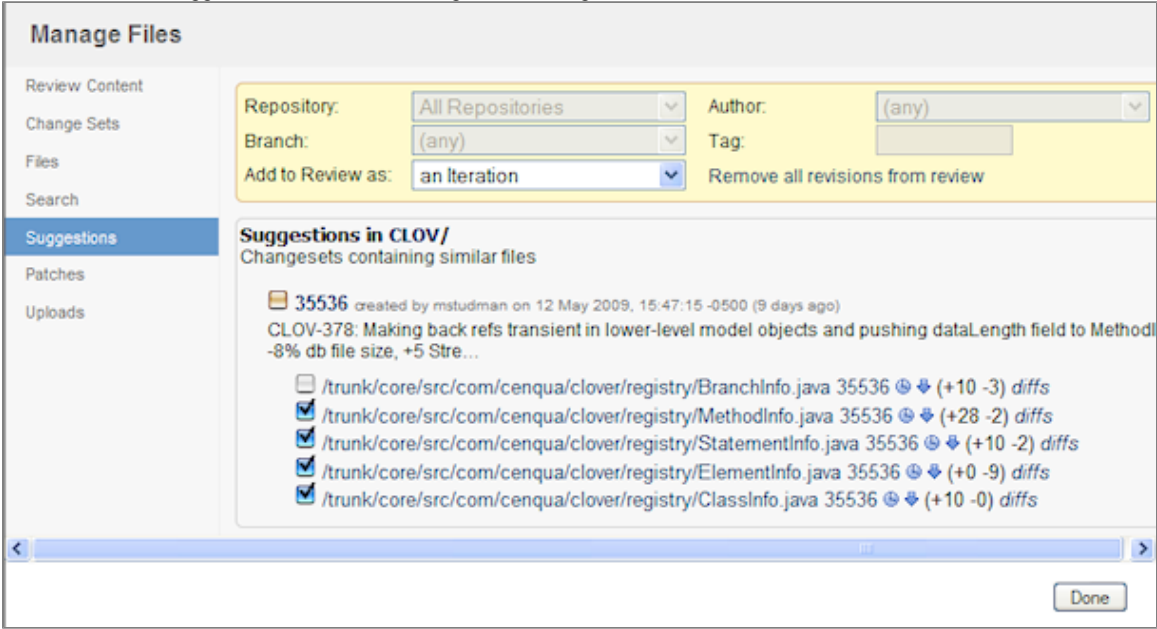
Using the Suggestions Feature When Adding Files to a Review

Crucible can make intelligent suggestions when you are creating a review. Once you have selected some files, click **'Suggestions'** in the left navigation bar to let Crucible analyse your selection and make some recommendations to you.

The Suggestions feature logic is based around the following:

- Most recent versions: If a newer version of a file exists, Crucible will suggest that you add it to the review.
- Similar files: Files with a similar filename may be of relevance to your review; Crucible will show them to you.

Screenshot: The Suggestions View in the Manage Files Dialog



Adding Patch Files to a Review

For a full explanation of the **'Patch View'** functions, read about [creating a patch review](#).

Uploading Files to a Review

You can upload additional files to be used in the review, including binary files, images or code files that are not stored in a version control repository.

The **'Upload'** view contains various controls to help you do this. These are listed in the table below.

File to review	By clicking 'Browse' , you can find the file you want to review.
File to diff with	By clicking 'Browse' , you can find the file you want to compare against the first file.
Comment	Here, you can put in an identifying comment about this particular piece of work.
Author username	Enter the author username into this field.
Character Set (if any)	In the drop-down list, you can choose the character set being used. 'US-ASCII' is the default.

When you've made your selection, simply click the **'Upload'** button. Once complete, a list of uploaded files is displayed at the bottom of the screen.

Screenshot: Upload View in the Manage Files Dialog

Choosing the Form of the Review

When adding files to a review, you can set the form of review taking place. Several options are listed along the top of the 'Manage Files' dialog, in the **Add to Review as:** drop-down menu:

Screenshot: File Addition Options in the Manage Files Dialog

Whole File	Adds the entire file with all content, rather than just a diff with context.
Diff to Previous Version	This is the default behaviour, and how previous versions of Crucible worked by default.
An Iteration	This allows you to add multiple revisions of a file to one review and compare them in-review, in context with the change history.
Diff to Last Reviewed Version	This creates a review with a diff to the last reviewed changeset.
Diff to... (a particular revision)	This allows you to specify the review to show the differences between two specific versions of a file. This is useful in various situations. For example, you may want to review all changes since you branched your project.
Whole file	The review content is not limited by changesets, showing the entire file contents.

Adding Reviewers

This page explains how to add reviewers to a new review, after it has been created. See [Creating a Review](#) for information about creating reviews.

On this page:

- [Entering Basic Information](#)
- [Adding Reviewers](#)
 - [Adding Users to a Review](#)
 - [Inviting Non-Registered Users to the Review](#)
 - [Checking the Draft and Starting the Review](#)
- [Next Steps](#)

Entering Basic Information

Once a review has been created, the Edit Review dialog opens.

Screenshot: The Edit Review dialog

Edit Review

Title:

Project: Moderator: Author:

Reviewers: ☐ Allow anyone to join

☒ Geoff Crain

Start typing the reviewer's name then press enter to select.

User	Most Contribution (%)	Total LoCOpen Reviews
brendan(brendan)	ClassInfo.java (84%)	526 0
Nick Pellow(npellow)	ClassInfo.java (1%)	3 0
Michael Studman(mstudman)	BranchInfo.java (41%)	64 3
Brendan Humphreys(bhumphreys)	MethodInfo.java (1%)	3 1

Due date:

Link to Jira Issue:

Issue key:

Statement of Objectives:

In the Edit Review dialog, the author enters information needed for the review. This includes entering a title and description for the review, a due date and the key for a related JIRA issue (if any). The project, moderator and author are pre-selected (for this example, the author should select himself as a moderator).

You must also select reviewers.

Adding Reviewers

Before a review can be issued to reviewers, you must decide who can review it. When adding reviewers, you can add registered users immediately. The usernames will auto-complete, showing partial matches before you finish typing. You can quickly select one of the matches shown with the keyboard arrow keys, pressing Enter or Tab to add them to the review.

In addition, you can easily invite external users who do not yet have accounts in Crucible to take part by typing their email address into the **Reviewers** field.

Adding Users to a Review

Select users by typing names into the text field under **Reviewers**. Crucible will show a list of matches. Press Enter to select one after each entry.

Crucible will suggest users that have contributed to the files you've selected and also don't have a lot of open reviews.

Clicking the 'Save' button will save the review as a draft for later issue.

You can also decide to allow any registered user to add themselves as a reviewer in the review. To enable this option, put a check next to 'Allow anyone to join'.

Inviting Non-Registered Users to the Review

You can invite users who don't have a Crucible account to join a review.


There are two prerequisites:

1. FishEye's SMTP server must be configured and capable of sending email.
2. The setting **Built-in Public Sign-up** must be set to **ON**. This setting can be accessed by opening the **Admin Menu**, then clicking '

Security' under **'Global Settings'** on the left navigation bar.

To invite an external user to a review,

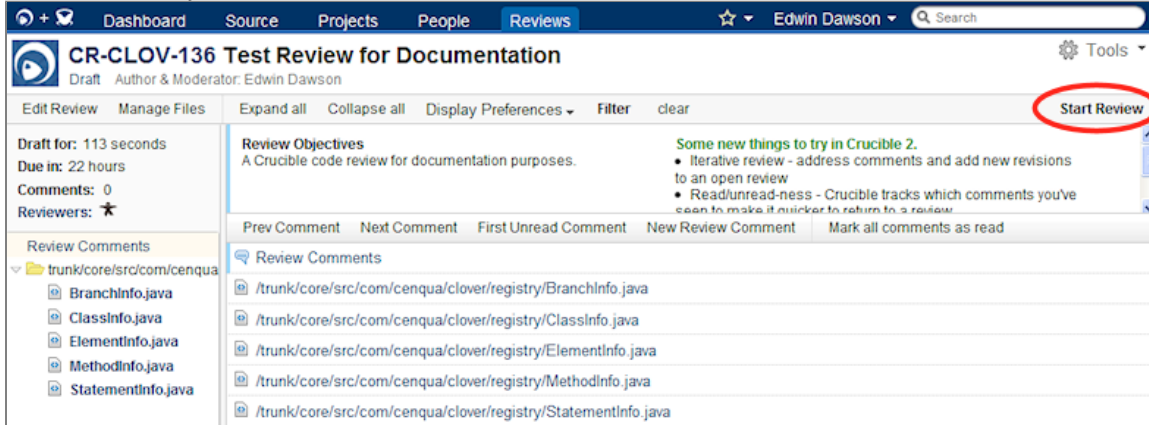
- a. i. 1. Create a new review.
2. On the **'Create New Review'** screen, simply type the user's email address into the **'Reviewers'** field, then press Enter to select.
3. Click Save to save the draft review. The users are not sent any information at this time.
4. When you click **'Start Review'**, this is when all email invites and notifications are sent out.
5. The external user will receive an email address from the Crucible server, containing a special URL that they can visit.
6. When the user loads the URL they received via email, they are taken to a special Crucible log in screen. On this screen, the user can create a new account that will be linked to the current email address. (If they already have a Crucible account under another address, they can simply sign-in with that username and password.)
7. When the user has successfully created a Crucible account, they will be able to access the review(s) associated with their email address and take part.

 You can enter multiple addresses separated by commas, allowing you to paste in a list of email addresses from your favourite email application.

When finished, the author clicks **'Save'**. The review will now be created in a draft form.

Checking the Draft and Starting the Review

Screenshot: A newly created Crucible review



The draft review opens. In the draft stage, the author can check the contents of the review files to ensure they are correct and put in any notes for reviewers as comments. During the draft phase, no notification email is sent out to reviewers. Once the author is finished with the draft phase, he clicks **'Start Review'**.

The review will now be started and notification email will go out to all participants. Crucible will now send out an email notification to all the participants. This lets them know that the review is under way and prompts them to take action, providing a URL for direct access to the review. (You can also [subscribe to an RSS feed](#).)

Next Steps

You can now begin [Performing the Review](#).

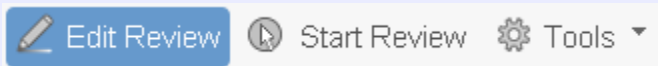
If you have a moderator controlling your review process, you can move onto [Issuing a Review](#).

Removing Reviewers From An Active Review

Reviewers can be removed from an active review at any time.

To remove a reviewer from an active review,

1. Log in to Crucible as the review creator or moderator.
2. Open the review in question.
3. Click the **'Edit Review'** button:



4. Find the user you want to remove and click the checkbox next to their name to remove them (so that the checkbox becomes empty):


Select Reviewers:


Start typing the reviewer's name then press enter to select.

☒ Geoff Crain

☐ Allow anyone to join

5. Click **'Save'**.
6. The user will be removed from the review and notified by email.

 Reviewers can be only removed by the review creator or moderator.

 You cannot remove the review creator or moderator from the review.

Issuing a Review

Issuing a review in Crucible is known as **approving the review**. This simply means formally starting the review and inviting people to take part.

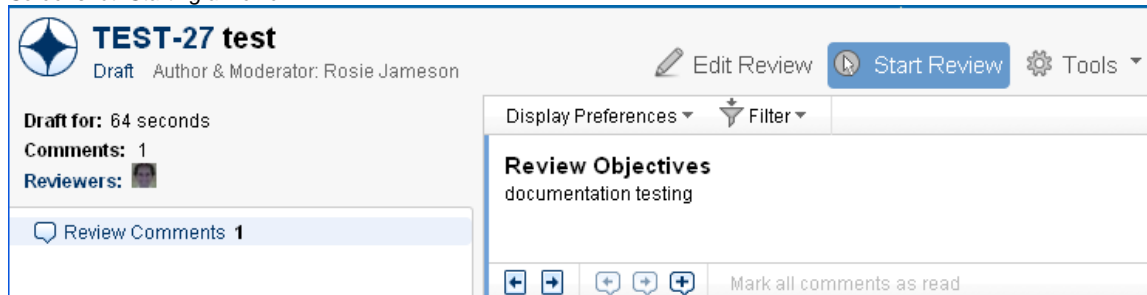
Once you have [selected the reviewers](#), the next stage is to notify the [reviewers](#) and the [author](#) (if different to the [moderator](#)) that they can start reviewing. The review has been in 'Draft' [state](#) until this point. Only the moderator has the permission to approve a review.

To issue the review:

- If you are the moderator of the review, click the **'Start Review'** button. Or;
- If you are not the moderator of your review, click **'Send to Moderator'**. This changes the [state](#) to 'Require Approval' and notifies the moderator. The moderator can change any aspect of the review before approving it.

Once the review has been approved, the review [state](#) becomes **'Under Review'**.

Screenshot: Starting a Review



 Note that only people with the **'Approve'** [permission](#) can issue a review.

Performing the Review

This page describes how to find and manage the Crucible reviews that relate to you.

On this page:

- [Browse Your Reviews Under the 'Dashboard' Tab](#)
- [Browse All Reviews Under the 'Reviews' Tab](#)
- [When files change during a review](#)
- [Next steps](#)



Deciding what needs to be reviewed

Crucible does not dictate how or what to review. It simply provides a mechanism to record comments. The **'Statement of Objective'** is a brief description of what the review is intended to achieve.

Browse Your Reviews Under the 'Dashboard' Tab

When you first load Crucible, the **'Dashboard'** screen will load, which shows your current reviews and other activity related to you.

Use the Crucible **'Dashboard'** to manage your reviews. Read the overview on [filtering your view](#).

Active reviews are listed on each [reviewer's](#) dashboard under the default **'To Review'** filter. Reviews are listed under **'Out for Review'** until all reviewers indicate they are complete. Then the reviews move to the **'To Summarize'** list.

Read more about using the [Dashboard tab](#).

Browse All Reviews Under the 'Reviews' Tab

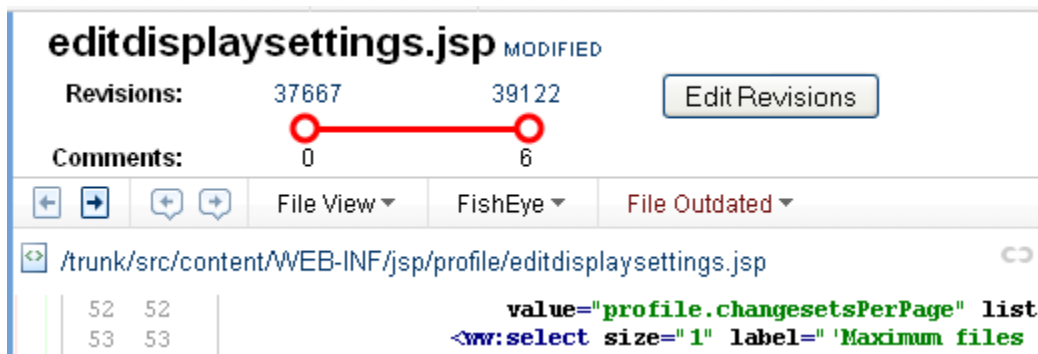
All reviews that involve you in any role are listed when you click **'Open'** or **'Closed'** in the left navigation bar. For instance, use the 'My Reviews' --> 'Open' filter to locate a review that doesn't require further action from you, but is still under way.

If email notifications are enabled (see [SMTP settings](#) in the FishEye documentation), reviewers will receive an email with information about the review. Click the link within the email to go directly to the review.

When files change during a review

If a file in the repository changes during a review, Crucible will visually alert you by showing the **'File Outdated'** menu:

Screenshot: Visual Cue for Updated Repository Files



From the **'File Outdated'** menu, you can choose to view the latest revision of the updated file, or add the latest revision to the review:

Screenshot: 'File Outdated' menu

editdisplaysettings.jsp MODIFIED

Revisions: 37667 39122 [Edit Revisions](#)

Comments: 0 6

File View ▾ FishEye ▾ File Outdated ▾

View Latest
Add Latest

```

52 52 value="
53 53 <vw:select list="#@java.util.LinkedHashMap@{"Y": "Ye
54 54 value="profile.maxFilesPerChangeset" list=
55 - <vw:radio label=" 'Always Expand Changesets in
55 + <vw:radio label=" 'Always expand changesets in
56 56 name=" '@@alwaysExpandChangesets' " value="p
57 57 list="#@java.util.LinkedHashMap@{"Y": "Ye
58 - <vw:radio label=" 'Show My Activity On Home Pa
58 + <vw:radio label=" 'Show my activity on home pa

```

Next steps

- Adding Comments
- Flagging Defects
- Completing your Review
- Sending all of a Review's Comments via Email

Adding Comments

Comments can be added at the level of a review, revision, or line. You can also reply to a comment.

On this page:

- Locating existing comments
- Adding a Comment
- Draft Comments

Locating existing comments

The number shown next to a filename, in the left-hand column of the screen, indicates the number of comments that apply to that file.

(The number of unread comments, if there are any, is shown in brackets.)

Screenshot: Comments

CR-CLOV-136 Test Review for Documentation
 Under Review Author & Moderator: Edwin Dawson [Edit Review](#) [Summarize](#) [Tools](#)

Under Review for: 45 days
Overdue: 52 days
Comments: 4
Reviewers:

[Review Comments](#)
trunk
 core/src/com/cenqua/clover/

- BranchInfo.java
- ClassInfo.java
- ElementInfo.java 4
- MethodInfo.java
- StatementInfo.java

 idea7/src/com/cenqua/clover/

- IdeaTestFilter.java

 src/tutorial

- build.xml

ElementInfo.java MODIFIED
 Revisions: 32288 35536 36696 [Edit Revisions](#)
 Comments: 4 4 0
 File View FishEye File Outdated

```

/trunk/core/src/com/cenqua/clover/registry/ElementInfo.java
4 4  import com.cenqua.clover.context.ContextSet;
5 5  import com.cenqua.clover.context.NamedContext;
6 6
7 -  public abstract class ElementInfo extends SourceRegion implements
8 -  private ContextSet context;
9 -  private int relativeDataIndex;
  
```

Edwin Dawson says: 13 May
 What is this relative to?
[Reply](#) [Edit](#) [Delete](#) [Add Favourite](#)

Geoff Crain says: 13 May
 the speed of light
[Leave Unread](#)

```

10 -  private int dataLength = 1; // by default an element occurs
11 -  private int complexity;
  
```

Edwin Dawson says: 13 May
 Is this variable name sensible?
[Reply](#) [Edit](#) [Delete](#) [Add Favourite](#)

Geoff Crain says: 13 May
 I think so - may not be complex enough, though
[Leave Unread](#)

```

7 +  import java.io.Serializable;
12 8
13 -  public ElementInfo(
14 -  BaseFileInfo containingFile, int relativeDataIndex, Co
15 -  SourceRegion region, int complexity) {
16 -  super(containingFile, region);
17 -  this.relativeDataIndex = relativeDataIndex;
9 +  public abstract class ElementInfo<T extends SharedElementInfo>
10 +  protected BaseFileInfo containingFile;
11 +  protected ContextSet context;
  
```

Adding a Comment

- To add a comment that applies to the whole review, select the **'Review Comments'** line in the left-hand navigation panel, then click the following icon: Unable to render embedded object: File (icon_add_comment.png) not found.
- To add a comment that applies to a revision/change, select the filename in the left-hand navigation panel, then click the following icon: Unable to render embedded object: File (icon_add_comment.png) not found.
- To add a source-level comment, expand the source view then click a line of code. You can click and drag to select multiple lines from one revision or diff, or click individual lines to select/deselect them. The comment will appear in the source at the last line selected. Hover over the comment to highlight the selected lines.
 - To select text on the page without adding a comment, hold down the **Alt** button while dragging the cursor.
- To reply to a comment, click the **'Reply'** link at the bottom of the comment.

Only people with the **'Comment'** permission can add comments.

Read about [flagging defects](#) too.

Screenshot: Adding a Comment

Is this variable name sensible?

☐ Defect

Draft Comments

You can save your comments as drafts and then edit them later. When you [complete the review](#), you will be prompted to post/discard/edit any remaining draft comments.

Screenshot: Draft comments



Edwin Dawson says:
Is this variable name sensible?


Flagging Defects

Any [comment](#) that is not a reply to an existing comment can be flagged as a **defect** by its author.

Screenshot: Defects

Is this variable name sensible?

☒ Defect



Geoff Crain says:
i think so - may not be complex enough, though

```

7 + import java.io.Serializable;
12 8
13 - public ElementInfo{
14 -     BaseFileInfo containingFile, int re
15 -     SourceRegion region, int complexity,

```

Select Classification

Missing
Extra (superfluous)
Ambiguous
Inconsistent
Improvement desirable
Not conforming to standards
Risk-prone
Factually incorrect
Not implementable
Editorial

13 May

context,

You may want to mark comments as defects to associate defect classifications, or simply to highlight to the [author](#) or [moderator](#) that the issue you raised in your comment requires attention.



Crucible intentionally does not mandate how defects are to be used. The Crucible administrator can [customise the defect classifications](#).

Completing your Review

Once each [reviewer](#) has added [comments](#) to the review and has nothing further to add, the next step is to **Complete** their individual review.

To complete your individual review, go to the review and click the '**Complete**' button at the right of the screen, next to the '**Tools**' menu:

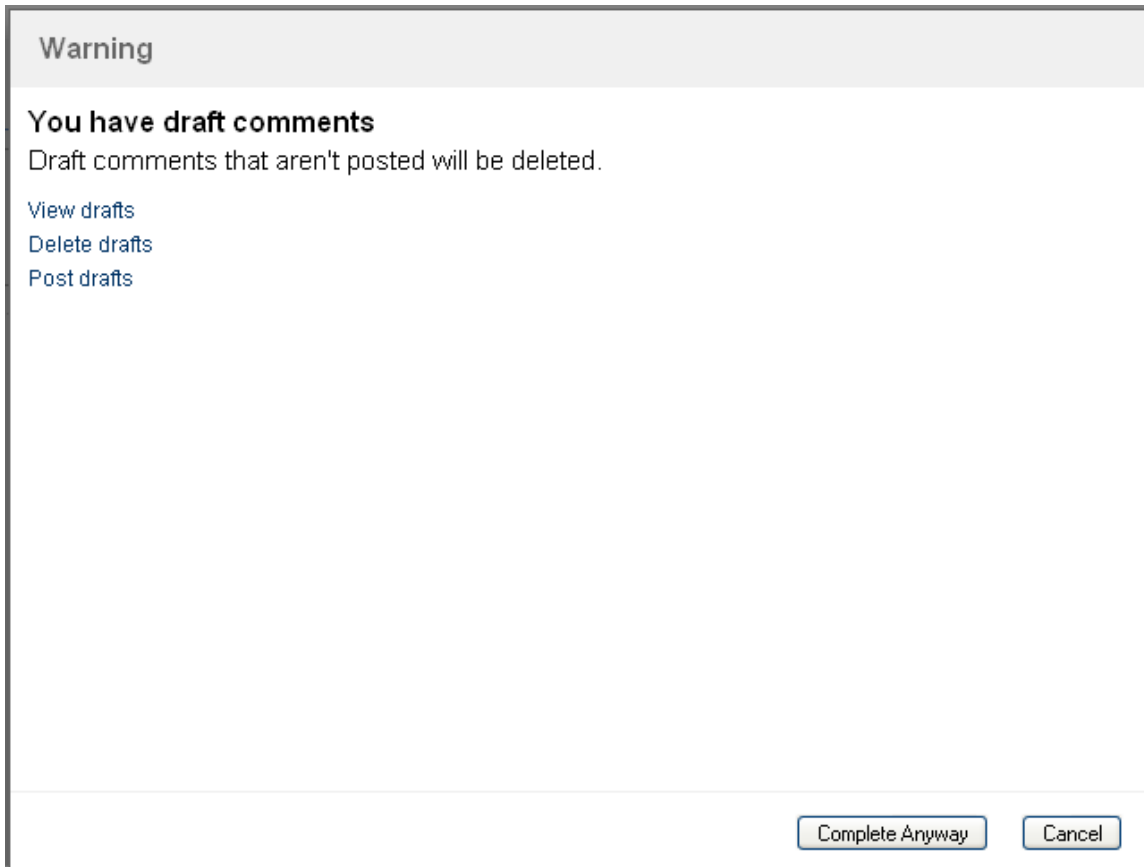
 Only people with the '[Complete](#)' permission can complete a review.

This notifies the [moderator](#) (via email if configured) that you have completed your review.

Reviewers can still continue to add comments until the moderator [summarises](#) the review. The moderator does **not** have to wait for all reviewers to complete their individual reviews before summarising.

If you have any draft [comments](#), you will be prompted to post/discard/edit any comments before completing the review.

[Screenshot: Draft comments](#)



[Screenshot: Review complete](#)

Review Complete



You have marked TEST-27 as completed.

The following reviewer is not yet finished:

 **Geoff Crain**

You can continue to add new comments and reply to comments until the moderator (✖ **Rosie Jameson**) summarizes the review.

Reviews requiring your attention:

Task	Review	Owner	Due
Review	CR-FE-2170		03 Jul
Summarize	TEST-11		-

Dashboard

Close


Sending all of a Review's Comments via Email

You can send all of the comments from a review to anyone you want via email. You may wish to do this to allow a person outside the review to quickly scan the content of the comments, or oversee the review activity. Alternatively, you may wish to send all participants this information to let them read the current status of the review and its comments in full.

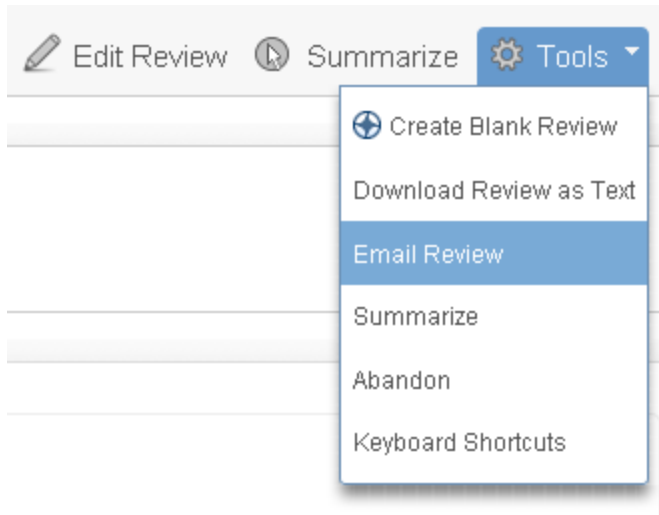
Sending a Review's Comments via Email

To send all of a review's comments via email,

1. In Crucible, navigate to the review in question.
2. From the **'Tools'** menu, select **'Email Review'** (see *Screenshot 1* below).
3. The **'Recipients'** page appears (see *Screenshot 2* below). On that page:
4. In the **'To:'** field you can enter multiple email addresses, separated by commas.
5. In the **'Recipients:'** field, you can type usernames from your Crucible instance to add them to the distribution list. You can also simply tick the **'Send to Review Participants'** check-box to include all of the review's reviewers.
6. When you have finished the distribution list, click the **'Next'** button.
7. The **'Recipients'** page appears (see *Screenshot 3* below). On that page:
8. In the **'To:'** field you can enter multiple email addresses, separated by commas.
9. When you have finished your message, click the **'Send'** button.
10. The **'Status'** page appears (see *Screenshot 4* below), confirming that your email has been sent

 Users that are not logged in cannot send email, but instead can view the text content of the review's comments by clicking the **'View Text'** option which will appear instead of **'Email Review'**.

Screenshot 1: The 'Email Review' option in Crucible



Screenshot 2: The 'Recipients' Screen in Crucible

Email Review Comments

1 Specify recipients

2 Message details

3 Status

Recipients

Specify the people who should receive the comments of this review. You can specify a list of comma separated email addresses and/or specify one or more Crucible users.

To:

Recipients:

Start typing a user name then press enter to select.

☐ Send To Review Participants

Cancel

Next

Screenshot 3: The 'Message' Screen in Crucible

Email Review Comments

1 Specify recipients

2 Message details

3 Status

Message Details

This is the message as it will be sent. You can modify both the message content, as well as the subject of the email before sending it.

Subject: [TEST-27] test: All Comments

Message:

Subject: [TEST-27] Review: test

This is a list of all comments for TEST-27.

Review Summary: No summary

ID: TEST-27 <https://extranet.atlassian.com/crucible/cru/TEST-27>

Title: test

Statement of Objectives:
documentation testing

State: Review

Author: Rosie Jameson

Moderator: Rosie Jameson

CancelPreviousSend

Screenshot 4: The Email Confirmation Screen in Crucible

Email Review Comments

1 Specify recipients

2 Message details

3 Status


Message Sent Successfully

Mail sent successfully.

CancelPreviousClose

Summarising and Closing the Review

As the [moderator](#), you can choose to **summarise a review** at any time.

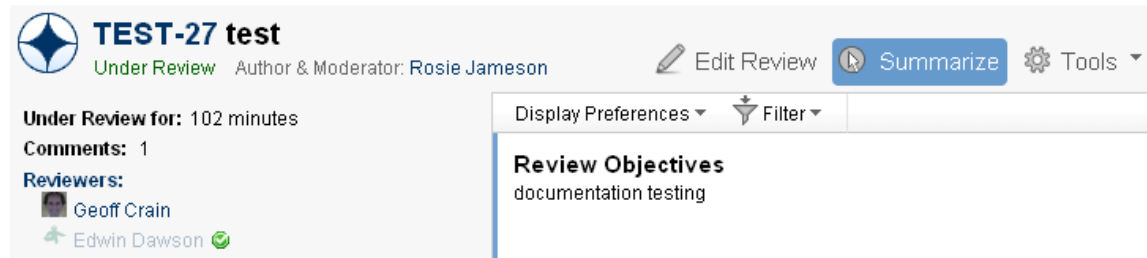
 Normally, we recommend that you wait for all reviewers to [complete their reviews](#).

The reviews that the reviewers have completed will be in your **'To Summarize'** menu on the [Dashboard](#).

To summarise a review,

- Click the **'Summarize'** button at the right of the screen.
- Optionally enter a summary of the review.
- If you have no further [comments](#) to add, click the **'Close Review'** button; otherwise, click **'Continue Without Closing'**.

Screenshot: 'Summarize' button



The above review is not yet complete

We can see that Geoff Crain has still not finished reviewing, because there is no green tick next to his name.

On clicking 'Summarize', the moderator may be prompted to confirm the action if there are incomplete reviews or draft comments in the review.



The requests for confirmation are warnings only

The review can still be summarised and closed.

Once the review is in the 'Summarize' [state](#), the moderator can optionally add a review summary, i.e. describe the outcomes/tasks/etc.

Screenshot: Summarize Review

Summarize Review

Summarize the review outcomes (optional)

Testing successful. Thanks for your time, everyone.

Continue Without Closing

Close Review


Screenshot: Review Closed

Review Closed

Review Summary

Testing successful. Thanks for your time, everyone.

Reviews requiring your attention:

Task	Review	Owner	Due
Review	CR-FE-2170		03 Jul

Dashboard

Close

The summary is sent to all participants and displayed at the top of the closed review.

The moderator is the only participant who can add comments in 'Summarize' state. This gives the moderator the responsibility of the 'last word'.

Reviews in the 'Summarize' state can be closed.

Reviews in the 'Summarize' or 'Closed' state can be re-opened. Re-opening changes the review's state back to 'Under Review', allowing all participants to add comments.



Re-opening a review is not the recommended way to 're-review'. You should create a new review with the reworked changes and link it to its parent review.

Note that you need the **'Summarise'**, **'Close'** or **'Re-Open'** permission to summarise, close or re-open a review.

Deleting an Abandoned Review

You can delete reviews that have been abandoned. To do this, follow the instructions below.



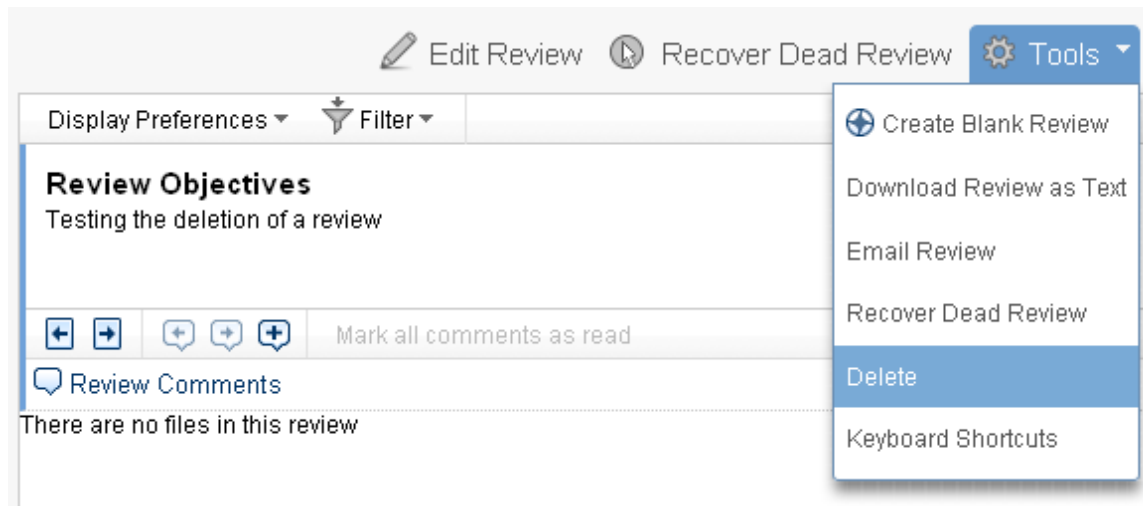
Deleted reviews cannot be retrieved.

Deleting Reviews from the 'Abandoned' list

To delete a review from the 'Abandoned' list;

1. From the **'Crucible Dashboard'**, click **'My Abandoned Reviews'** in the left-hand navigation bar.
2. A list of abandoned reviews appears. Click the name of the review you wish to remove.
3. Once the review details are displaying, select **'Delete'** from the **'Tools'** menu. The review will be instantly deleted.

Screenshot: Deleting a Review in Crucible



Moving a Review to Another Project

You can move reviews between projects once they have been created.

To move a review between projects,

1. Open the review. Click the **'Edit Review'** button at the top of the screen.
2. The **'Edit Review'** window will open, allowing you to change various aspects of the review.
3. Under **'Project'** click the drop-down menu. This will allow you to select a new parent project for the review.
4. Click the **'Done'** button at the bottom of the screen.

Screenshot: Changing a Review's Parent Project

The screenshot shows the 'Edit Review' dialog for a review titled 'TEST-29 A test review'. The 'Project' dropdown menu is highlighted with a red circle, showing 'Testing Project' as the selected option. Other fields include 'Title' (A test review), 'Moderator' (Rosie Jameson), 'Author' (Rosie Jameson), 'Select Reviewers' (Geoff Crain), 'Get Reviewer Suggestions' (Suggest Reviewers), 'Due Date', 'Linked JIRA Issue', 'Linked Review', and 'Statement of Objectives' (Test moving a review to another project.).

TEST-29 A test review
Under Review Author & Moderator: Rosie Jameson

Under Review for: 0 seconds Display Preferences Filter

Edit Review

Review Details

Title:
A test review

Project: Testing Project **Moderator:** Rosie Jameson **Author:** Rosie Jameson

Select Reviewers:
Start typing the reviewer's name then press enter to select.
☒ Geoff Crain
☐ Allow anyone to join

Get Reviewer Suggestions:
Suggest Reviewers

Due Date:

Linked JIRA Issue:
Issue key: Link

Linked Review
Review Key: Link

Statement of Objectives:
Test moving a review to another project.

Reset

Abandon Review Done

Changing your User Profile

Users can change their own account settings such as passwords, watches and display settings. These include the user avatar and profile image.

To view your user profile, log into Crucible, and click the User Menu (labelled with your username) at the top of the screen, then select '**Settings**'.

Screenshot: Settings

Settings

Display Settings

Profile & Email

Author Mapping

Watches

Reviews

Display Settings

Length of tag list:

☐ Hide
☒ Medium
☐ Long

Show linecount history graph:

☒ Yes
☐ No

Show hidden directories:

☐ Yes
☒ No

Show empty directories:

☐ Yes
☒ No

File history view mode:

☒ Physical
☐ Logical

Timezone:

Default (America/Chicago)

Changelog

Changesets per page:

30

Maximum files shown in a changeset:

5

Always Expand Changesets in Streams:

☒ Yes
☐ No

Show My Activity On Home Page:

☒ Yes
☐ No

Diff view

Truncate long diffs:

☒ Yes
☐ No

Diff mode:

☒ Unified
☐ Side-by-side

Line wrapping:

☒ None
☐ Soft

Highlighting colours:

☒ Default
☐ Classic (muted)

Source view

Default annotation mode:

☒ Age
☐ Author
☐ None

Tab width:

8

IDE Connector

Enable IDE Icons:

☒ Yes
☐ No

Port Number:

51235

Close

Refer to the [FishEye documentation](#) for information about the tabs labelled 'Display settings', 'Profile and Email' and 'Watches'.

Click 'Save'

Always click the 'Save' button after making any changes.

Reviews Tab

If the SMTP server is set up, then you will receive emails when different actions occur within Crucible.

You can change the options described below, to specify the stages at which emails will be sent.

State change	Default is 'Yes'. A Crucible review moves through different states e.g: 'Draft', 'Under Review'. An email is sent when the state changes.
Comment added	Default is 'Yes'. An email is sent when a comment is added to a review.
Participant finished	Default is 'Yes'. An email is sent (to the Moderator only) when any reviewer has completed their review .
General message	Default is 'Yes'. An email is sent when a reviewer is added or removed from a review, after it has gone into the 'Under Review' state .
My actions	Default is 'No'. If set to 'Yes', an email is sent every time you perform an action on a review.

Author Mapping Tab

The 'Author mapping' tab allows you to make an association between you (as a logged-in [user](#) in Crucible) and an [author](#) in each repository.

This is only necessary if the name of the user within Crucible is different to the name within the repository. Crucible will by default check to see whether the usernames match.

Customising Your User Avatar

If your administrator has [enabled an external avatar server](#) (e.g. [Gravatar](#)), you can upload an avatar image of your choice. Note that your login name to the external server must be the email address that is specified in your User Profile.

Defining your Workflow

This document describes several forms of Crucible Workflow in detail. Depending on the size of your team, there are four different ways that a development team could use Crucible for code reviews. Choose the workflow which suits your team.

- [Lightweight Code Commenting with Crucible \(individual\)](#)
- [One-to-One Reviews \(Agile Pair\)](#)
- [One-to-Many Reviews Without a Moderator \(Agile Team\)](#)
- [Formal Group Reviews \(CMM Team\)](#)

Lightweight Code Commenting with Crucible (individual)

1. [Author](#) commits new work.
2. Author creates the review, and adds comments using the easy web interface.
3. Author summarizes and closes the review, saving the code comments in Crucible's database, which is stored outside the repository.

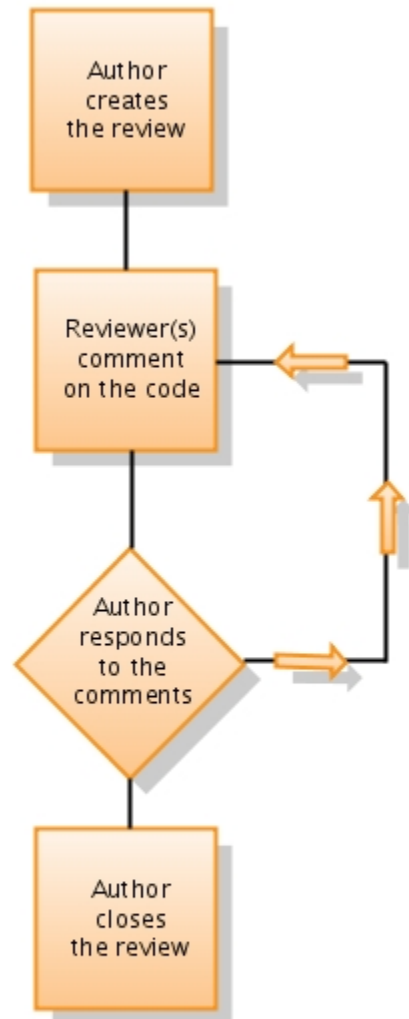
Diagram: Workflow for Lightweight Code Commenting



One-to-One Reviews (Agile Pair)

1. [Author](#) creates the review.
2. Author invites reviewer to take part in the review.
3. [Reviewer](#) creates comments on the code.
4. Author responds to reviewer comments.
5. Follow-up comments are made if necessary.
6. Reviewer finishes own review process.
7. Author summarizes and closes the review.

Diagram: Workflow for One-to-One Reviews



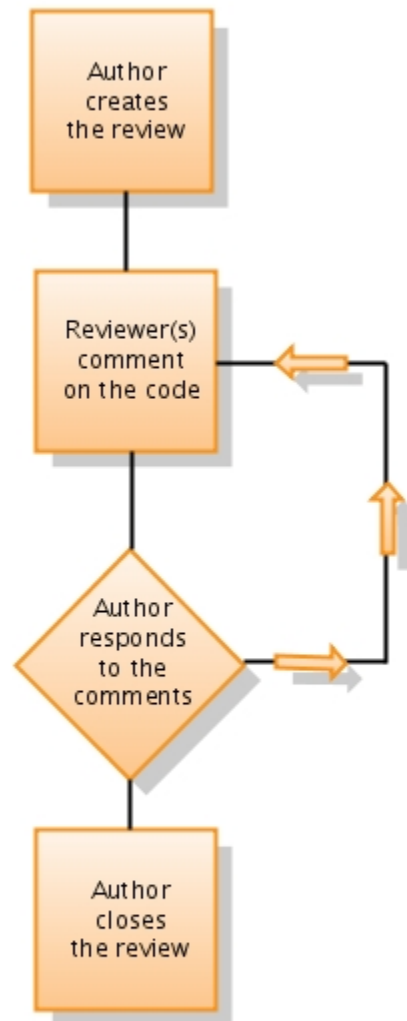
For more information on one-to-one reviews, see [Getting Started with Crucible](#). The workflow process in Crucible is covered in detail within this document.

One-to-Many Reviews Without a Moderator (Agile Team)

1. [Author](#) creates the review.
2. Author invites [reviewers](#) to take part in the review.
3. Reviewers make comments on the code.

4. Author responds to reviewer comments, follow-up comments are made if necessary.
5. Reviewers complete their reviews.
6. Author summarizes and closes the review.

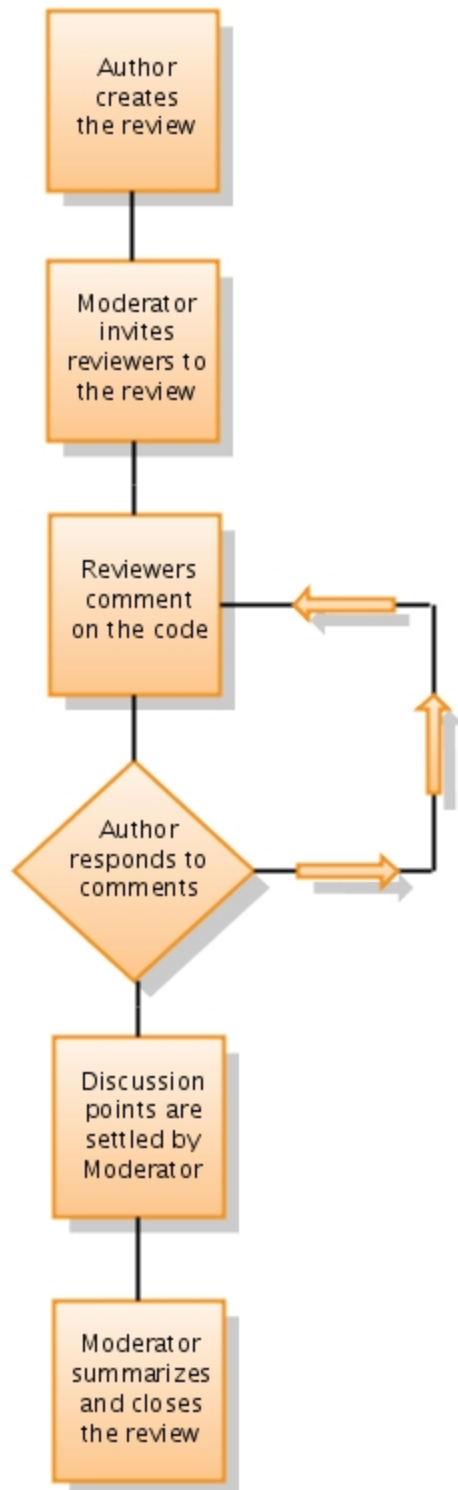
Diagram: Workflow for One-to-Many Reviews



Formal Group Reviews (CMM Team)

1. Author creates the review.
2. Moderator invites reviewers to take part the review.
3. Reviewers make comments on the code.
4. Author responds to reviewer comments.
5. Follow-up comments are made if necessary.
6. Each discussion point is settled by the Moderator.
7. Moderator summarizes and closes the review.

Diagram: Workflow for Formal Group Reviews



To see a simple example of how to use Crucible with two people, see [Getting Started with Crucible](#).

Roles and Status Classifications

This page explains the roles & status classifications in Crucible.

- [Roles in Crucible](#)
 - [Author](#)
 - [Creator/Moderator](#)
 - [Reviewer](#)
 - [User](#)
- [Status Classifications in Crucible](#)
 - [Draft](#)
 - [Under Review](#)
 - [Summarized](#)
 - [Closed](#)
 - [Abandoned](#)

Roles in Crucible

Author

The *author* is the person primarily responsible for acting on the outcomes of the review. In the vast majority of cases the author will be the person who made the code change under review.

Creator/Moderator

The *creator* is the person who [creates the review](#). In most cases this person will also act as [moderator](#).

The *moderator* is the person responsible for [creating](#) the review, [approving](#) the review, determining when reviewing is finished, [summarising](#) the outcomes and [closing](#) the review. By default, the moderator is the [creator](#).

Reviewer

A *reviewer* is a person assigned to [review the change](#). Reviewers can make [comments](#) and indicate when they have [completed their review](#). The [moderator](#) and [author](#) are implicitly considered reviewers.

User

A *user* is a person using Crucible.

Status Classifications in Crucible

Draft

Draft Reviews are not yet completed or released to the reviewers.

Under Review

Reviews Under Review are either waiting for attention by reviewers or waiting to be summarized.

Summarized

Summarized reviews are past the reviewing phase. The moderator can still add conclusions or comments.

Closed

Closed reviews are complete.

Abandoned

Abandoned reviews are 'in the trash'. Reviews must be Abandoned before they can be deleted.

See also the [Glossary of terms](#) used in Crucible.

Using Favourites

This page contains instructions on using the '**Favourites**' feature in Crucible to select, view and manage items of interest.

On this page:

- [Favourites Overview](#)
- [Adding Items to Your Favourites](#)
 - [Adding a Review to Your Favourites](#)
 - [Adding a Review Comment Thread to Your Favourites](#)
 - [Adding a Project to Your Favourites](#)
 - [Adding a Person to Your Favourites](#)
 - [Adding a Changeset to Your Favourites](#)
 - [Adding a File or Folder to Your Favourites](#)
 - [Adding a Repository to Your Favourites](#)
- [Viewing Your Favourite Items](#)
- [Renaming an Item In Your Favourites](#)
- [Removing an Item From Your Favourites](#)

Favourites Overview

Crucible allows you to tag certain items as your favourites. You can select code reviews, changesets, files, people and repositories to be added to your favourites. Once your favourites list is created, you can view it or see a stream of all activity relating to your favourite items. We suggest you select items that you are currently working on as favourites, to create a more relevant personalised view.

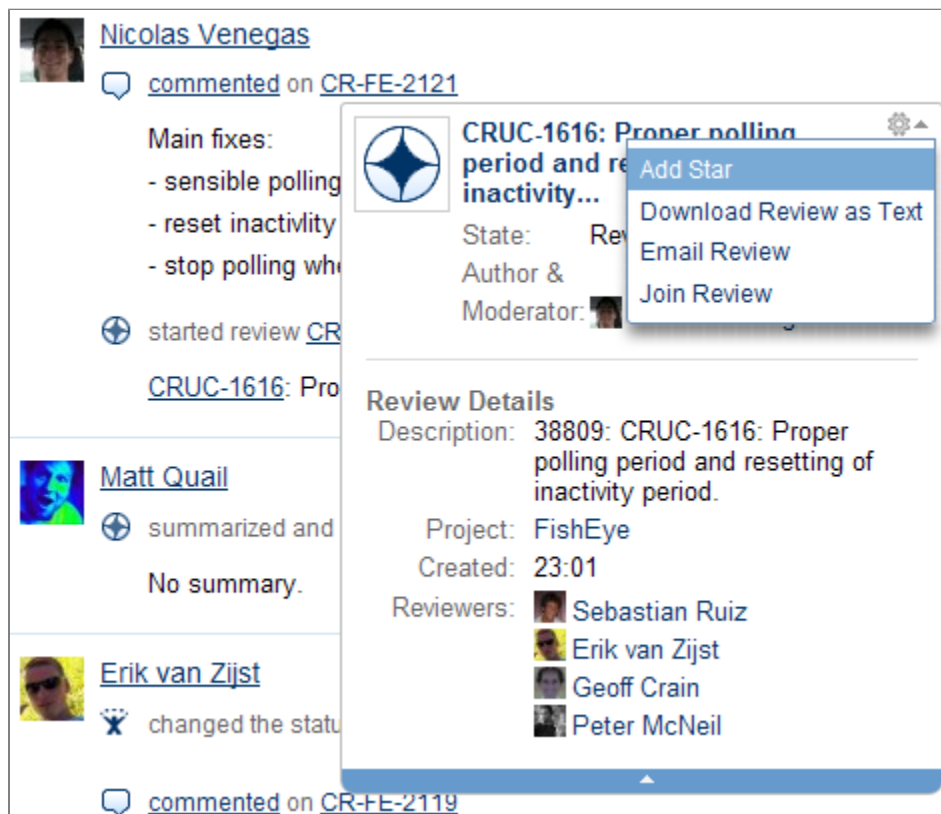
Adding Items to Your Favourites

To add an item to your favourites, follow one of the steps below.

Adding a Review to Your Favourites

To add a review to your favourites, hold the mouse cursor over the review name when it appears in a menu screen. The Review Hover menu appears. At the top right of the Review Hover menu, click the small grey cog icon that indicates the '**Tools**' menu. The Tools menu opens. In the Tools menu, click '**Add Star**'. This will add it to your favourites.

Screenshot: Adding a Review To Your Favourites

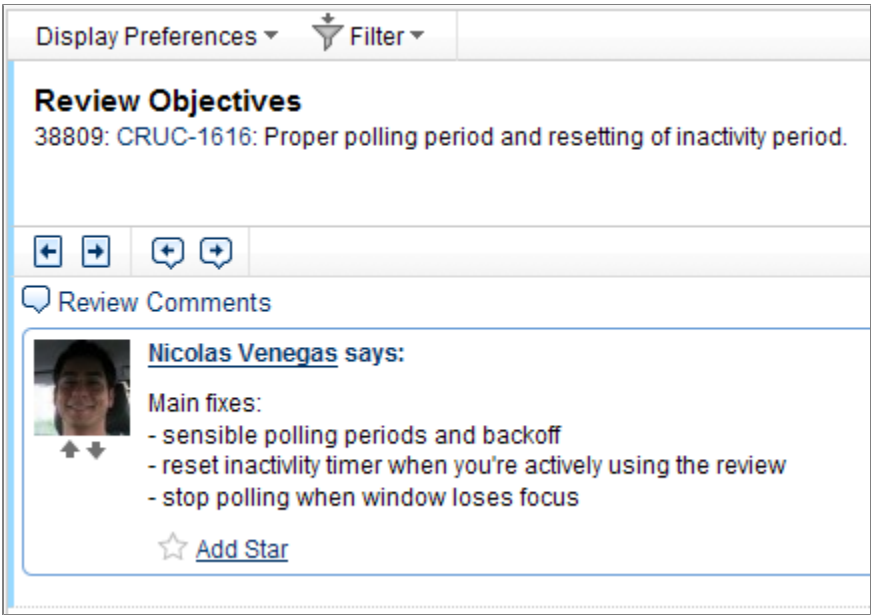


Adding a Review Comment Thread to Your Favourites

To add a review comment thread to your favourites, click the link '**Add Star**' next to the grey star icon at the bottom of the first comment of the

thread. From then on, new comments will be shown in your favourites activity stream.

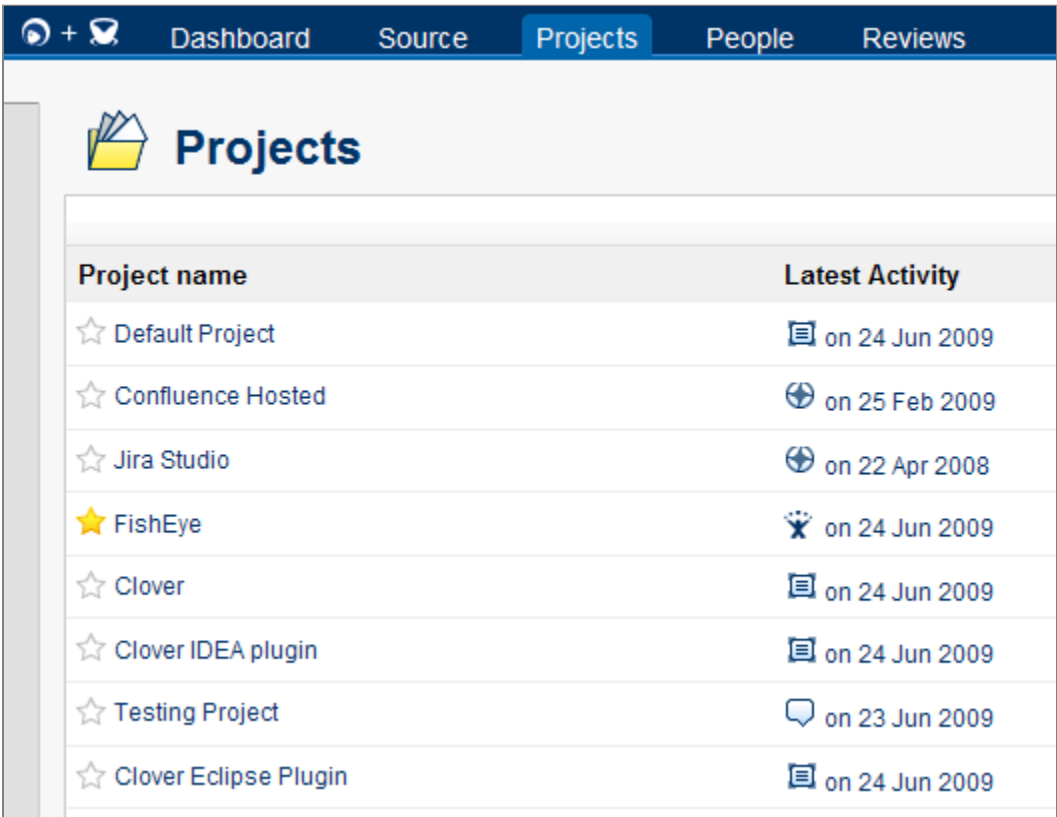
Screenshot: Adding a Review Comment Thread to Your Favourites



Adding a Project to Your Favourites

To add a project to your favourites, click the 'Projects' tab. The Projects view appears. Here, simply click the grey star icon that appears next to the desired project name. The star icon will turn yellow, showing that it is selected as a favourite.

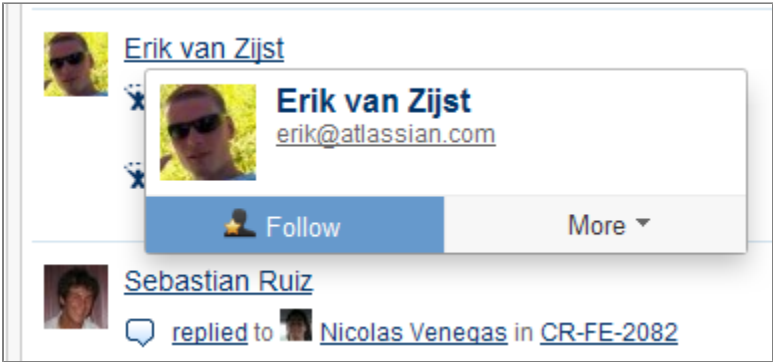
Screenshot: Adding a Project to your Favourites



Adding a Person to Your Favourites

To add a person to your favourites, simply hold the mouse cursor over their username wherever it appears. The User Hover menu will appear. In the User Hover menu, click **Follow**. This will add the person to your favourites.

Screenshot: Adding a Person to Your Favourites



Adding a Changeset to Your Favourites

To add a changeset to your favourites, firstly open the changeset desired from the **Source** tab. Once the changeset is open in Crucible, simply click the grey star icon that appears next to its name. The name appears in the breadcrumb links at the top of the screen.

Screenshot: Adding a Changeset to Your Favourites



Adding a File or Folder to Your Favourites

To add a file to your favourites, firstly open the file or folder desired, from the **Source** tab. Once the file or folder is open in Crucible, simply click the grey star icon that appears next to its name. The name appears in the breadcrumb links at the top of the screen.

Screenshot: Adding a File or Folder to Your Favourites



Adding a Repository to Your Favourites

Adding a repository to your favourites (requires FishEye), click the **Source** tab. The the **Source** view appears. Here, simply click the grey star icon that appears next to the name of the desired repository. The star icon will turn yellow, showing that it is selected.

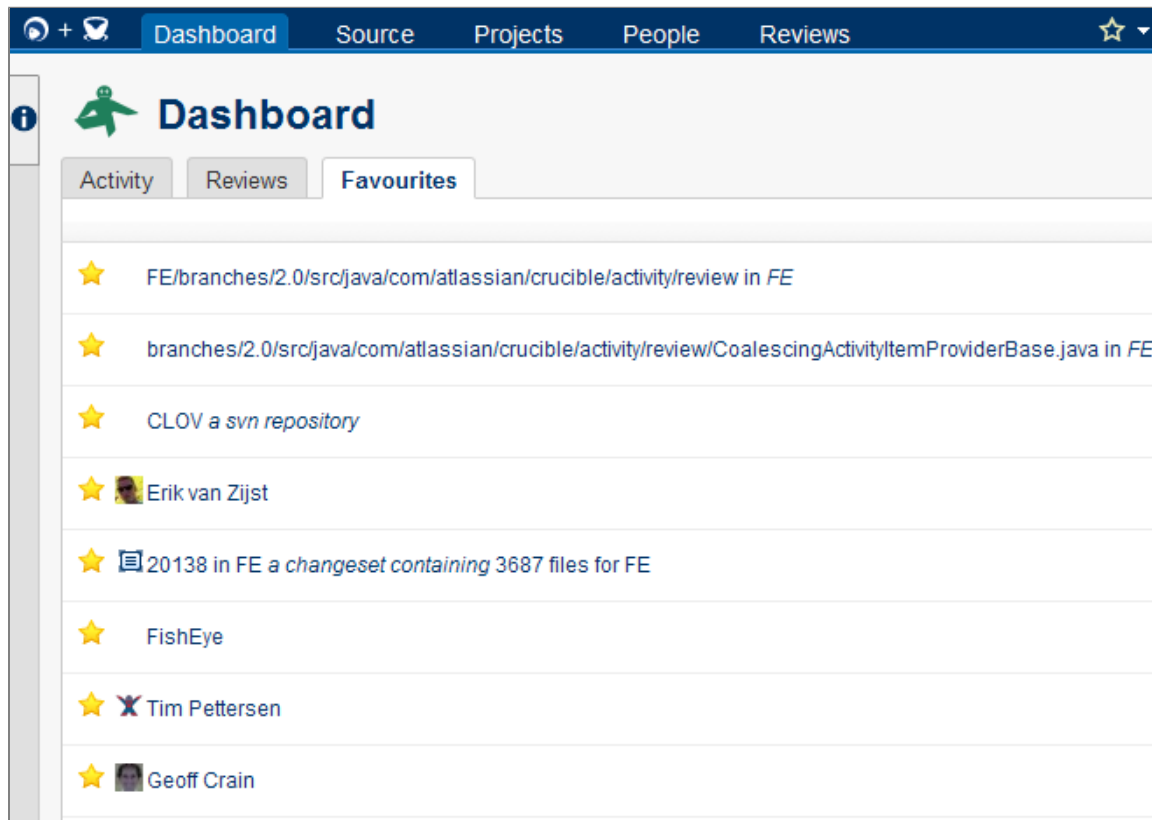
Screenshot: Adding a Repository to Your Favourites

Repository	State	Commit History (12 Months)
☆ FE	Running	
★ CLOV	Running	

Viewing Your Favourite Items

To view your favourite items, click **'Dashboard'** tab at the top left of the page and then the **'Favourites'** second level tab, beneath that.

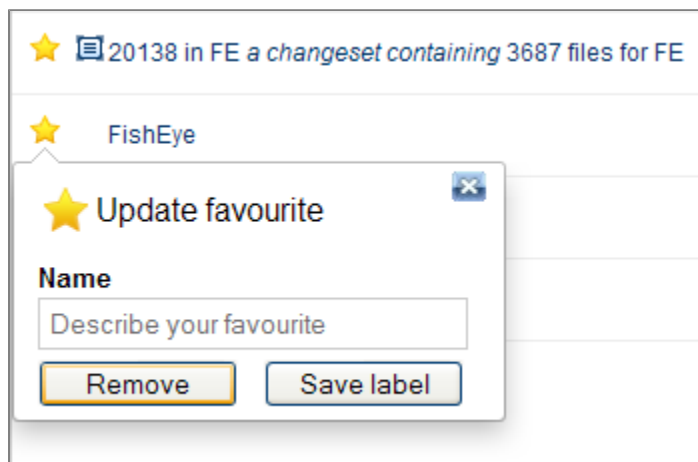
Screenshot: Viewing Your Favourites



Renaming an Item In Your Favourites

To rename an item in your favourites, open the Favourites drop-down menu (the gold star icon located at the top centre of the Crucible screen, next to your user menu). Select the option called **'Manage favourites'**. The Dashboard favourites page opens, showing all of your favourites in the system. To rename any item (changing its favourite display name only — not the name of item itself), simply click the yellow star to the left of its name. A small pop-up menu will appear, the **'Update Favourites'** menu. Type the desired name into the **'Name'** field and click the **'Save label'** button. The label will be updated for the favourites view.

Screenshot: Renaming an Item in Your Favourites

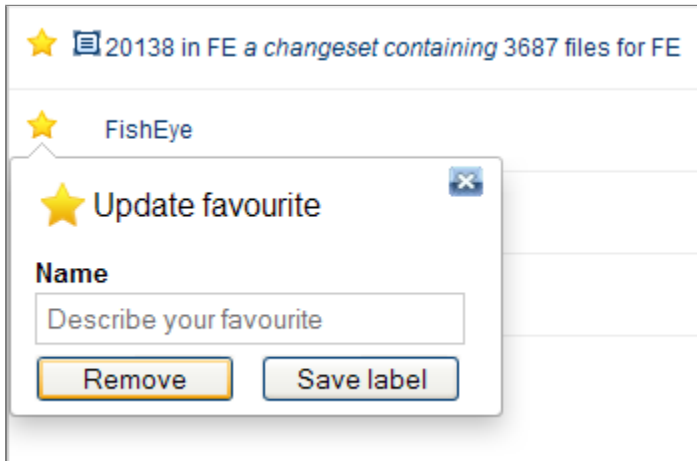


Removing an Item From Your Favourites

To remove an item from your favourites, open the Favourites drop-down menu (the gold star icon located at the top centre of the Crucible screen, next to your user menu). Select the option called **'Manage favourites'**. The Dashboard favourites page opens, showing all of your favourites in the system. To remove any item, simply click the yellow star to the left of its name. A small pop-up menu will appear, the **'Update Favourites'**

menu. Click the '**Remove**' button. The star will turn grey, showing that it has been removed from your favourites.

Screenshot: Removing an Item From Your Favourites



Crucible Administrator's Guide

Once you have [installed](#) and [configured](#) Crucible, you can access the **Administration** pages at the following address (where 'HOSTNAME' is the name of the server where you installed Crucible).

```
http://HOSTNAME:8060/admin/
```

The '**Admin Menu**' allows you to administer your Crucible/FishEye instance and manage your repositories.

For more information on administering FishEye, please refer to the [FishEye documentation](#).

Information in the *Crucible Administrator's Guide*:

- [Backing Up and Restoring Crucible Data](#)
- [Creating a Permission Scheme](#)
- [Creating a Project](#)
- [Crucible and FishEye](#)
- [Customising Email Notifications](#)
- [Customising the Defect Classifications](#)
- [Customising the Welcome Message](#)
- [Deleting a Project](#)
- [Editing a Project](#)
- [JIRA Integration in Crucible](#)
- [Migrating to an External Database](#)
- [Trusted Applications](#)

Backing Up and Restoring Crucible Data

Crucible data can be backed up from the admin interface or command line. This page contains the command syntax, options and the required procedure to backup and restore your Crucible instance.

On this page:


- [Backing Up Crucible Data](#)
 - [The Crucible Admin Interface Backup Process](#)
 - [The Crucible Command Line Backup Process](#)
 - [Components of a Crucible Backup](#)
 - [Backup Command Line Options](#)
 - [Command Line Examples](#)
 - [Advanced Backup Command Line Settings](#)
 - [Known Limitations](#)
 - [Scheduling Crucible Backups](#)
- [Restoring Crucible Data](#)
 - [The Crucible Data Command Line Restoration Process](#)
 - [Restore Command Line Options](#)

- [Advanced Command Line Restore Settings](#)
 - [Notes on Migrating Backup Data](#)
 - [Command Line Example: Migrating Backup Data to MySQL](#)

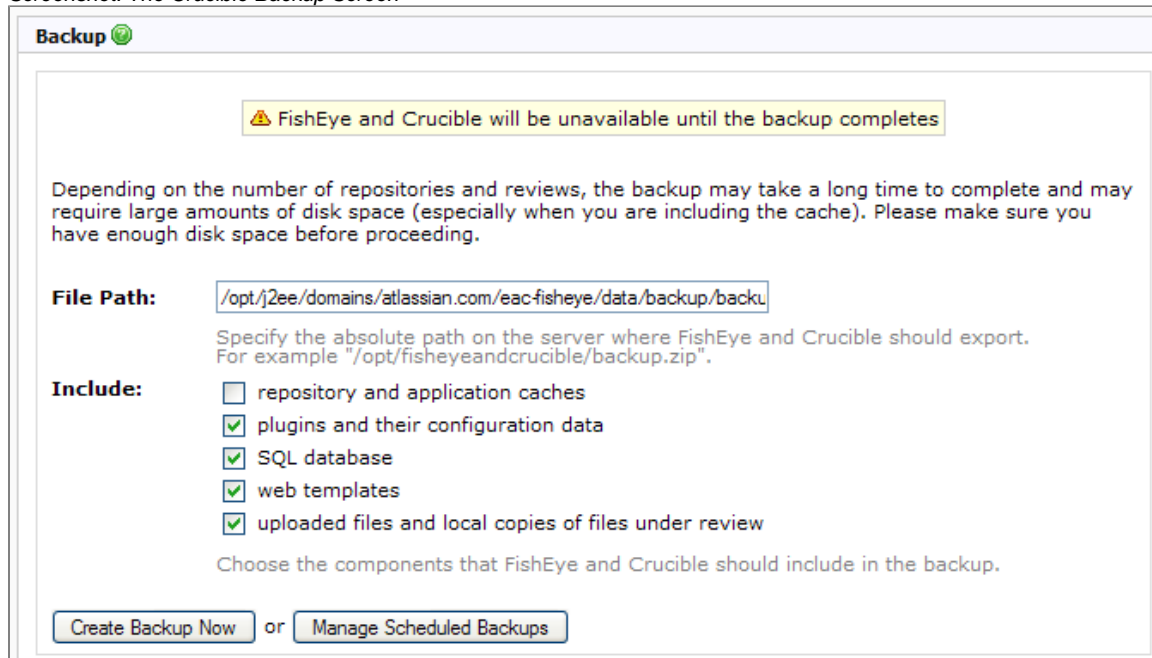
Backing Up Crucible Data

The Crucible Admin Interface Backup Process


1. Navigate to the Crucible **'Admin'** screen (click the **'Administration'** link in the footer of any Crucible page).
2. On the Admin screen, click **'Backup'** under the **'System'** heading in the left navigation bar. The Backup screen opens.
3. On the Backup screen, the **'File Path'** field indicates where the backup file (in .zip format) will be stored. You can manually edit this path to change it. Under the heading **'Include'**, a list of checkboxes is shown, with the following items:
 - Plugins and their configuration data
 - SQL database
 - Web templates
 - Uploaded files and local copies of files under review.
 - Repository and application caches.


 Repository and application caches contain temporary data stored from repository scans and library caches that improve startup time. Both will be recreated automatically by re-scanning the source repositories, so the backup files can be reduced by a significant amount by excluding these (if the cost of re-scanning is acceptable).
4. Once you have chosen your options, click **'Create Backup Now'**.

Screenshot: The Crucible Backup Screen



The Crucible Command Line Backup Process

-  Your Crucible instance must be running during the backup.
1. Open a command line interface on the Crucible server computer.
 2. Navigate to the `FISHEYE_HOME/bin/` directory.
 3. Run the backup command on the command line with the desired options.
 4. The backup is created as a new Zip archive file and placed in the `FISHEYE_INST/backup/` directory.

 Note that if your Crucible instance uses a custom `FISHEYE_INST` directory, make sure the environment variable is properly set when running the backup command.

Components of a Crucible Backup

The Crucible backup is highly configurable and allows for many different configurations. This table shows the various components of the backup, what they are for and how they can be used.

Component	Purpose	Defaults
SQL Database	Refers to the SQL content database (used by both FishEye and Crucible and containing all user profile data, reviews and their comments).	Backed up by default.
Cache	The cache contains data that reflects the state of FishEye's repositories. Without it, FishEye must re-scan its repositories after a backup is restored. The cache also contains OSGI library data that increases startup time. These too can be excluded and will be generated automatically when the application is started.	The cache is not backed up by default as it tends to be large (running a risk of pushing the maximum file size for Java backups), whilst also representing replaceable data.
Plugins	Plugins are 3rd-party extensions that you may have installed, and configuration for all plugins (this includes configuration for Crucible's set of standard plugins).	Configuration data for all plugins are backed up by default, as well as all plugins installed in <code>FISHEYE_INST/var/plugins/user</code> .
Templates	In this context, these are custom freemarker templates that you or your users have created. They live in <code>FISHEYE_INST/template</code> .	Templates are backed up by default. You can choose to exclude them from the backup if your templates directory is covered by some other backup mechanism.
Uploads	In this context, uploads refers to files which are added to Crucible via the web interface (such as patch file reviews). It also includes each repository-backed file that went under review, when Crucible is configured to make a local copy of every reviewed file .	Uploads are backed up by default. You can choose not to back them up for example when the <code>FISHEYE_INST/var/data/uploads</code> directory is already covered by some other backup mechanism.

Note that the backup will always include the configuration data (`config.xml`), your license file and the FishEye user data.

Backup Command Line Options

These examples are for use in a Linux-like operating system. When using these commands on Windows, use the filename `fisheyectl.bat` and use the correct slashes. Run the command from the `FISHEYE_HOME/bin/` directory.

The basic syntax of the backup command is as follows:

```
$ ./fisheyectl.sh backup [OPTIONS]
```


To see inline help for all backup options, run the following command in the `FISHEYE_HOME/bin/` directory:

```
$ ./fisheyectl.sh backup --help
```

|| Option || Switch || Description || Default setting ||

Quiet mode	-q OR --quiet	Suppresses output	No
Output filename	-f OR --file	Specify a different path and filename to the <code>FISHEYE_INST/backup/backup_YYYY-DD-MM_HHmm.zip</code> file. When filename is omitted, the backup filename contains the date and time.	<code>FISHEYE_INST/backup/</code> is the default directory.
Compression level	--compression OR -c	Sets the Zip compression level, from 1-9. Runs at level 6 if no argument is passed.	Yes (6)
Anonymise	-a OR --anonymise	Anonymises the SQL database by replacing all text with 'x'. This is only useful when sending a backup to Atlassian as part of a support case. Please do not anonymise data unless the Support Engineer handling your support case has specifically requested the data anonymised (as often anonymised data will not help reproduce the issue).	No
Cache Backup	--cache	Include the repository caching files in the backup. These hold information gained from scanning the repositories and can be quite large (many gigabytes). However, it can shorten the time needed to re-scan the repositories after data is restored.	No. By default, the cache data is excluded from backups.

Command Line Examples

 These examples are for use in a Linux-like operating system. When using these commands on Windows, use the filename `fisheyectl.bat` and use the correct slashes. Run the command from the `FISHEYE_HOME/bin/` directory.

Backing up with compression of 9, quiet mode and setting an output location

```
$ ./fisheyectl.sh backup --compression 9 -q -f /application_backups/fisheye/20090215.zip
```

Backup including cache data (also includes all default components)

```
$ ./fisheyectl.sh backup --cache
```

Restoring a backup with cache data (also restores all default components)

```
$ ./fisheyectl.sh restore --cache
```

Advanced Backup Command Line Settings

In some cases it might be preferable to only backup a limited set of items. This could be useful when your instance uses an external database such as MySQL or PostgreSQL and your DBA has already configured automatic backups in the database. The commands below allow this.

Option	Switch	Description	Default
Exclude Plugins	--no-plugins	Excludes plugins from the backup.	No. By default, plugins are included in every backup.
Exclude Templates	--no-templates	Excludes templates from the backup.	No. By default, templates are included in every backup.
Exclude Uploads	--no-uploads	Excludes uploaded files (such as patch reviews, stored in Crucible's internal database) from the backup.	No. By default, uploads are included in every backup.
Exclude SQL Database	--no-sql	Excludes the SQL content database used by both FishEye and Crucible.	No. By default, this data is included in every backup.
Show help	--help OR -h	Shows inline help on the command line.	No

Known Limitations

Please note that the below limitations are common for any Java based backup tool.

Archives Containing Over 65535 Files

Versions of Java earlier than v1.6 (b25) are incapable of handling zip files that contain more than 65,535 files. The solution for this problem is to either upgrade to a version of Java later than v1.6 (b25), or ensure that the archive does not exceed the threshold (contains less than 65,535 files). The FishEye cache (not included in backups by default) can be a contributor of many small files. Hence, exclude the cache from backups if this is likely to be a concern.

Archives Larger Than 4GB

Java has trouble reading and writing zip files that are larger than 4GB. As of release 1.5 Java appears capable of reliably creating archives that are over 4GB, but remains unable to extract them. For details see [Sun's bug report](#). Also be aware of the fact that some file systems (including FAT32) have trouble with files larger than 4GB.

As a workaround, make sure you do not create archives that are larger than 4GB. The FishEye cache (not included in backups by default) can be a contributor of a lot of small files (although these tend to compress very well). If you still want to archive everything and end up with an archive that is too large, consider creating separate backups for the FishEye cache and uploaded files respectively.

Scheduling Crucible Backups

To set a schedule for automatic backups, open the administration screen and click '**Backup**' under '**System**' on the left navigation bar. The 'Backup' page opens. Now, click the link '**Manage Scheduled Backups**' at the bottom of the page. The 'Scheduled Backups' page opens.

On the 'Scheduled Backups' page, click '**Edit**' to adjust the backup schedule. Set the desired options and click '**Save**'.

The options for scheduled backups are detailed in the table below.

Option name	Description	Allowed Values
-------------	-------------	----------------

Disable Scheduled Backups	Stops regular backups from taking place.	On (disabled) or Off (enabled)
Backup path	The path where the backup .zip file will be stored.	Any system or network path that FishEye or Crucible can access.
Backup file prefix	Characters that will be added to the beginning of the backup file name.	Any string of characters that can be used as part of a filename on the local operating system.
Backup file date pattern	Sets a date for the next (or initial) backup to take place.	Any valid date in the format <code>yyyy_MM_dd</code> (year, month, day of the month).
Backup frequency	Sets how often the backup will take place.	Can be set to 'every day' , 'every Sunday' , 'Monday to Friday' and 'first day of the month' .
Backup time (HH:mm)	The time when the backup will take place.	Any valid 24-hour time in the format <code>HH:mm</code> (hours, minutes).
Include	Specifies which items must be included in the backups (these components are explained at the top of this page).	As per the options for regular on-demand backup (These components are explained at the top of this page).

Screenshot: Scheduling Backups in FishEye and Crucible

Be aware that scheduled backups can fill up disks unless you regularly move or delete old archives.

Restoring Crucible Data

The Crucible Data Command Line Restoration Process


There is currently no way to restore a backup from the web interface because Crucible must be shut down during a data restore.

Restoring a backup will irreversibly overwrite the data of your installation with the data from the backup archive.

1. [Install Crucible](#) into a new, empty directory (this must be the same version that the backup was created from, or later).
 Note that you cannot restore data into versions of Crucible which are older than the version that created the backup.
2. Make sure the Crucible instance is not running.
3. Open a command line interface on the Crucible server computer.
4. Run the restore command on the command line with the desired options.
5. The specified elements will be restored.
6. Start the Crucible instance.

7. When using FishEye integrated with Crucible, you will need to re-index your repositories after restoring data, unless the backup archive was created with the `--cache` option.

Restore Command Line Options

 These examples are for use in a Linux-like operating system. When using these commands on Windows, use the filename `fisheyectl.bat` and use the correct slashes. Run the command from the `FISHEYE_HOME/bin/` directory.

The basic syntax of the restore command is as follows:

```
$ ./fisheyectl.sh restore -f /path/to/backup_2009-10-02_1138.zip [OPTIONS]
```

To see inline help for all backup options, run the following command in the `FISHEYE_HOME/bin/` directory:

```
$ ./fisheyectl.sh restore --help
```

Restores a FishEye/Crucible backup instance.

If you are using an external database (as opposed to the default built-in database), make sure the JDBC driver file is present in the `FISHEYE_INST/lib` directory when running restore.

Option	Switch	Description	Default
Suppress output	<code>--quiet</code> OR <code>-q</code>	Suppress the output messages from the restore program on the command line.	No
Choose file to restore from	<code>--file PATH/FILENAME</code> OR <code>-f PATH/FILENAME</code>	Restore the backup from PATH/FILENAME.	Yes (required)
Show inline help	<code>--help</code> OR <code>-h</code>	Displays help for options on the command line.	No

Advanced Command Line Restore Settings

By default, the restore program will restore all items found in the backup archive (so if you included the caches using the `--cache` option, these will automatically be restored). However, it is possible to only restore a subset of items from the backup, by explicitly specifying the item names on the command line and only those will be restored.

Option	Switch	Description
Restore FishEye cache	<code>--cache</code>	Restore the repository cache backup.
Restore plugins	<code>--plugins</code>	Restore 3rd-party plugins and their configuration data.
Restore templates	<code>--templates</code>	Restore freemarker templates from the backup (the restored instance will use the built-in templates).
Restore uploads	<code>--uploads</code>	Restore uploads (e.g. patch files uploaded into Crucible and contents of files under review).
Restore Crucible reviews	<code>--sql</code>	Restore the SQL database containing user profiles, reviews and review comments.
Set database type	<code>--dbtype</code> OR <code>-t</code>	SQL database type ('mysql', 'postgresql' or 'built-in'). Only required when restoring to a database location different to that used at backup time.
Set JDBC URL	<code>--jdbcurl</code> OR <code>-j</code>	JDBC URL of the SQL database. Only required when restoring to a database location different to that used at backup time (not applicable for 'built-in').
Set JDBC username	<code>--username</code> OR <code>-u</code>	JDBC username of the SQL database. Only required when restoring to a database location different to that used at backup time (not applicable for 'built-in').
JDBC password	<code>--password</code> OR <code>-p</code>	JDBC password of the SQL database. Only required when restoring to a database location different to that used at backup time (not applicable for 'built-in').


JDBC class	--driver OR -d	Specifies the JDBC driver class name needed to access the SQL database. Only required when restoring to a database location different to that used at used at backup time and when using a different JDBC driver than the standard driver associated with the database specified though --dbtype. (Not applicable for 'built-in'.)
------------	----------------	--

Notes on Migrating Backup Data

When the process restores a SQL database, it looks at the configuration data (`config.xml`) included in the backup archive to learn which database product was used and how to connect to it. When Crucible uses the built-in HSQLDB database (which is the default), the restored instance will also use that.

However, when the restored instance will use a different database than the backed up instance (for instance, HSQLDB was used at the time the backup was created, but it needs to be restored on MySQL), use the command line options to point the process to the new database.

Command Line Example: Migrating Backup Data to MySQL

 These examples are for use in a Linux-like operating system. When using these commands on Windows, use the filename `fisheyectl.bat` and use the correct slashes. Run the command from the `FISHEYE_HOME/bin/` directory.

Restoring to a Crucible instance that uses a different database (ensure the mysql driver jar file is present in the `FISHEYE_INST/lib` directory)

```
$ ./fisheyectl.sh restore \
--username john \
--password smith \
--jdbcurl jdbc:mysql://localhost:3306/crucible \
--dbtype mysql \
--file /path/to/backup_2009-10-02_1138.zip
```

Creating a Permission Scheme

This page contains information on how to create a permission scheme in Crucible.

On this page:

- [Introduction to Crucible Permissions](#)
- [Creating a Permission Scheme](#)
- [Editing a Permission Scheme](#)
- [List of Crucible Permissions](#)

Introduction to Crucible Permissions

A *permission* is the ability to perform a particular action in Crucible, e.g. 'Create Review'.

A *permission scheme* assigns particular [permissions](#) to any or all of the following:

- Particular [Users](#).
- Particular [Groups](#).
- All logged-in users.
- [Anonymous Users](#)
- People in particular [Review Roles](#), such as:
 - 'Author';
 - 'Reviewer';
 - 'Creator';
 - 'Moderator'.


The scheme's permissions will apply to all reviews belonging to the [project\(s\)](#) with which the scheme is associated.


You can create as many permission schemes as you wish. Each permission scheme can be associated with many projects or just one project, allowing you to tailor appropriate permissions for individual projects as required.

Creating a Permission Scheme

[To create a permission scheme,](#)

1. From the '**Admin Menu**', click '**Permission Schemes**'.
2. The '**Permission Schemes**' page will be displayed, showing a list of existing permission schemes. Click the '**Create a New Permission Scheme**' link, which appears below the list.
3. In the '**Name**' field, type a short phrase to uniquely identify your project (see screenshot 1 below).
4. Click the '**Create**' button to create your new permission scheme. The '**Edit Permission Scheme**' page will be displayed for your new permission scheme (see screenshot two, below).

 Your new permission scheme will have the default assignees shown in the table above.
5. To edit the assignees for a permission, click the '**Edit**' link corresponding to the permission. The '**Edit Permission Scheme**' page will be displayed.
6. Choose the appropriate assignee(s) for this permission:

 **Note:** for ongoing ease of management, it is recommended that you grant permissions to [groups](#) or [participants](#) rather than to individual users.

 - To assign this permission to [anonymous users](#), select the '**Allow Anonymous users**' check-box.
 - To assign this permission to all logged-in users, select the '**Allow All logged in users**' check-box.
 - To assign this permission to a particular user, type their username into the '**Individual users**' field (hint: you can type just part of the name, then press <Enter> to select from a list of matching usernames).
 - To assign this permission to a particular group of users, type the group name into the '**Groups**' field (hint: you can type just part of the group name, then press <Enter> to select from a list of matching groups).
 - To assign this permission to users who belong to a particular [participant](#) ('**Reviewer**' / '**Moderator**' / '**Author**' / '**Creator**'), select the corresponding check-box.
7. Click the '**Save**' button.

 Next step: see [Associating a Permission Scheme with a Project](#).

Screenshot 1: Adding a Permission Scheme

Name	Projects using this scheme	
default	Default Project	edit copy
Name: <input type="text" value="Top Secret"/>	<input type="button" value="create"/>	<input type="button" value="cancel"/>

Editing a Permission Scheme

To edit a permission scheme,

1. From the '**Admin Menu**', click '**Permission Schemes**'.
2. Click '**edit**' next to the scheme you wish to change. The '**Edit Permission Scheme**' page will be displayed.
3. On the '**Edit Permission Scheme**' page, you can change the groups or users that are allowed individual permissions by clicking '**edit**' next to the permission in question.
4. When you have finished editing, click the '**Save**' button.

Screenshot: Edit a Permission Scheme

Edit Permission Scheme - Top Secret

Top Secret
Rename

Permissions	Users / Groups / Review Roles	
Edit Review Details Ability to change review details including the set of revisions being reviewed.	<ul style="list-style-type: none"> Anonymous users: false All logged in users: false Individual users: Groups: Review Roles: Moderator Creator 	edit
View Ability to view a review.	<ul style="list-style-type: none"> Anonymous users: true All logged in users: true Individual users: Groups: Review Roles: Moderator Reviewer Author Creator 	edit
Abandon Ability to abandon (i.e. cancel) a review.	<ul style="list-style-type: none"> Anonymous users: false All logged in users: false Individual users: Groups: Review Roles: Moderator Creator 	edit
Re-Open Ability to re-open a closed review.	<ul style="list-style-type: none"> Anonymous users: false All logged in users: false Individual users: Groups: Review Roles: Moderator Creator 	edit
Uncomplete Ability to indicate they have not completed a review, after indicating they have completed a review.	<ul style="list-style-type: none"> Anonymous users: false All logged in users: false Individual users: Groups: Review Roles: Reviewer 	edit
Reject Ability to reject a review submitted for approval.	<ul style="list-style-type: none"> Anonymous users: false All logged in users: false Individual users: 	edit

Screenshot: Editing the 'View' Permission

Edit User Action "View" on scheme "Top Secret"

View: Ability to view a review.

Allow Anonymous users: ☒

Allow All logged in users: ☒

Start typing a user name

Individuals: then press enter to select.

Start typing a group name

Groups: then press enter to select.

Review Participants: ☒ Reviewer ☒ Moderator ☒ Author ☒ Creator

save

List of Crucible Permissions

The following permissions are available:

Permission	Description	Default Assignees
'Edit'	Ability to edit a review's details and change the set of revisions being reviewed.	'Creator' 'Moderator'

'View'	Ability to view a review. (People without this permission will not know that the review exists.)	Anonymous users All logged-in users 'Creator' 'Author' 'Reviewer' 'Moderator'
'Abandon'	Ability to abandon (i.e. cancel) a review.	'Moderator' 'Creator'
'Re-Open'	Ability to re-open a closed or abandoned review.	'Creator' 'Moderator'
'Uncomplete'	Ability of a reviewer to change their individual review status from 'Complete' to 'Uncomplete'.	'Reviewer'
'Reject'	Ability to reject a review submitted for approval (i.e. prevent it from being issued to reviewers).	'Moderator'
'Complete'	Ability of a reviewer to change their individual review status to 'Complete'.	'Reviewer'
'Comment'	Ability to add or remove a comment to or from a review.	'Creator' 'Author' 'Reviewer' 'Moderator'
'Approve'	Ability to approve a review (i.e. issue it to the reviewers).	'Moderator'
'Submit'	Ability to submit a review for approval (i.e. request that the review be issued to the reviewers).	'Creator' 'Author'
'Close'	Ability to close a review once it has been summarised.	'Moderator'
'Delete'	Ability to delete a review.	'Creator' 'Moderator'
'Summarise'	Ability to summarise a review. (Normally this would be done after all reviewers have completed their review.)	'Moderator'
'Create'	Ability to create a review.	All logged-in users
'Recover'	Ability to resurrect an abandoned (i.e. cancelled) review.	'Creator' 'Moderator'

Associating a Permission Scheme with a Project


This page explains how to associate a permission scheme with a Crucible project and show details of the default permission schemes included with Crucible.

On this page:

- [Associating a Permission Scheme with a Crucible Project](#)
- [Overview of the Permission Schemes Bundled with Crucible](#)
 - [Default Permission Scheme Settings](#)
 - [Agile Permission Scheme Settings](#)
- [Related Links](#)

Associating a Permission Scheme with a Crucible Project

To associate a permission scheme with a project,

1. From the 'Admin Menu', click 'Project List'.
2. The 'Projects List' page will be displayed. Find the project you wish to associate with your permission scheme, and click its 'Edit' link.
3. The 'Edit Project' page will be displayed.
4. Under the heading 'Project Permissions Scheme', click the 'Permission Scheme' drop-down list to select your permission scheme.
 You will be shown a list of the schemes that have been created in Crucible. You can [create a new permission scheme](#) if necessary.
5. Click the 'Save' button.


Overview of the Permission Schemes Bundled with Crucible

Crucible comes with two permission schemes. 'Default' and 'Agile'. The following tables show the default settings in detail; note that these can be easily edited by admin users to suit your needs.

Default Permission Scheme Settings

This table shows the various permissions and which user groups have them by default.

Permission	Anonymous	All Logged In	Individuals	Groups	Review Roles
Abandon	false	false	None	None	Creator, Moderator
Approve	false	false	None	None	Moderator
Close	false	false	None	None	Moderator
Comment	false	false	None	None	Reviewer, Creator, Author, Moderator
Complete	false	false	None	None	Reviewer
Create	false	true	None	None	None
Delete	false	false	None	None	Creator, Moderator
Edit Review Details	false	false	None	None	Creator, Moderator
Recover	false	false	None	None	Creator, Moderator
Reject	false	false	None	None	Moderator
Re-Open	false	false	None	None	Moderator
Submit	false	false	None	None	Creator
Summarize	false	false	None	None	Moderator
Uncomplete	false	false	None	None	Reviewer
View	false	true	None	None	Reviewer, Creator, Author, Moderator

 The default permission scheme has changed since Crucible 1.6.

Agile Permission Scheme Settings

This table shows the various permissions and which user groups have them by default.

Permission	Anonymous	All Logged In	Individuals	Groups	Review Roles
Abandon	false	false	None	None	Creator, Author, Moderator
Approve	false	false	None	None	Creator, Author, Moderator
Close	false	false	None	None	Reviewer, Creator, Author, Moderator
Comment	false	false	None	None	Reviewer, Creator, Author, Moderator
Complete	false	false	None	None	Reviewer
Create	false	true	None	None	None
Delete	false	false	None	None	Creator, Author, Moderator
Edit Review Details	false	false	None	None	Reviewer, Creator, Author, Moderator
Recover	false	false	None	None	Reviewer, Creator, Author, Moderator
Reject	false	false	None	None	Creator, Author, Moderator
Re-Open	false	false	None	None	Reviewer, Creator, Author, Moderator
Submit	false	false	None	None	Creator, Author, Moderator
Summarize	false	false	None	None	Reviewer, Creator, Author, Moderator

Uncomplete	false	false	None	None	Reviewer
View	true	true	None	None	Reviewer, Creator, Author, Moderator

Related Links

- [Creating a Permission Scheme](#)

Creating a Project


A Crucible *project* is a collection of [reviews](#), typically reviews that all relate to the same application. In addition to providing a logical way of grouping reviews together, a project allows you to

- define default [moderators](#), [authors](#) and [reviewers](#) for the reviews in that project.
- define which people are eligible to be [reviewers](#) for the reviews in that project.
- use [permission schemes](#) to restrict who can perform particular actions (e.g. 'Create Review') in that project.

Every Crucible review belongs to a project. Each project has a *name* (e.g. **ACME Development**) and a *key* (e.g. **ACME**). The project key becomes the first part of that project's *review keys*, e.g. **ACME-101**, **ACME-102**, etc:

By default, Crucible contains one project. This default project has the key 'CR' and the name 'Default Project'.

To create a new project,

1. From the '**Admin Menu**', click '**Project List**'.
2. The '**Projects List**' page will be displayed. Click the '**Create a New Project**' link, which appears at the bottom of the list of existing projects.
3. The '**Create Project**' page will be displayed.
4. In the '**Name**' field, type a short phrase that describes your project.
5. In the '**Key**' field, type a few characters to uniquely identify your project. This key must consist of alphabetic and/or numeric characters and hyphens only.
6. In the '**Default Repository**' field, select the repository which contains source code relating to this project.
 This repository is the one that will be searched by default when you [add files to a review](#).
7. In the '**Default Moderator**' field, type the name of the person who will appear by default in the 'Moderator' field when you [create a new review](#); or leave this field blank to force the review's creator to choose a moderator.
8. (Optional) Under '**Default Reviewers**', select the people to whom new reviews in this project will be assigned by default:
 - Select the '**Let allowed review participants join a review**' check-box if you wish to determine the default for the '**Allow anyone to join**' option on the '[Adding Reviewers](#)' screen.
 - In the '**Users**' field, type the name(s) of individual users to whom new reviews will be assigned by default.
 - In the '**Groups**' field, type the name(s) of groups to whose members new reviews will be assigned by default.
9. (Optional) Under '**Allowed Review Participants**', select who will be allowed to have a role (i.e. be an [author/creator/moderator/reviewer](#)) in this project's reviews:
 - In the '**Users**' field, type the name(s) of individual users who will be eligible to be authors/creators/moderators/reviewers for reviews in this project*.
 - In the '**Groups**' field, type the name(s) of groups whose members will be eligible to be authors/creators/moderators/reviewers for reviews in this project*.

* These users will be the only ones whose names appear when a review is [assigned](#).
10. In the '**Permission Scheme**' drop-down list, select the relevant [permission scheme](#) for this project. (A permission scheme controls who can perform particular actions, e.g. 'Create Review'.)
11. In the '**Review Duration**' section, you can enter a value for the number of working days that you want the review to run for. Simply type the number of days into the text entry field marked '**Default duration in week days**' and click '**Save**'.

Screenshot: The Create/Edit Project Screen

Admin Menu	Edit Project
Repository Settings <ul style="list-style-type: none"> Repository List Repository Defaults Global Settings <ul style="list-style-type: none"> Server Settings Security Users Groups ViewCVS URL Mappings Change Admin Password Customize Crucible Defect Classifications Projects Permission Schemes System <ul style="list-style-type: none"> Sys-Info/Support Content Backup Plugins Shutdown 	Identification <p>Name: <input type="text"/></p> <p>Key: <input type="text"/></p> Repository <p>Default Repository: <input type="text" value="svn"/></p> Moderator <p>Default Moderator: <input type="text"/> <small>Start typing a user name then press enter to select.</small></p> Default Reviewers <p><input type="checkbox"/> Let allowed review participants join a review</p> <p>Users: <input type="text"/> <small>Start typing a user name then press enter to select.</small></p> <p>Groups: <input type="text"/> <small>Start typing a group name then press enter to select.</small></p> Allowed Review Participants <p>Users: <input type="text"/> <small>Start typing a user name then press enter to select.</small></p> <p>Groups: <input type="text"/> <small>Start typing a group name then press enter to select.</small></p> Project Permissions <p>Permission Scheme: <input type="text" value="default"/></p> <p><input type="button" value="Save"/></p>

Setting Crucible to Store all Revisions

When creating a project or editing a project's properties, you can set Crucible to save all revisions that are associated with a review to Crucible's database. This allows you to be able to view that file content whether or not the repository is online or accessible to Crucible. It also creates an enhanced audit trail should you require it, saving the review content regardless of whether or not it is deleted or lost from the repository.

Note that the storage of revisions must be set per-project. Also, the storage only applies to reviews created after Revision Storage is enabled. This means that for existing projects, pre-existing reviews will not be stored unless you look at them again after Revision Storage is enabled.

Enabling Revision Storage on a new project

To enable Revision Storing on a new project,

1. When creating a new project, you have the option to turn on revision storing on the '**Create New Project**' page.
2. Under '**Default Content Review Repository**', Click the checkbox labelled '*Store the contents of files in reviews*'.
3. Click '**Save**' to finish.

Enabling Revision Storage on an existing project

To enable Revision Storing on an existing project,

1. From the '**Admin**' screen, click '**Projects**' from the left navigation bar.
2. Click '**Edit**' next to the desired project.
3. Under '**Default Content Review Repository**', Click the checkbox labelled '*Store the contents of files in reviews*'.
4. Click '**Save**' to finish.

Default Review Content Repository

Default Repository:

☒ Store the contents of files in reviews

Screenshot: Enabling Revision Storage

Crucible and FishEye

This page gives an overview of the joint installation of [Crucible](#) and [FishEye](#). Both Crucible and FishEye are Atlassian products.

- **FishEye** allows you to extract information from your source code repository and display it in sophisticated reports.
- **Crucible** allows you to request, perform and manage code reviews.
- Both of these products can run in isolation. However if you are using CVS, Subversion or Perforce you can significantly enhance your Crucible experience by also using FishEye.



Your Crucible installation package includes the files required for FishEye

If you use FishEye and Crucible together, they run as one instance.

Purchasing and Installing Crucible/FishEye

- If you install Crucible, there is no need to do a separate installation of FishEye.
- Upgrading Crucible to also use FishEye requires only a simple licence change in the admin screens.
- When upgrading to Crucible when you have an existing FishEye installation, you can either keep the original FishEye installation or install Crucible and FishEye as a fresh install. Refer to the guide on [upgrading from FishEye to Crucible](#).

FISHEYE_HOME and FISHEYE_INST

Throughout the Crucible documentation, references are made to `FISHEYE_HOME`, which refers to the location of the FishEye application. Because most Crucible users also run FishEye, we use a single value for the sake of simplicity.

Crucible also makes use of this FishEye environment variable:

- `FISHEYE_INST` – the location of the FishEye data.

Refer to the FishEye documentation for more about the [environment variables](#) and how they are used in the [FishEye installation](#).

Detailed Documentation

You can find more information in:

- [Crucible Installation Guide](#)
- [FishEye Installation Guide](#)

Customising Email Notifications

Email notifications in Crucible can be customised to change their formatting, by editing template files. This page contains instructions for this process.

Editing Crucible Email Templates

Template files for Crucible are stored in the `FISHEYE_HOME/templates/` folder.

For Crucible, the set of templates is for plain-text email only. Note that these templates do not support embedding full diffs into notifications. They are only for changing the appearance and order of certain content inside the messages.



If you edit the templates of an operational Crucible instance, you may disrupt notifications that are being sent at that time. To avoid this, shut Crucible down during template editing.

Editing the Subject Line

1. Open the '**crucible-notification-subject.ftl**' template file from `FISHEYE_HOME/templates/` in a text editor.
2. Type in your new text for the email subject, ensuring that all of the content is contained within line 1 of the template. '**crucible-notification-subject.ftl**' is used as the subject template for all Crucible email notifications.
3. Save and close the file.
4. Restarting Crucible will activate the new templates.

Editing the Header

Header information will be included at the beginning of the email body text.

1. Open the '**crucible-notification-header.ftl**' template file from `FISHEYE_HOME/templates/` in a text editor.
2. Add your new header content. '**crucible-notification-header.ftl**' is used as the header template for all Crucible email notifications.
3. Save and close the file.
4. Restarting Crucible will activate the new templates.

Editing the Footer

Footer information will be included at the end of the email body text.

1. Open the '**crucible-notification-footer.ftl**' template file from `FISHEYE_HOME/templates/` in a text editor. '**crucible-notification-footer.ftl**' is used as the footer template for all Crucible email notifications.
2. Add your new footer content.
3. Save and close the file.
4. Restarting Crucible will activate the new templates.

After an edit, the change to the email template will take place immediately. No restart is required.



Try and avoid editing the live template file, as Crucible may try to use it while you are editing. This could have unpredictable results. Instead, back up the template file (it's wise to keep original versions of all these files), edit a copy you have made, then overwrite the 'live' template once you have finished.

Advanced Editing of Crucible Email Templates

The email notification templates use the [Freemarker](#) format. Freemarker is a general templating engine enabling automated content.

If you are familiar with Freemarker, more advanced customisations can be made to the email notification templates. However, you make such adjustments at your own risk.

Note: In Crucible, email notifications are limited to plain-text format only.

Crucible Email Template File List

The following template files for Crucible notification are stored in the `FISHEYE_HOME/templates/` folder.

Template filename	Purpose
crucible-notification-subject.ftl	Subject template
crucible-notification-header.ftl	Header template
crucible-notification-footer.ftl	Footer template
state-closed-notification.ftl	State Closed template
all-completed-notification.ftl	All Completed template
state-changed-notification.ftl	State Changed template

completed-notification.ftl	Completed template
general-notification.ftl	General notification template
uncompleted-notification.ftl	'Uncompleted' template
all-no-longer-completed-notification.ftl	All-No-Longer-Completed template
comment-notification.ftl	Comment template
reply-notification.ftl	Reply template
review-precis-plain.ftl	Precis template

See also [Customising FishEye Email Notifications](#).

Freemarker Data Model for Email Templates

Customising Crucible email templates with Freemarker

See the [Freemarker documentation](#) for instructions on Freemarker syntax. Use the templates that ship with Crucible as a guide to the properties available on each object.

Specific email types will have extra data associated with them, and this data will be available in that particular template (but not in others).

Example

The syntax to access the data-model, using the data model object '**link**' as an example, place this code into the email at the desired position.

```
${notification.link}
```

Customising the Defect Classifications

This page explains how to customise defects and their classifications in Crucible.

On this page:

- [Defects in Crucible Comments](#)
- [Changing Classification Settings](#)
- [Default Crucible Classifications](#)
 - [Ranking](#)
 - [Classification](#)

Defects in Crucible Comments

[Defects](#) are comments made by [reviewers](#) that indicate a problem in a review. Defects can be classified by rank and type, custom classifications can also be defined. The default classifications are shown in the screenshot below.

Screenshot: The List of Defect Classifications

107 + public ClassFileInfo getClassFileInfoInPackage(String name) {
 108 + return getClassFileInfo(getPath() + name);
 114 109 }
 115 110
 116 111 public List getClassFiles() {

Why on Earth are we getting a list of classes here?

Defect: Major Select Classification: Risk-prone

Keep as Draft Discard Post Autosaved at 4:17 PM

165 160
 166 161 protected
 167 162 Array
 168 - for
 169 -
 170 -
 163 + for
 164 +
 165 +
 171 166
 172 - for
 173 -
 174 -
 167 + for
 168 +
 169 +

Select Classification
 Missing
 Extra (superfluous)
 Ambiguous
 Inconsistent
 Improvement desirable
 Not conforming to standards
 Risk-prone
 Factually incorrect
 Not implementable
 Editorial

list();
 Files.values().iterator(); iterator.hasNext();) {
 (AbstractFileInfo) iterator.next();
 getClassFiles();
 Files.values().iterator(); iterator.hasNext();) {
 Info = (BaseSourceFileInfo) iterator.next();
 Info.getClassFiles();
 Files.values().iterator(); iterator.hasNext();) {
 AbstractFileInfo fileInfo = (AbstractFileInfo) iterator.next();
 tmpClasses.addAll(fileInfo.getClassFiles());
 (Iterator iterator = classFiles.values().iterator(); iterator.hasNext();) {
 ClassFileInfo classFileInfo = (ClassFileInfo) iterator.next();
 tmpClasses.add(classFileInfo.getDeclaredClass());

Changing Classification Settings

To change the default classifications:

1. Open the Crucible Admin screen. The 'Admin Menu' opens.
2. Click 'Customize Crucible Defect Classifications' under 'Global Settings' in the Admin Menu.

Only Crucible Admin users have access to this screen.

Any changes made within 'Customize crucible defect classifications' will only affect reviews created after the setting is changed.

Default Crucible Classifications

There are two default defect classifications that are preset in Crucible; ranking and classification. These settings (and their sub-categories) can be edited or removed; other custom classifications can be added.

Ranking

This classification can be set to 'Major' or 'Minor', indicating the importance of the defect.

Classification

This setting helps to define the nature of the defect in particular detail. This classification can be set to one of the options in the following table; the meaning of these is detailed in the table below.

Value	Description
Missing	The defect applies to code or information that is missing (absent).
Extra (superfluous)	The defect applies to code or information that should be removed.
Ambiguous	The defect applies to code or information that is not clear or easy to understand.
Inconsistent	The defect applies to code or information that is applied in several different ways.
Improvement desirable	The defect applies to code or information that needs to be revised.
Not conforming to standards	The defect applies to code or information that breaks established conventions.
Risk-prone	The defect applies to code or information that takes unacceptable risks.
Factually incorrect	The defect applies to code or information that is wrong.

Not implementable	The defect applies to code or information that may be impossible to create.
Editorial	The defect applies to code or information where the classification as a defect may be subject to personal opinion.

Screenshot: Editing Defect Classifications in Crucible

Edit defect classification configuration

Note: Changes will only apply to new reviews
Current defect classification version 4.

Metrics Classifications:

Name
Ranking

Values

Major
remove

Minor
remove

Add field

Remove Classification

Name
Classification

Values

Missing
remove

Extra (superfluous)
remove

Ambiguous
remove

Inconsistent
remove

Improvement desirable
remove

Not conforming to standards
remove

Risk-prone
remove

Factually incorrect
remove

Not implementable
remove

Editorial
remove

Add field

Remove Classification

Add Classification

Save

Cancel

Customising the Welcome Message

To customise the welcome message which is shown when Crucible opens, access the administration page and click '**Customize Front Page**' under '**Global Settings**' on the left navigation bar.

The '**Customize Front Page Messages**' page opens.

On this page, you can provide your own custom text for the Crucible welcome message that is displayed to users when they first log in. You can also provide custom Support text, providing the contact details of your own support organisation, which also appears on the opening page.

You can enter text into the boxes provided for either message and click the small '**Save Welcome Message**' or '**Save Support Message**' button to save it, or enter text for both messages and click '**Save All**'. The changes are made immediately.

Screenshot: Crucible Customize Welcome and Support Messages

Customize Front Page Messages

Here, you can provide your own custom text for the FishEye and Crucible welcome message that is displayed to users when they first log in.

You can also provide custom Support text, providing the contact details of your own support organisation, which also appears on the opening page.

Note: If left blank, the default support or welcome message will be used. See the documentation for more information

Welcome Message

Hail; warm greetings heartily extended to our esteemed customers and partners.

Save welcome message

Support Message

BTCYS

Save support message

Save both

Using HTML

The content in the welcome screen can be arranged using basic HTML tables, image references or anchor tags such as the following:

```
<a href="http://www.atlassian.com">Link to Atlassian Home Page</a>
```

Restoring the default messages

To revert to the default Welcome or Support messages, simply delete all text shown in the text box and click the corresponding 'Save' button.

Manually editing the opening screen

You can also directly edit the XML file that contains the welcome and support messages. This file is called `config.xml`, located in your installation folder.

To do this, simply add the following XML tags to `config.xml`:

```
<content>
  <front-page-message>Example welcome message here</front-page-message>
  <support-message>Example support message here</support-message>
</content>
```

Deleting a Project

Admin users can delete projects under Crucible. To do this, follow the instructions below.




Deleted projects cannot be recovered.

Deleting Projects from the Project List

To delete a project from the Project List;

1. From the '**Admin Menu**', click '**Project List**'.
2. A list of projects appears. With care, click the '**Delete**' link situated to the right of the project you wish to remove. If empty, the project instantly disappears.
3. If the project contains reviews, you will be prompted to either delete all reviews in the project, or move them into the default project.


Screenshot: The Crucible Project Listing

Projects List 								
A project is a set of Crucible content combined with a set of customisable FishEye content.								
Key	Name	Default Repository	Default Moderator	Default Reviewer Users	Default Reviewer Groups	Default Review Time	Edit Crucible Settings	Edit Fisheye Content
CR	Default Project	CLOV				No time restriction	edit	view/edit
CR-CH	Confluence Hosted	CH				No time restriction	edit delete	view/edit
CR-JST	Jira Studio	JST	Don Brown			No time restriction	edit delete	view/edit

Editing a Project

Once projects are created, you can return to the project settings page to change the defaults such as repository, moderator, allowed reviewers, allowed groups and permissions.

To edit project settings,

1. From the '**Admin Menu**', click '**Project List**'.
2. The list of projects will be displayed. Click the '**Edit**' link for the desired project, which appears to the right of the existing project name.
3. The '**Edit Project**' page will be displayed. You can now adjust any of the given settings as desired.
4. In the '**Identification**' section, you can change the the plain language name (as displayed in the Crucible interface) and the project key (used when giving reviews their unique code names).
5. In the '**Default Review Content Repository**' field, you can adjust the repository which contains source code relating to this project.
 This repository is the one that will be searched by default when you [add files to a review](#).
The check box here labelled '**Store the contents of files in reviews**' will cause the source files under review to be stored in the Crucible database along with the comments and review data. This will retain a copy of all the source files that go under review even in the event of disconnecting the repository from Crucible.
6. In the '**Default Moderator**' field, you can adjust the name of the person who will appear by default in the 'Moderator' field when you [create a new review](#); or leave this field blank to force the review's creator to choose a moderator.
7. (Optional) Under '**Default Reviewers**', you can adjust the people to whom new reviews in this project will be assigned by default:
 - Select the '**Let allowed review participants join a review**' check-box if you wish to determine the default for the '**Allow anyone to join**' option on the '[Adding Reviewers](#)' screen.
 - In the '**Users**' field, you can adjust the name(s) of individual users to whom new reviews will be assigned by default.
 - In the '**Groups**' field, you can adjust the name(s) of groups to whose members new reviews will be assigned by default.
8. (Optional) Under '**Allowed Review Participants**', you can adjust who will be allowed to have a role (i.e. be an [author/creator/moderator/reviewer](#)) in this project's reviews:
 - In the '**Users**' field, you can adjust the list of individual users who will be eligible to be authors/creators/moderators/reviewers for reviews in this project*.
 - In the '**Groups**' field, you can adjust the list of groups whose members will be eligible to be authors/creators/moderators/reviewers for reviews in this project*.

* These users will be the only ones whose names appear when a review is [assigned](#).
9. In the '**Permission Scheme**' drop-down list, you can adjust the relevant [permission scheme](#) for this project. (A permission scheme controls who can perform particular actions, e.g. 'Create Review'.)
10. In the '**Review Duration**' section, you can define the default length of time (in week days) for reviews in this project.

Screenshot: The Edit Project screen in Crucible

[Edwin Dawson](#) | [Profile](#) | [Logout](#) | [Help](#)

FishEye and Crucible > Admin > Edit Project

Admin Menu

Repository Settings

- Repository List (new)
- Repository Defaults

Project Settings

- Project List (Create)

Global Settings

- Server Settings
- Security
- Users
- User Mapping
- Avatar Settings
- Groups
- Administrators
- ViewCVS URL Mappings
- Change Admin Password
- Customize Crucible Defect Classifications
- Permission Schemes
- Trusted Applications
- JIRA Servers
- Customize Front Page

System

- Database Configuration
- Sys-Info/Support
- Content
- Backup
- Plugins
- Shutdown

Edit Project

Identification

Name:

Key:

Default Review Content Repository

Default Repository:

☒ Store the contents of files in reviews

Moderator

Default Moderator:

Start typing a user name then press enter to select.

Default Reviewers

☐ Let allowed review participants join a review

Users:

Start typing a user name then press enter to select.

Groups:

Start typing a group name then press enter to select.

Allowed Review Participants

Users:

Start typing a user name then press enter to select.

Groups:

Start typing a group name then press enter to select.

Project Permissions Scheme

Permission Scheme:

Review Duration

Default duration in week days:


Save

Setting the Default Review Duration for a Project

You can set a default time period (duration) that all reviews under a given project will run for. Reviews that are overdue will show up in red on the reviewer's [dashboards](#).

To set a default duration for all reviews in a project,

1. From the '**Admin Menu**', click '**Project List**'.
2. The list of projects will be displayed. Click the '**Edit**' link for the desired project, which appears to the right of the existing project name.
3. The '**Edit Project**' page will be displayed. You can now adjust any of the given settings as desired.
4. In the '**Review Duration**' section, you can define the default length of time (in week days) for reviews in this project.

 Note that the 'Review Duration' only affects the default due date that appears when [creating a review](#) . The review's [creator](#) or [moderator](#) can specify a different date if they wish.


To see instructions for the other items on this page, see the documentation for [Editing a Project](#).


JIRA Integration in Crucible

This page contains instructions for setting up JIRA integration in Crucible.

On this page:

- [Opening the Administration Screen for JIRA Integration](#)
- [Adding a New JIRA Server](#)
 - [Obtaining the Subtask Type ID](#)
 - [Obtaining the Subtask Resolution ID and Subtask Resolution Action ID](#)
- [Editing Default JIRA Server Mappings](#)
- [Operations on Existing Servers](#)
 - [Edit settings for an existing JIRA server](#)
 - [Edit mappings for an existing JIRA server](#)
 - [Delete an existing JIRA server](#)

 JIRA is Atlassian's issue tracking product, which can be used to manage projects and associated work.


 Before you begin: Ensure that you configure your JIRA instance to [enable sub-tasks](#), [enable unassigned issues](#) and [allow Remote API access](#) . The instructions on this page have been tested with JIRA 3.13.4.

JIRA issues can be viewed in the main Dashboard view in Crucible. This requires you to enter details on the required JIRA server(s) via the Crucible administration screens.

Opening the Administration Screen for JIRA Integration

To set up JIRA integration, open the Administration screen and then click '**JIRA Servers**' under the '**Global Settings**' sub-menu on the left navigation bar. The '**View JIRA Servers**' administration page opens.

Screenshot: The View JIRA Servers Page

View JIRA Servers 								
Name	Url	Subtask ID	Subtask Resolution Action ID	Subtask Resolution ID	Allow Unassigned	Activity Stream Provider	Trusted Application	
jac	https://jira.atlassian.com	6	2	6	true	true	false	Edit Mappings Delete
Add a JIRA Server Edit default JIRA Server mappings								

On the View JIRA Servers page, you can carry out a number of operations as listed on this page.

Adding a New JIRA Server

To add a new JIRA server from the View JIRA Servers page, click '**Add JIRA Server**'.

The '**Add JIRA Server**' page opens.

Add JIRA Server

Server Details

Name:
URL:

Default Subtask Settings (leave blank to disable subtasks)

Subtask Type ID:
Subtask Resolution Action ID:
Subtask Resolution ID:

Allow Unassigned:
☒ Yes
☐ No

Default Authentication

Username:
Password:

Options

☐ Include in Activity Streams
☐ Authenticate as Trusted Application

Test
Save
Cancel


Screenshot: The Add JIRA Server Page

A number of fields and options must be filled out or selected on this page. See the table below for information on each field.

Option	Type	Description	Required
Name	Text Field	A descriptive name for the JIRA server.	Yes
URL	Text Field	The Internet address of the JIRA server.	Yes
Subtask Type ID	Number	This is required to enable creating issues from a Crucible comment.	No
Subtask Resolution Action ID	Number	This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.	No
Subtask Resolution ID	Number	This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.	No
Allow Unassigned	True/False Button	Allow unassigned sub-tasks.	No
Username	Text Field	The username of an account on the JIRA instance (All activity that takes place will be attributed to this user, unless using the Trusted Application setting).	Yes
Password	Text Field	The password for the account on the JIRA instance.	Yes
Include in Activity Streams	Check Box	Allows JIRA information to appear on the Dashboard. (Requires a JIRA instance with the streams plugins installed)	No
Authenticate as Trusted Application	Check Box	Allows the system to interface with JIRA and let users log on with their own accounts (and use their own accounts on the JIRA server. See complete FishEye documentation and complete JIRA documentation .	No

Once you've filled out the necessary fields, click '**Test**' to ensure that your details are correct. If you have a positive message return from the test, click '**Save**'.

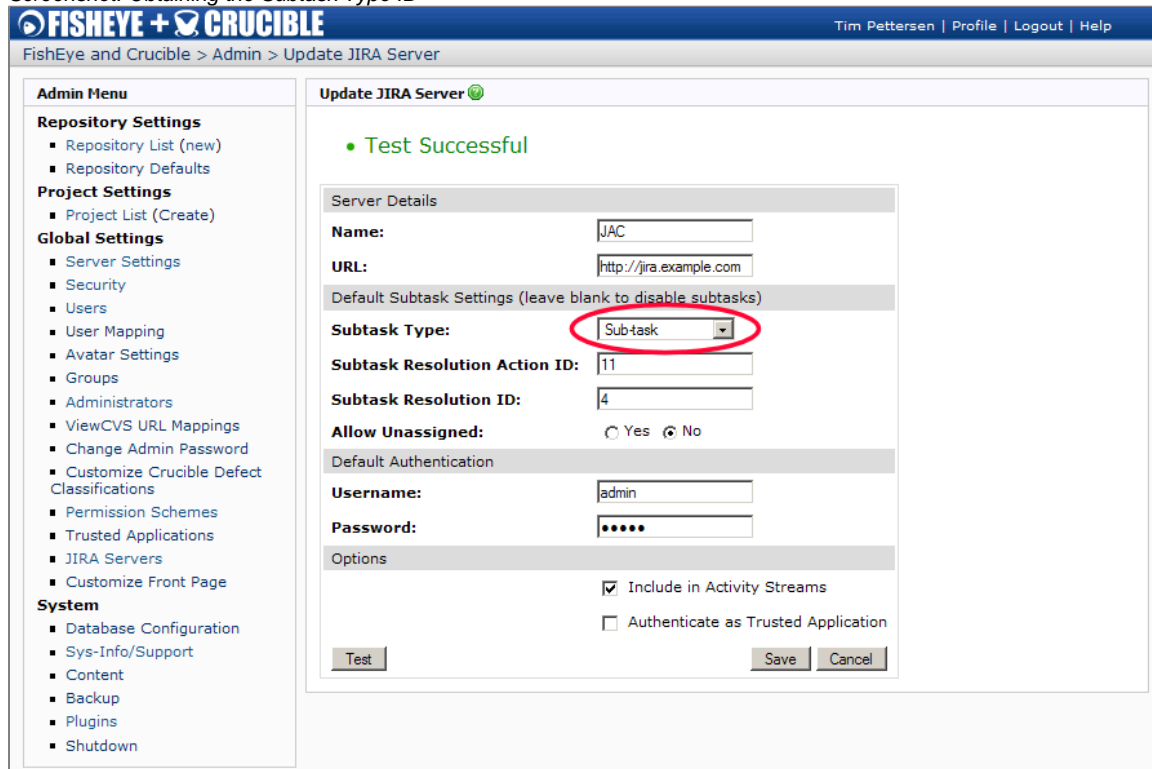
Obtaining the Subtask Type ID

 This value is required (along with the Subtask Resolution ID and Subtask Resolution Action ID) to enable creating issues from a Crucible comment. This is the subtask type that will be created when you create a JIRA subtask in Crucible.

To obtain this value, carry out the following steps.

1. Enable sub-tasks on your JIRA instance from the '**JIRA Administration**' > '**Sub-Tasks**' page. See the [JIRA documentation](#) for details on this step.
2. Return to the FishEye Administration screen and then click '**JIRA Servers**' under the '**Global Settings**' sub-menu on the left navigation bar. Click '**Edit**' next to the JIRA server you have configured.
3. Your JIRA server's basic details should appear. Click '**Test**' once again. The field for Subtask Type ID will change to a drop-down menu, showing the available subtask types. Choose the correct one.
4. Save your Crucible configuration settings.

Screenshot: Obtaining the Subtask Type ID



Obtaining the Subtask Resolution ID and Subtask Resolution Action ID

 These values are required (along with the Subtask Type ID) to enable creating issues from a Crucible comment.

To obtain these values, carry out the following steps.

1. Open your JIRA instance and go to '**Administration**' > '**Workflows**'. The '**Workflows**' screen opens. By default, the '**JIRA**' workflow is shown on screen in a table.
2. Click the '**Steps**' link in the far right table cell. The '**View Workflow Steps — JIRA**' page opens.
3. The '**Subtask Resolution Action ID**' is in the '**Open**' row, under the '**Transitions**' column. Look at the link in that cell named '**Resolve Issue**'. The ID number is shown in brackets next to that heading '**Resolve Issue**' (shown in the screenshot below as 5).
4. Save your Crucible configuration settings.
5. The '**Subtask Resolution ID**' is the '**Resolved ID**' on this page. The ID number is shown in brackets next to the heading '**Resolved**' (shown in the screenshot below as '4'). Note it down and enter it into the Crucible configuration screen.
6. Save your Crucible configuration settings.

Screenshot: Obtaining the Subtask Resolution ID & Subtask Resolution Action ID

View Workflow Steps — jira

This shows all of the steps for **jira**.

Not editable because workflow is **Active**.

☐ View all [workflows](#).
☐ View all [statuses](#).

Step Name (id)	Linked Status	Transitions (id)	Operations
Open (1)	Open	Start Progress (4) >> In Progress Resolve Issue (5) (circled) >> Resolved Close Issue (2) >> Closed	View Properties
In Progress (3)	In Progress	Stop Progress (301) >> Open Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	View Properties
Resolved (4) (circled)	Resolved	Close Issue (701) >> Closed Reopen Issue (3) >> Reopened	View Properties
Reopened (5)	Reopened	Resolve Issue (5) >> Resolved Close Issue (2) >> Closed Start Progress (4) >> In Progress	View Properties
Closed (6)	Closed	Reopen Issue (3) >> Reopened	View Properties

Editing Default JIRA Server Mappings

This setting enables the Crucible feature that shows JIRA information in a dynamic window when you hover the mouse over a JIRA issue key in Crucible. It will also turn every issue key into a hyperlink to that issue in Crucible.

To enable this feature, click **'Edit Default JIRA Server Mappings'** from the View JIRA Servers page. The **'Map JIRA Project Default'** page opens.

Screenshot: The Default JIRA Server Mappings Page

Map JIRA Project Default

Default Mappings for JIRA Projects

Default mappings for JIRA servers

Choose Fisheye Repository: CLOV [add all](#)

Selected Repositories:

- CLOV

Choose Crucible Project: TEST [add all](#)

Selected Crucible Projects:


- POTATO
- TEST

[Save](#) [Cancel](#)

On this page, select the FishEye repositories or Crucible Projects that you wish to associate with all the JIRA servers you have configured for use

in Crucible. You can click **'add all'** to quickly include them all in this category. You can remove individual items by clicking the small 'X' marks.


Once you've finished, click **'Save'**.

 You should disable any existing Crucible [linkers](#) you have set up for JIRA, as they will override this feature and prevent the dynamic dialog box from appearing when you mouse over an issue.

Operations on Existing Servers

Once you have configured an existing JIRA server, there are three main operations you can carry out on it: **'Edit'**, **'Mappings'** and **'Delete'**. These options appear on the far right of the screen.

Screenshot: Operations in the JIRA Servers Page

View JIRA Servers 								
Name	Url	Subtask ID	Subtask Resolution Action ID	Subtask Resolution ID	Allow Unassigned	Activity Stream Provider	Trusted Application	
jac	https://jira.atlassian.com	6	2	6	true	true	false	Edit Mappings Delete
Add a JIRA Server Edit default JIRA Server mappings								

Edit settings for an existing JIRA server

When you click **'Edit'**, you can adjust any of the general settings you configured when you first added the server.

Edit mappings for an existing JIRA server

When you click **'Mappings'**, a page is loaded that is almost identical to the **'Default Mapping'** screen, but allows you to choose mappings only for that specific JIRA server.

Delete an existing JIRA server

Clicking **'Delete'** will remove the server from the list.

Migrating to an External Database

This page contains instructions on migrating your Crucible database from its default embedded form to an external database.

On this page:

- [Overview](#)
- [Migrating to MySQL](#)
- [Migrating to PostgreSQL](#)

Overview

As of release 2.0, Crucible and FishEye offer alternatives to the built-in HSQLDB database for storing its relational data. At the time of writing, MySQL and PostgreSQL are fully supported. This page outlines the steps required for switching to an external database.

Migrating to MySQL

To switch from the built-in HSQLDB database to MySQL, install MySQL and follow the steps below.

1. Download the MySQL JDBC driver jar file from the [MySQL website](#) and copy the jar to your `FISHEYE_INST/lib` directory (create the `lib/` directory if it doesn't already exist). Restart FishEye or Crucible to have it pick up the driver.
2. Create a UTF-8 Database:

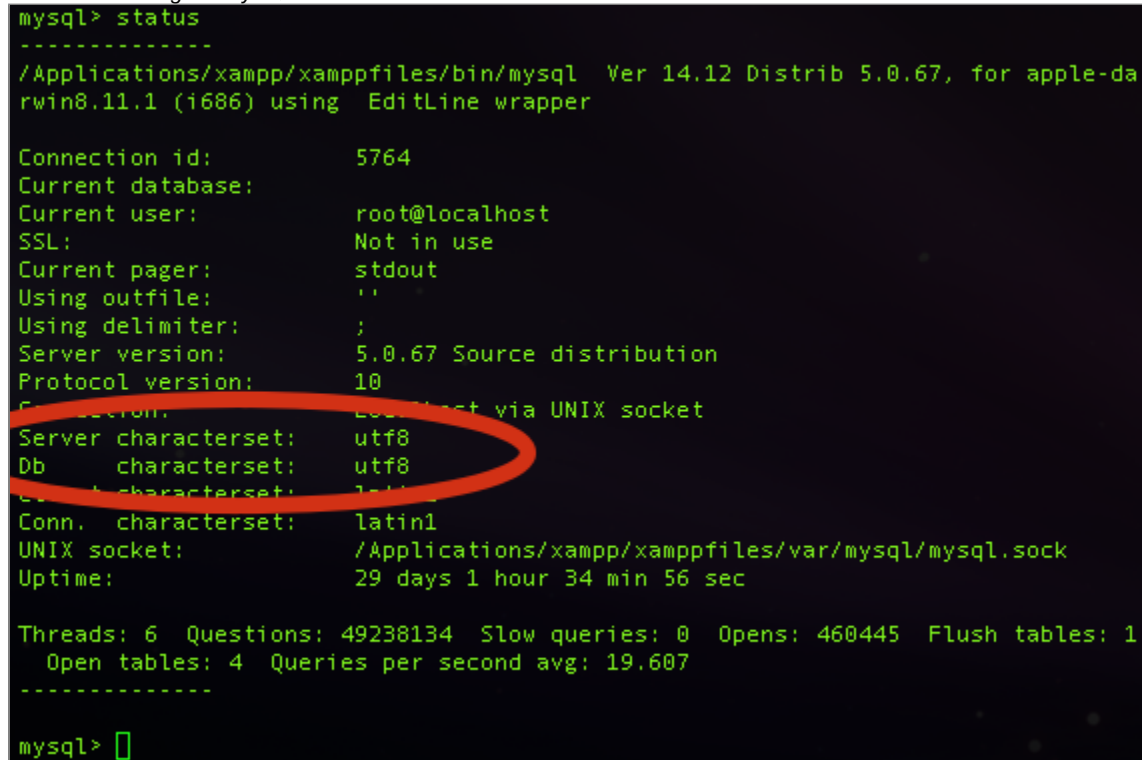
```
CREATE DATABASE crucible CHARACTER SET utf8 COLLATE utf8_bin;
```

3. You will also need to set the `Server CharacterSet` to `utf8`. This can be done by adding the following in `my.ini` for Windows or `my.cnf` for other OS. It has to be declared in the `Server` section, which is the section after `[mysqld]`:

```
[mysqld]  
default-character-set=utf8
```

4. Use the **status** command to verify database character encoding information:

Screenshot: Using the MySQL Status Command



```
mysql> status  
-----  
/Applications/xampp/xamppfiles/bin/mysql Ver 14.12 Distrib 5.0.67, for apple-da  
rwin8.11.1 (i686) using EditLine wrapper  
  
Connection id:          5764  
Current database:  
Current user:           root@localhost  
SSL:                    Not in use  
Current pager:          stdout  
Using outfile:           ''  
Using delimiter:        ;  
Server version:         5.0.67 Source distribution  
Protocol version:       10  
Connection:              Localhost via UNIX socket  
Server characterset:     utf8  
Db characterset:         utf8  
Collation:               latin1_swedish_ci  
Conn. characterset:      latin1  
UNIX socket:             /Applications/xampp/xamppfiles/var/mysql/mysql.sock  
Uptime:                  29 days 1 hour 34 min 56 sec  
  
Threads: 6 Questions: 49238134 Slow queries: 0 Opens: 460445 Flush tables: 1  
Open tables: 4 Queries per second avg: 19.607  
-----  
  
mysql> 
```

5. Create a user that can log in from the host that Crucible or FishEye is running on and make sure that the user has full access to the newly created database. In particular, the user should be allowed to create and drop tables, indexes and other constraints.

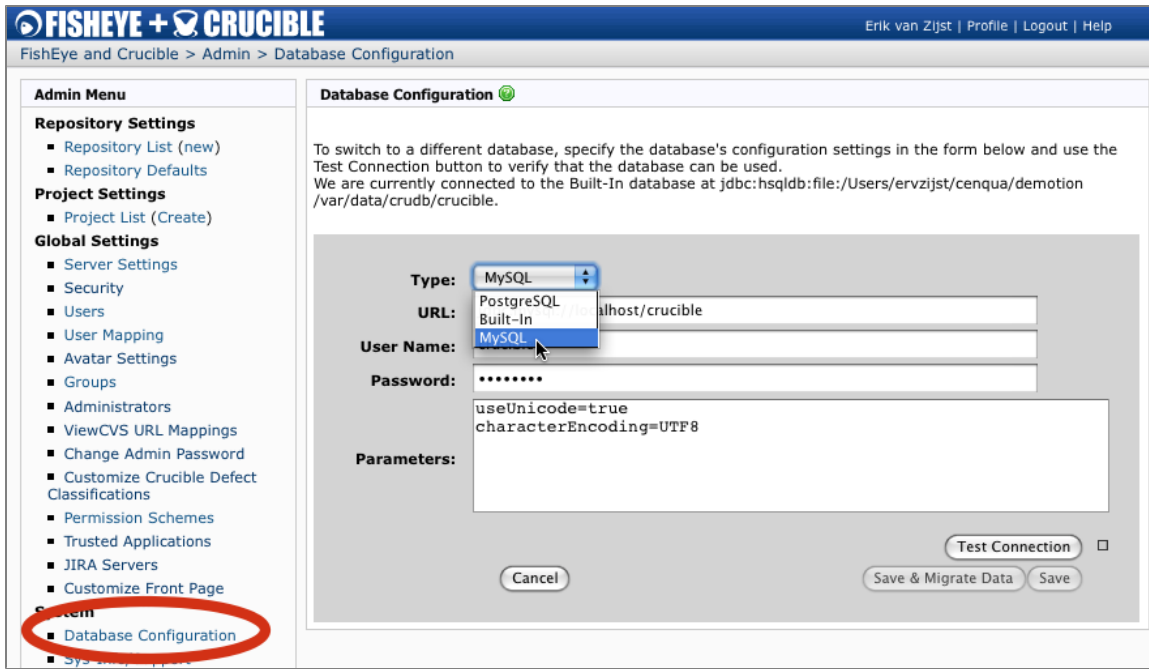
For instance, when Crucible and MySQL run on the same machine (accessible through `localhost`), issue the following commands (replacing username and password with the appropriate values):

```
mysql> grant all on crucible.* to 'username'@'localhost' identified by 'password';  
Query OK, 0 rows affected (0.00 sec)  
  
mysql> flush privileges;  
Query OK, 0 rows affected (0.01 sec)
```

6. With the database prepared, navigate to the '**Database Configuration**' section in the admin interface, select MySQL from the drop down and fill out the database URL, username and password.

Then click '**Test Connection**' to verify that Crucible or FishEye can log in to the database:

Screenshot: Testing the Connection

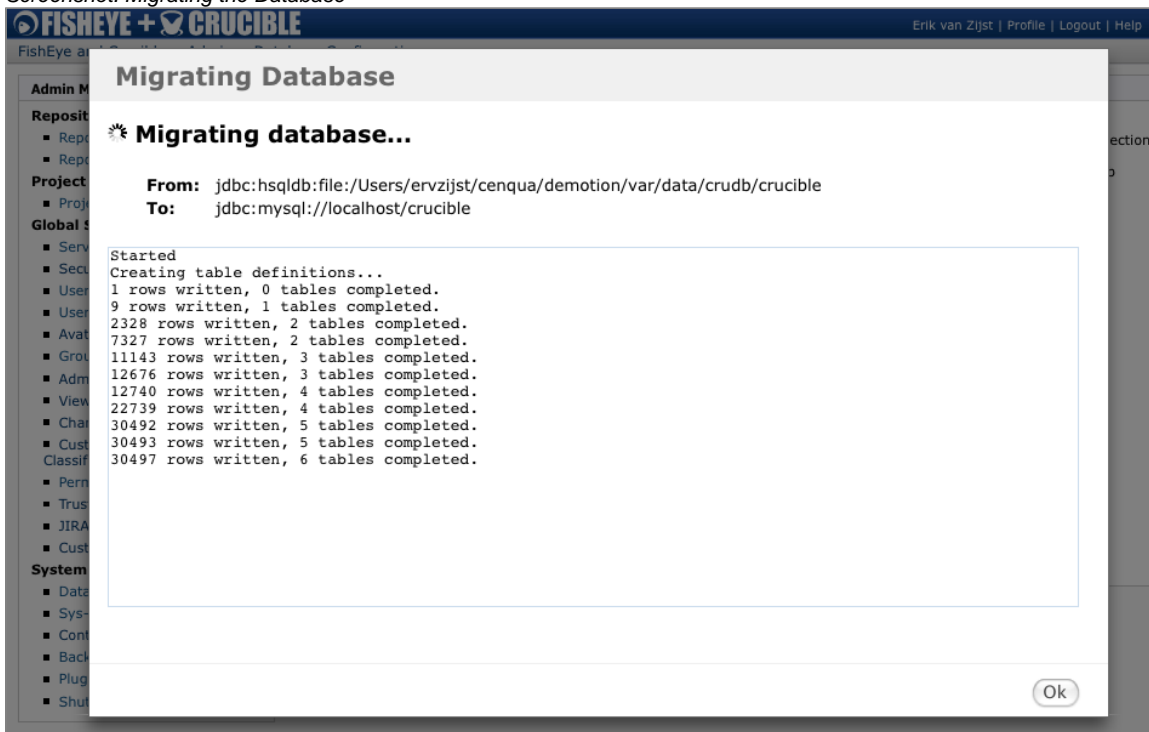


If this fails, verify that you have the MySQL JDBC driver .jar file in the classpath (by placing the .jar file in `FISHEYE_INST/lib`). Also, ensure that the database user can log in to the database from the machine that Crucible or FishEye is running on and that all the required privileges are present.

7. Click '**Save & Migrate Data**' to start the migration process.

During the migration process (which will take several minutes, depending on the size of your database and network throughput), the product will be inaccessible to users and external API clients. Users will see a maintenance screen that informs them of the process. Should the migration fail for any reason, Crucible will not switch to the new database and report on the encountered problems. Because the destination database may now contain some, but not yet all data, drop all tables, indexes and constraints before attempting a new migration.

Screenshot: Migrating the Database



Migrating to PostgreSQL

To switch from the built-in HSQLDB database to PostgreSQL, install PostgreSQL and follow the steps below.

1. Download the PostgreSQL JDBC driver jar file from the [PostgreSQL website](#) and copy the jar to your `FISHEYE_INST/lib` directory (create the `lib/` directory if it doesn't already exist). Restart FishEye or Crucible to have it pick up the driver.

2. Create a new database user (replacing 'username' and 'password' with the appropriate values):

```
$ psql
> create user username password 'password';
```

3. Create a UTF-8 database and make the newly created user the owner:

```
> create database crucible ENCODING 'UTF-8' OWNER username;
```

4. Make sure the user has full access to the database:

```
> grant all on database crucible to username;
```

During the migration process (which will take several minutes, depending on the size of your database and network throughput), the product will be inaccessible to users and external API clients. Users will see a maintenance screen that informs them of the process. Should the migration fail for any reason, Crucible will not switch to the new database and report on the encountered problems. Because the destination database may now contain some, but not yet all data, drop all tables, indexes and constraints before attempting a new migration.

Trusted Applications

This page contains information about trusted application support in Crucible and how you can configure a trusted application relationship between Crucible and JIRA or Confluence.

On this page:

- [Adding a Trusted Application](#)
 - [Configuring Identification Settings](#)
 - [URL field](#)
 - [Id field](#)
 - [Configuring Access Permissions](#)
 - [URL Patterns field](#)
 - [IP Address Patterns field](#)
 - [Certificate Timeout field](#)
- [Editing Trusted Application Settings](#)

A 'trusted application' is an application that can access specific functions in Crucible, on behalf of any user — without the user logging in to Crucible.

 Crucible and FishEye share the same trusted applications; an application trusted by FishEye is also trusted by Crucible. At this time, JIRA 3.12 and Confluence 2.7 onwards can be configured as trusted applications.



Before you begin, note that configuring a trusted application requires the transmission of sensitive data. To prevent 'man-in-the-middle attacks', it is recommended that you use an [encrypted SSL connection](#) while configuring a trusted application.

Adding a Trusted Application

To add a trusted application to Crucible:

1. Access the '**Administration Screen**'.
2. Click '**Trusted Applications**' under '**Global Settings**' on the left navigation bar.
3. Click '**Add a Trusted Application**'. The '**Trusted Application**' screen opens.

Screenshot: Configuring Trusted Applications

On this page, there are two areas, the '**Identification**' area and the '**Access Permissions**' area.

Configuring Identification Settings

Under the '**Identification**' heading, there are two fields, '**URL**' and '**Id**'.

URL field

In this field is where you will enter the Trusted Application Public Key URL of the application you wish to trust. For example, if your application's base URL is;

'http://www.mycompany/jira/'

you would enter that into the URL field. Once you've done this, click the '**Get ID**' button. Crucible will then retrieve the Trust Certificate Id from the other application and display it in the Id field. If this step fails, you may not have not entered the correct URL for the other application.

Id field

This field contains the **Trust Certificate ID**, once you have filled out the URL field correctly (see above) and clicked the '**Get ID**' button. The contents of this field are not editable.

Configuring Access Permissions

Under the **Access Permissions** heading, there are three fields, **URL Patterns**, **IP Address Patterns** and **Certificate Timeout**. These allow you to further restrict requests from a trusted application.

URL Patterns field


With this field, you can limit the access a trusted application has to Crucible. It is not necessary to specify anything for this field; in fact a blank value is a sensible default. The default behaviour is no restriction.

The text that you specify should not include your hostname, IP address or port number, rather it relates to folders on the server, that start with the text you provide.

For example, if you use this setting:

/foo

then Crucible will trust only the requests to Crucible URLs starting with /foo, e.g. /foo/bar, /foobar and /foo/bar/baz/x. You can specify multiple URLs by separating them with a comma.

 URL Patterns do not support wildcard characters or regular expressions in Crucible.

IP Address Patterns field

With this field, you can limit the trusted network addresses for other applications. You can use wildcards to specify a number range, and multiple addresses can be separated with commas. For example, if you use this setting:

192.168.*.* , 127.0.0.0

then Crucible will only trust requests from machines with the IP addresses 192.168.*anything.anything*(a group of network addresses) and 127.0.0.0 (a single host). The default is no restriction.

Certificate Timeout field

With this field, you can set the number of milliseconds before the certificate times out. This feature's purpose is to prevent '[replay attacks](#)'. For example, if an attacker intercepts a request, they may attempt to extract the certificate and send it again independently. With the certificate timeout, the application will be able to tell that this is no longer a valid request. The default value is 1000 (one second).




A shorter time out is more secure, but if set too short, it may cause valid requests to be rejected on slower networks.

Once you've finished entering the settings for the Trusted Application, click the '**Save**' button to confirm and activate the trust relationship.

Editing Trusted Application Settings

Once you have configured your trusted application(s), you can view the settings on the main '**Trusted Applications**' page.

Screenshot: Trusted Applications list

Trusted Applications List 				
Name	Id	Url Patterns	IP Address Patterns	
https://extranet.atlassian.com	confluence:3427555			Edit Delete
Add a Trusted Application				

From this screen, you can click '**Edit**' to make changes to the trusted application settings, or click '**Delete**' to remove the trust relationship for that application.

Crucible Development Hub

If you're doing custom development with Crucible, you've come to the right place.

Crucible Developer Resources

- [Developer Documentation](#) - Get started developing plugins for Atlassian Products.
- [The Atlassian Developer Blog](#) - Find out more about what our developers are up to.
- [Crucible Plugin Types](#) - Learn about the different kinds of plugin technologies.
- [Live Code Examples](#) - See the source of real plugins in the wild.
- [Plugin Hosting](#) - Atlassian can host your plugin on our servers.

Documentation

Here you'll find everything you need to code up a storm with Crucible. This includes guides for setting up your environment, building a project and creating a plugin, with real-world examples you can try.

How to Build a Crucible Plugin
How to Build a Crucible Plugin - start here to learn how to set up your development environment, create a plugin template and start coding.
Development Platform for Crucible
Crucible API Javadocs
Crucible REST API
Crucible Developer FAQ

Crucible Plugin Types

Crucible plugins come in a variety of flavours, read on to see how the plugin technology interacts with the core of Crucible and what rules can be bent, or possibly *broken* in this world.

Source Code Management (SCM) Plugins
<ul style="list-style-type: none">• Crucible SCM Plugins
Event Listeners
<ul style="list-style-type: none">• Event Listener Plugins
Servlets
<ul style="list-style-type: none">• Servlet Modules

Live Code Examples

Below is a list of real-world plugin examples that showcase the various sides of Crucible development. The following items are an excellent resource for the Atlassian developer community. Feel free to investigate these examples, hack them to pieces, or use them as inspiration to really innovate.

Bundled SCM Plugins
<ul style="list-style-type: none">• Subversion SCM Plugin• Perforce SCM Plugin• Confluence SCM Plugin• File System SCM Plugin
Other SCM Plugins
<ul style="list-style-type: none">• Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)• Crucible ClearCase plugin
Servlet Examples
<ul style="list-style-type: none">• Basic Servlet Example• Crucible Reporting plugin

Plugin Hosting

Atlassian can host your plugin development project. We'll provide a Subversion repository, Confluence space and a JIRA project. [Find out more.](#)

The Atlassian Developer Blog

For up-to-date news and opinions from the Crucible, FishEye and other Atlassian development teams.

Atlassian Developer Blog



Agile Development - Per On Dogfooding and Frequent Internal Releases

Shipping a new feature-version of software every few months - instead of every few years - is great.

Agile Book Club Explained

Last year, when the engineering team leads & managers were thinking about ways to keep ourselves trained up without going old-school, we came up with the idea of an Agile Book Club. Sounds dull, but the results can be kinda cool.

Agile With A Remote Product Owner

We Are From Mars All agile methodologies stress the need of co-locating development with the customer's representative - the Product Owner - or at least, having them in close proximity -...

Make Your Code Agile: Refactoring

In this post I define and promote refactoring in productivity terms through controlling code complexity.

Help us Integrate Confluence with Alfresco

At Atlassian we're always looking for ways to expand the utility and functionality of our products. Sometimes this means we develop that code ourselves, and sometimes it means our community fills in...

Also see the [FishEye Development Hub](#).

Documentation for Crucible Development

Here you'll find everything you need to code up a storm with Crucible. This includes guides for environment set-up, building a project, plugin creation, and real-world examples you can try.

How to Build a Crucible Plugin

[How to Build a Crucible Plugin](#) - start here to learn how to set up your development environment, create a plugin template and start coding.

[Development Platform for Crucible](#)

[Crucible API Javadocs](#)

[Crucible REST API](#)

Crucible's URL Structure

This page contains information about the Crucible URL structure for plugin developers. Knowing the structure, you will be able to construct hyperlinks for use in plugins or gadgets and find API specifications for your version of Crucible.

On this page:

- [Create Review](#)
- [Crucible Reviews](#)
- [Crucible Projects](#)
- [Search Crucible Reviews](#)
- [Search Crucible Review Comments](#)



There is also a page about the [FishEye structure](#).

Create Review

This creates a Crucible review on the specified changeset and repository.

In the example below, insert the desired changeset ID in place of "**MY_CSID**" and the desired repository name in place of "**REPNAME**".

Basic form


```
/cru/create?csid=MY_CSID&repo=REPNAME
```

Example with typical values

```
http://example.com/crucible/cru/create?csid=18905&repo=CLOV
```

Crucible Reviews

This opens a Crucible review page with the specified review key.

In the example below, insert the desired review key in place of "**MY_REVIEW_KEY**".

Basic form

```
/cru/REVIEW_KEY
```

Example with typical values

```
http://example.com/crucible/cru/CR-1
```

Crucible Projects

This opens a Crucible project page with the specified project key.

In the example below, insert the desired project key in place of "**MY_PROJECT_KEY**".

Basic form

```
/cru/browse/MY_PROJECT_KEY
```

Example with typical values

```
http://example.com/crucible/cru/browse/CR-CLOV
```

Search Crucible Reviews

This searches Crucible reviews with the specified search string.

In the example below, insert the desired string that you want to match against review titles in place of "**QUERYSTRING**".

Basic form

```
/cru/search?query=QUERYSTRING
```

Example with typical values

```
http://example.com/crucible/cru/search?query=november-audit
```

Search Crucible Review Comments

This searches comments on reviews in Crucible.

In the example below, you would insert your search string in place of the word "**TEST**".

Basic form

```
/cru/commentSearch?search.text=TEST
```

Example with typical values

```
http://example.com/crucible/cru/commentSearch?search.text=imho
```

Looking for a page on the FishEye URL structure? [Click here](#).

Crucible REST API

These pages contain information relating to the REST API for Crucible.

A list of available services and a detailed example page are currently documented.

Crucible REST API documentation:

- [Crucible REST API Usage Example](#)
- [Conditional Get](#) — Conditional Get allows lightweight polling of resources.
- [Data Types](#) — Definitions of data types used by the REST API.
- [Project Service](#) — Provides access to the projects defined in a Crucible instance.
- [Repository Service](#) — Provides information about the repositories configured in a Crucible instance.
- [Review Service](#) — The Review Service allows you to list, examine, create and modify reviews.

Crucible REST API Usage Example

This page describes using the Crucible REST API to retrieve comments from reviews in Crucible. It's an overview of using the API, not a comprehensive reference.

The [Crucible REST API](#) provides a reference for all the REST operations supported by Crucible.

On this page:

- [Authentication](#)
- [Retrieving Reviews](#)
- [Retrieving Reviews in a Specific State](#)
- [Retrieving Comments From a Review](#)
- [Retrieving Properties of a File Under Review](#)
- [Creating a New Review](#)
- [JSON](#)
- [Retrieving a Specific Review](#)
- [Making a JSON Request](#)

The Crucible REST API lives under the URL <http://HOSTNAME:PORT/CONTEXT/rest-service/>, where `HOSTNAME:PORT` is the IP address and port of your FishEye instance and `CONTEXT` is the web application context it is deployed under.

This page doesn't assume any particular REST client is being used – it just discusses the URLs to use and the responses which they will give. The information returned is in XML format.

This page assumes Crucible 1.6 – the examples (in particular JSON support) may not work with earlier versions.

Authentication

Requests to the REST API are simply HTTP requests, which can use any of the normal Crucible authentication methods. An unauthenticated request will execute as the anonymous user.

Authentication options are:

- **The normal Crucible login cookie.** A cookie named 'remember' in the request with the token returned by the REST authentication service on <http://HOSTNAME:PORT/rest-service/auth-v1/login?userName=jim&password=jimspassword>. This will return `<?xml version="1.0" encoding="UTF-8" standalone="yes"?><loginResult><token>jim:2:4455f9a4387298a83aae6902e8843f89</token></loginResult>`. The value of the cookie should be set to `jim:2:4455f9a4387298a83aae6902e8843f89`.
- **Trusted Applications.** If Crucible trusts the application which is making the request, the user logged in to the trusted application will be authenticated in Crucible.
- **Crowd.** If Crucible is configured to use Crowd, then a request containing Crowd authentication will authenticate the Crowd user in Crucible.
- **Basic Authentication.** An RFC 2617 Basic Authentication header.

Retrieving Reviews

This example will use the reviews service, at the URL <http://HOSTNAME:PORT/rest-service/reviews-v1>. A simple get on this URL will return every review in the system. The results will look like this:

```
<reviews>
  <reviewData>
    <author>pmcneil</author>
    <creator>pmcneil</creator>
    <description>14699: CRUC-230: allow links to be removed
14698: CRUC-230: don't allow linking cycles</description>
    <moderator>pmcneil</moderator>
    <name>CRUC-214: Generate comment/defect and open review report graphs</name>
    <permaId>
      <id>CR-FE-1</id>
    </permaId>
    <projectKey>CR-FE</projectKey>
    <repoName>FE</repoName>
    <state>Review</state>
  </reviewData>
  ...
</reviews>
```

Retrieving Reviews in a Specific State

If you don't want to retrieve every review, you can specify a value for the `state` parameter:

<http://HOSTNAME:PORT/rest-service/reviews-v1?state=Review,Summarize> to retrieve only those reviews in particular states.

The request only returns those reviews that the authenticated user is allowed to see.

Once you have the reviews you can use their `permaId` to get more details, so:

<http://HOSTNAME:PORT/rest-service/reviews-v1/CR-FE-1> will return a single `reviewData` element, identical to the one shown above.

Retrieving Comments From a Review

URLs like <http://HOSTNAME:PORT/rest-service/reviews-v1/CR-FE-1/thing> will return information about `thing` records belonging to the review.

So <http://HOSTNAME:PORT/rest-service/reviews-v1/CR-FE-1/comments> returns all the comments in the review:

```

<comments>
  <versionedComment>
    <createDate>2008-03-03T22:22:00.920+11:00</createDate>
    <defectApproved>false</defectApproved>
    <defectRaised>false</defectRaised>
    <deleted>false</deleted>
    <draft>false</draft>
    <message>why use roll instead of add??</message>
    <permaId>
      <id>CMT:200</id>
    </permaId>
    <reviewItemId>
      <id>CFR-281</id>
    </reviewItemId>
    <user>mquail</user>
    <fromLineRange>196-199, 230-233</fromLineRange>
    <toLineRange>206-209, 240-251</toLineRange>
  </versionedComment>
  ... more versioned comments ...
  <generalComment>
    <createDate>2008-03-25T17:15:20.380+11:00</createDate>
    <defectApproved>false</defectApproved>
    <defectRaised>true</defectRaised>
    <deleted>false</deleted>
    <draft>false</draft>
    <message>when there are no comments in the last week the vertical axis shows -5 as the starting
point</message>
    <user>pmcneil</user>
  </generalComment>
  ... more general comments ...
</comments>

```

Retrieving Properties of a File Under Review

If you need more information about the file a versioned comment was on, the URL

<http://HOSTNAME:PORT/rest-service/reviews-v1/CR-FE-1/reviewitems/CFR-281> gives more details:

```

<fisheyeReviewItemData>
  <permId>
    <id>CFR-281</id>
  </permId>
  <fromPath></fromPath>
  <fromRevision></fromRevision>
  <repositoryName>FE</repositoryName>

  <toPath>branches/iteration03/src/java/com/cenqua/crucible/reports/CommentsDefects/CommentDatasetMaker.java
  <toRevision>13947</toRevision>
</fisheyeReviewItemData>

```

That particular review item is a new file, so the `fromPath` and `fromRevision` elements are empty.

Creating a New Review

To create a review, do a POST call to the reviews url (<http://HOSTNAME:PORT/rest-service/reviews-v1>) with the following XML document as request body (note that you need to be authenticated to be able to create a new review, so use Basic HTTP authentication for this call):

Request to Create a New Review

```
<?xml version="1.0"?>
<createReview>
  <reviewData>
    <author> <!-- required element -->
      <userName>joe</userName>
    </author>
    <creator> <!-- required element -->
      <userName>fred</userName>
    </creator>
    <moderator> <!-- required element -->
      <userName>erik</userName>
    </moderator>
    <description>These is the Statement of Objectives.</description>
    <name>Title of the new review</name> <!-- required element -->
    <projectKey>CR</projectKey> <!-- required element -->
    <allowReviewersToJoin>true</allowReviewersToJoin>
  </reviewData>
</createReview>
```

JSON

As of Crucible 1.6.3, JSON serialization is supported for REST requests and responses. Using the `Accept` request header, clients can specify whether the response document should be encoded in XML or JSON. Unless specified differently, Crucible will respond using XML and will interpret requests as XML. Crucible will always include the `Content-Type` header in the response to identify the encoding. Likewise, when a client sends a JSON request document, it must use the `Content-Type: application/json` header. It is possible to use a different encoding for the request and the response.



Note

JSON support is currently experimental.

Retrieving a Specific Review

To retrieve the contents of a specific review as a JSON document, rather than XML, include the `Accept: application/json` header in your HTTP request. The example below includes the HTTP headers of both the request and the response to illustrate this:

Request:

```
GET /rest-service/reviews-v1/CR-3/details HTTP/1.1
Host: localhost:8060
Authorization: Basic am9lOmpvZQ==
Accept: application/json
```

Response:

```

HTTP/1.1 200 OK
Cache-Control: private
Content-Type: application/json
Last-Modified: Sun, 26 Oct 2008 23:05:45 GMT
ETag: "1225062345005-140"

{"detailedReviewData": {
  "allowReviewersToJoin":false,
  "author":{"displayName":"Joe","userName":"joe"},
  "createDate":"2008-10-27T09:50:05.064+1100",
  "creator":{"displayName":"Joe","userName":"joe"},
  "description":"",
  "metricsVersion":1,
  "moderator":{"displayName":"Joe","userName":"joe"},
  "name":"readme ",
  "permaId":{"id":"CR-3"},
  "projectKey":"CR",
  "state":"Draft",
  "actions": {
    "actionData":[{"name":"action:deleteReview"}, {"name":"action:rejectReview"}, {"name":
"action:abandonReview"}, {"name":"action:summarizeReview"},
    { "name":"action:modifyReviewFiles"}, {"name":"action:approveReview"}, {"name":
"action:recoverReview"}, {"name":"action:commentOnReview"},
    { "name":"action:submitReview"}, {"name":"action:createReview"}, {"name":"action:viewReview"}, {
"name":"action:reopenReview"}, {"name":"action:closeReview"}]
  },
  "generalComments":"",
  "reviewItems":"",
  "reviewers":"",
  "transitions": {
    "transitionData":[{"name":"action:approveReview"}, {"name":"action:abandonReview"}]
  },
  "versionedComments":""
}}
```

Note that the response document above has been indented to increase readability in this example.

Making a JSON Request

When sending a request document using JSON, include the Content-Type: application/json header in the HTTP request. The example below creates a new review using JSON. Again, the relevant HTTP request and response headers are included:

Request:

```

POST /rest-service/reviews-v1 HTTP/1.1
Host: localhost:8060
Content-Length: 269
Authorization: Basic am9lOmpvZQ==
Accept: application/json
Content-Type: application/json

{"createReview":
  {"reviewData": {
    "allowReviewersToJoin":false,
    "author":{"userName":"joe"},
    "creator":{"userName":"joe"},
    "moderator":{"userName":"matt"},
    "description":"JSON Test Review",
    "metricsVersion":1,
    "name":"readme ",
    "projectKey":"CR"
  }}
}
```

Response:

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: http://localhost:8060/rest-service/reviews-v1/CR-12

{ "reviewData": {
  "allowReviewersToJoin": false,
  "author": { "displayName": "joe lowercase", "userName": "joe" },
  "createDate": "2008-10-27T17:21:29.779+1100",
  "creator": { "displayName": "joe lowercase", "userName": "joe" },
  "description": "JSON Test Review",
  "metricsVersion": 1,
  "moderator": { "displayName": "Matt Quail", "userName": "matt" },
  "name": "readme ",
  "permaId": { "id": "CR-12" },
  "projectKey": "CR",
  "state": "Draft"
}}
```

Conditional Get

Conditional Get allows lightweight polling of resources.

The REST API makes Crucible available to remote applications. Depending on the type of application, it can be quite common to poll a certain resource periodically to be able to detect changes. For example, an application may request `/reviews-v1/filter/Review` at regular intervals and notify the user when a new review was added.

However, polling a server constantly for potential updates could cause undesired overhead, especially when the response is large. To better facilitate applications that need to poll frequently, Crucible implements HTTP Conditional Get.

Conditional Get

With Conditional Get, the server keeps track of when the last change was made to a resource and sends this timestamp along in each HTTP response (`Last-Modified: Wed, 01 Oct 2008 03:37:58 GMT`). At the same time, a client that understands Conditional Get and polls the same resource periodically, will in turn keep track of the `Last-Modified` timestamp of each resource it has requested and send it along as a request header at every request (`If-Modified-Since: Mon, 29 Sep 2008 06:47:04 GMT`).

When the resource has not been modified since the last time the client requested it (`Last-Modified <= If-Modified-Since`), the server will not serve the request, but return status 204 "Not Modified" with an empty response body. If the resource was modified, the server will respond normally (200 with the resource in the body).

Note that Crucible also sends the `ETag` response header along with `Last-Modified`. The `ETag` header contains a checksum of the response document and allows the client to detect changes even when the `Last-Modified` time did not change. A client that implements Conditional Get should send the value of the `ETag` response header in the `If-None-Match` request header.

For more information on HTTP Conditional Get, please refer to the [HTTP specification](#).

Compatibility

Servers implementing Conditional Get are completely compatible with clients that don't understand it and vice versa.

Data Types

Definitions of data types used by the REST API.

- [ReviewData](#)
- [ReviewItemData](#)
- [DetailedReviewData](#)
- [Error](#)

ReviewData

Contains basic information about a review.

Sample XML:

```

<reviewData>
  <allowReviewersToJoin>false</allowReviewersToJoin>
  <author>
    <displayName>Matt Quail</displayName>
    <userName>matt</userName>
  </author>
  <createDate>2008-08-25T12:38:14.603+1000</createDate>
  <creator>
    <displayName>Matt Quail</displayName>
    <userName>matt</userName>
  </creator>
  <description>Review things and stuff</description>
  <metricsVersion>1</metricsVersion>
  <moderator>
    <displayName>Matt Quail</displayName>
    <userName>matt</userName>
  </moderator>
  <name>Test review 1</name><permaId>
  <id>CR-1</id></permaId>
  <projectKey>CR</projectKey>
  <state>Review</state>
</reviewData>

```

Sample JSON:

```

{ "reviewData": {
  "allowReviewersToJoin": false,
  "author": { "displayName": "Matt Quail", "userName": "matt" },
  "createDate": "2008-10-27T09:50:05.064+1100",
  "creator": { "displayName": "Matt Quail", "userName": "matt" },
  "moderator": { "displayName": "Matt Quail", "userName": "matt" },
  "description": "Review things and stuff",
  "metricsVersion": 1,
  "name": "Test review 1",
  "permaId": { "id": "CR-3" },
  "projectKey": "CR",
  "state": "Draft"
} }

```

ReviewItemData

Describes a single item that is under review. An item can represent the changes between two or more revisions of a file in a source repository, a change that was uploaded to Crucible as a *unified diff* or *patch file*, or it can represent any arbitrary file uploaded and attached to a review. Below are three examples of `reviewItemData` in XML, followed by the same three in JSON:

Sample XML:


```

<!-- ReviewItemData representing a changes between two revisions: -->
<reviewItem expand="revisions">
  <permId>
    <id>CFR-40</id>
  </permId>
  <authorName>evzijst</authorName>
  <commitDate>2008-10-14T15:25:08.755+1000</commitDate>
  <commitType>Modified</commitType>
  <fileType>File</fileType>
  <fromContentUrl>
/cru/CR-4/rawcontent/89/test2%2Ftrunk%2Fsrc%2Fmain%2Fjava%2Fcom%2Fatlassian%2FTest.java
</fromContentUrl>
    <fromPath>test2/trunk/src/main/java/com/atlassian/Test.java</fromPath>
    <fromRevision>9</fromRevision>
    <repositoryName>Local</repositoryName>
    <toContentUrl>
/cru/CR-4/rawcontent/80/test2%2Ftrunk%2Fsrc%2Fmain%2Fjava%2Fcom%2Fatlassian%2FTest.java</toContentUrl>
    <toPath>test2/trunk/src/main/java/com/atlassian/Test.java</toPath>
    <toRevision>10</toRevision>
    <revisions size="4"/>
  </reviewItem>

<!-- ReviewItemData representing a file from an uploaded unified diff: -->
<reviewItem expand="revisions">
  <permId>
    <id>CFR-38</id>
  </permId>
  <authorName>joe</authorName>
  <commitDate>2008-11-19T15:45:57.953+1100</commitDate>
  <commitType>Modified</commitType>
  <fileType>File</fileType>
  <fromPath>src/java/com/atlassian/crucible/spi/rpc/AbstractJAXBContextResolver.java</fromPath>
  <fromRevision>2:F</fromRevision>
  <patchUrl>/cru/CR-4/downloadPatch/4/CRUC-582.patch</patchUrl>
  <repositoryName>PATCH:4</repositoryName>
  <toPath>src/java/com/atlassian/crucible/spi/rpc/AbstractJAXBContextResolver.java</toPath>
  <toRevision>2:T</toRevision>
  <revisions size="2"/>
</reviewItem>

<!-- ReviewItemData representing an uploaded binary file: -->
<reviewItem expand="revisions">
  <permId>
    <id>CFR-31</id>
  </permId>
  <authorName>joe</authorName>
  <commitDate>2008-11-13T10:15:05.510+1100</commitDate>
  <commitType>Added</commitType>
  <fileType>File</fileType>
  <fromPath/>
  <fromRevision/>
  <repositoryName>UPLOAD:4</repositoryName>
  <toContentUrl>/cru/CR-4/rawcontent/52/scm-plugin.tgz</toContentUrl>
  <toPath>jackrabbit-scm-plugin.tgz</toPath>
  <toRevision>34</toRevision>
  <revisions size="1"/>
</reviewItem>

```

Sample JSON

```

{
  "reviewItem": {
    "@expand": "revisions",
    "permId": { "id": "CFR-40" },
    "authorName": "evzijst",
    "commitDate": "2008-10-14T15:25:08.755+1000",
    "commitType": "Modified",
    "fileType": "File",
    "fromContentUrl": "\/cru\/CR-4\/rawcontent\/89\/test2%2Ftrunk%2Fsrc%2Fmain%2Fjava%2Fcom%2Fatlassian%2FTest.java",
    "fromPath": "test2\/trunk\/src\/main\/java\/com\/atlassian\/Test.java",
    "fromRevision": 9,
    "repositoryName": "Local",
    "toContentUrl": "\/cru\/CR-4\/rawcontent\/80\/test2%2Ftrunk%2Fsrc%2Fmain%2Fjava%2Fcom%2Fatlassian%2FTest.java",
    "toPath": "test2\/trunk\/src\/main\/java\/com\/atlassian\/Test.java",
    "toRevision": 10,
    "revisions": { "@size": "4" }
  }
}

{
  "reviewItem": {
    "@expand": "revisions",
    "permId": { "id": "CFR-38" },
    "authorName": "joe",
    "commitDate": "2008-11-19T15:45:57.953+1100",
    "commitType": "Modified",
    "fileType": "File",
    "fromPath": "src\/java\/com\/atlassian\/crucible\/spi\/rpc\/AbstractJAXBContextResolver.java",
    "fromRevision": "2:F",
    "patchUrl": "\/cru\/CR-4\/downloadPatch\/4\/CRUC-582.patch",
    "repositoryName": "PATCH:4",
    "toPath": "src\/java\/com\/atlassian\/crucible\/spi\/rpc\/AbstractJAXBContextResolver.java",
    "toRevision": "2:T",
    "revisions": { "@size": "2" }
  }
}

{
  "reviewItem": {
    "@expand": "revisions",
    "permId": { "id": "CFR-31" },
    "authorName": "joe",
    "commitDate": "2008-11-13T10:15:05.510+1100",
    "commitType": "Added",
    "fileType": "File",
    "fromPath": "",
    "fromRevision": "",
    "repositoryName": "UPLOAD:4",
    "toContentUrl": "\/cru\/CR-4\/rawcontent\/52\/jackrabbit-scm-plugin.tgz",
    "toPath": "jackrabbit-scm-plugin.tgz",
    "toRevision": 34,
    "revisions": { "@size": "1" }
  }
}

```

The above section contains three `reviewItemData` instances that illustrate the use of the individual elements. The main elements are the `<to../>` and `<from../>` elements. These describe the two revisions of a review item where the `to..` is the most recent and `from..` the oldest revision of the item that is under review.

When using *Iterative Reviewing*, an item can contain more than two file revisions. The `<revisions/>` element contains information on every revision under review, while `<to../>` and `<from../>` always point to the first and the last revisions (the cumulative changes). By default, the `<revisions/>` element is collapsed and contains the `size=` attribute that indicates the total number of file revisions in the review item. To expand this list, use the `?append=revisions` url parameter, e.g.:

```
http://localhost:6060/crucible/rest-service/reviews-v1/CR-FE-2033/reviewitems/CFR-23489?expand=revisions
```

Note that when a new file is added, it will not have the `from..` elements and likewise, when a file gets removed, it will lack the `to..` elements. The `<to../>` and `<from../>` elements also include urls that point to the file contents hosted in Crucible. These are the `<fromContentUrl/>` and `<toContentUrl/>` elements. These are relative URLs that come after the web application context, so for example to download the file from the third `reviewItemData`, access: `http://HOSTNAME:PORT/CONTEXT/cru/CR-4/rawcontent/52/scm-plugin.tgz`.

Note that `<fromContentUrl/>` and `<toContentUrl/>` only apply to either uploaded files or revisions on files in one of the Crucible repositories. Uploaded patch files lack these elements because a unified diff file usually only contains the sections of two files that were changed, but not the code that was unchanged. As a result, Crucible is unable to provide links for the individual files. Instead, the `<patchUrl/>` element contains a relative link to the original patch file that was uploaded by the creator of the review.

DetailedReviewData

Note that the `reviewItems` element is empty when multiple reviews are retrieved via REST. To include the `reviewItems` in a `detailedReviewData` structure you must retrieve a single review via the URL `/rest-service/reviews-v1/<review id>/details`.

Sample XML:

```
<detailedReviewData>
  <allowReviewersToJoin>false</allowReviewersToJoin>
  <author>
    <displayName>Matt Quail</displayName>
    <userName>matt</userName>
  </author>
  <createDate>2008-09-16T10:50:26.862+1000</createDate>
  <creator>
    <displayName>Matt Quail</displayName>
    <userName>matt</userName>
  </creator>
  <description/>
  <metricsVersion>1</metricsVersion>
  <moderator>
    <displayName>Matt Quail</displayName>
    <userName>matt</userName>
  </moderator>
  <name/>
  <permaId>
    <id>CR-1</id>
  </permaId>
  <projectKey>CR</projectKey>
  <state>Draft</state>
  <actions>
    <actionData>
      <name>action:abandonReview</name>
    </actionData>
    <actionData>
      <name>action:closeReview</name>
    </actionData>
    <actionData>
      <name>action:submitReview</name>
    </actionData>
    <actionData>
      <name>action:reopenReview</name>
    </actionData>
    <actionData>
      <name>action:summarizeReview</name>
    </actionData>
    <actionData>
      <name>action:rejectReview</name>
    </actionData>
    <actionData>
      <name>action:deleteReview</name>
    </actionData>
    <actionData>
      <name>action:approveReview</name>
    </actionData>
    <actionData>
      <name>action:modifyReviewFiles</name>
    </actionData>
    <actionData>
      <name>action:viewReview</name>
    </actionData>
    <actionData>
      <name>action:commentOnReview</name>
    </actionData>
  </actions>
</detailedReviewData>
```

```
<actionData>
  <name>action:recoverReview</name>
</actionData>
<actionData>
  <name>action:createReview</name>
</actionData>
</actions>
<generalComments/>
<reviewItems>
  <reviewItem>
    <permId>
      <id>CFR-1</id>
    </permId>
    <authorName>admin</authorName>
    <commitDate>2008-08-27T10:19:17.000+1000</commitDate>
    <commitType>Modified</commitType>
    <fileType>File</fileType>
    <fromPath>ds/Home</fromPath>
    <fromRevision>1</fromRevision>
    <repositoryName>localhost</repositoryName>
    <toPath>ds/Home</toPath>
    <toRevision>2</toRevision>
  </reviewItem>
  <reviewItem>
    <permId>
      <id>CFR-2</id>
    </permId>
    <authorName>tomd</authorName>
    <commitDate>2008-09-09T16:42:28.786+1000</commitDate>
    <commitType>Added</commitType>
    <fileType>File</fileType>
    <fromPath/>
    <fromRevision/>
    <repositoryName>mylocalsvn</repositoryName>
    <toPath>aaa/bbb/qqq.txt</toPath>
    <toRevision>3</toRevision>
  </reviewItem>
</reviewItems>
<reviewers/>
<transitions>
  <transitionData>
    <name>action:approveReview</name>
  </transitionData>
  <transitionData>
    <name>action:abandonReview</name>
  </transitionData>
</transitions>
```

```
<versionedComments/>
</detailedReviewData>
```

Sample JSON:

```
{
  "detailedReviewData": {
    "allowReviewersToJoin": false,
    "author": { "displayName": "joe lowercase", "userName": "joe" },
    "createDate": "2008-10-27T09:50:05.064+1100",
    "creator": { "displayName": "joe lowercase", "userName": "joe" },
    "description": "",
    "metricsVersion": 1,
    "moderator": { "displayName": "joe lowercase", "userName": "joe" },
    "name": "readme ",
    "permId": { "id": "CR-3" },
    "projectKey": "CR",
    "state": "Draft",
    "actions": {
      "actionData": [
        { "name": "action:rejectReview" }, { "name": "action:closeReview" },
        { "name": "action:modifyReviewFiles" }, { "name": "action:abandonReview" },
        { "name": "action:commentOnReview" }, { "name": "action:reopenReview" },
        { "name": "action:createReview" }, { "name": "action:recoverReview" },
        { "name": "action:deleteReview" }, { "name": "action:approveReview" },
        { "name": "action:viewReview" }, { "name": "action:submitReview" },
        { "name": "action:summarizeReview" }
      ]
    },
    "generalComments": "",
    "reviewItems": { "reviewItem": [
      {
        "permId": { "id": "CFR-1" },
        "authorName": "evzijst",
        "commitDate": "2008-10-14T15:25:08.755+1000",
        "commitType": "Modified",
        "fileType": "File",
        "fromPath": "test2\\trunk\\src\\main\\java\\com\\atlassian\\Test.java",
        "fromRevision": 9,
        "repositoryName": "Local",
        "toPath": "test2\\trunk\\src\\main\\java\\com\\atlassian\\Test.java",
        "toRevision": 10,
        "permId": { "id": "CFR-2" },
        "authorName": "evzijst",
        "commitDate": "2008-10-14T15:25:08.755+1000",
        "commitType": "Added",
        "fileType": "Directory",
        "fromPath": "",
        "fromRevision": "",
        "repositoryName": "Local",
        "toPath": "test2\\trunk\\src\\test\\java\\com",
        "toRevision": 10
      }
    ]
  },
  "reviewers": "",
  "transitions": { "transitionData": [
    { "name": "action:approveReview" },
    { "name": "action:abandonReview" }
  ]
},
  "versionedComments": ""
}
```

Error

When a request cannot be serviced properly due to either a server-side problem, or invalid client input, Crucible will return an error document, combined with an HTTP status code other than 200. This XML document contains a number of elements that describe the problem. Note that the HTTP status code distinguishes between client- and server-side causes.

Below is the error that is returned when asking for a non-existent resource. The status code for this response is 404 "Document Not Found". Other possible status codes for error responses include 400 "Bad Request" (for example when a request contains an invalid POST body) and 403 "Forbidden" (when accessing a resource without permission).

Sample XML:

```
<error>
  <code>NotFound</code>
  <message>Unknown metrics version: 45</message>
  <stacktrace>com.atlassian.crucible.spi.services.NotFoundException: Unknown metrics version: 45
    at com.atlassian.crucible.spi.impl.DefaultReviewService.getMetrics(DefaultReviewService.java:689)
    at com.atlassian.crucible.spi.rpc.RestReviewService$22.doGet(RestReviewService.java:645)
    at com.atlassian.crucible.spi.rpc.RestReviewService$22.doGet(RestReviewService.java:644)
    at com.atlassian.crucible.spi.rpc.ConditionalGet.doConditionalGet(ConditionalGet.java:46)
    at com.atlassian.crucible.spi.rpc.RestReviewService.getMetrics(RestReviewService.java:643)
    ...
  </stacktrace>
</error>
```

Sample JSON:

```
{
  "code": "NotFound",
  "message": "No review exists with permId 'CR-333'",
  "stacktrace": "com.atlassian.crucible.spi.services.NotFoundException: Unknown metrics version: 45
    at com.atlassian.crucible.spi.impl.DefaultReviewService.getMetrics(DefaultReviewService.java:689)
    at com.atlassian.crucible.spi.rpc.RestReviewService$22.doGet(RestReviewService.java:645)
    at com.atlassian.crucible.spi.rpc.RestReviewService$22.doGet(RestReviewService.java:644)
    at com.atlassian.crucible.spi.rpc.ConditionalGet.doConditionalGet(ConditionalGet.java:46)
    at com.atlassian.crucible.spi.rpc.RestReviewService.getMetrics(RestReviewService.java:643)
    ..."
}
```

Project Service

Provides access to the projects defined in a Crucible instance. At present this interface is read-only.

Get Project List

Method: GET

URL:

```
/projects-v1
```

Description:

Returns a list of projects.

Status Code:

200 (OK) on success

Example XML:

```
<projects>
  <projectData>
    <allowReviewersToJoin>>false</allowReviewersToJoin>
    <id>1</id>
    <key>CR</key>
    <name>Default Project</name>
    <permissionSchemeId>1</permissionSchemeId>
  </projectData>
  <projectData>
    ...
  </projectData>
</projects>
```

Example JSON:

```
{ "projects": { "projectData": {  
  "allowReviewersToJoin": false,  
  "id": 1,  
  "key": "CR",  
  "name": "Default Project",  
  "permissionSchemeId": 1}  
}}
```

Repository Service

Provides information about the repositories configured in a Crucible instance.

Get Repositories

Method: GET

URL:

```
/repositories-v1
```

Description:

Get a list of all the repositories.

Status Code:

200 (OK) on success.

Example XML:

```
<repositories>  
  <repoData>  
    <enabled>true</enabled>  
    <name>local</name>  
    <type>svn</type>  
  </repoData>  
  <repoData>  
    <enabled>true</enabled>  
    <name>test</name>  
    <type>cvs</type>  
  </repoData>  
</repositories>
```

Example JSON:

```
{ "repositories": {  
  "repoData": [  
    { "@type": "svnRepositoryData",  
      "enabled": true,  
      "name": "Local",  
      "type": "svn",  
      "path": "",  
      "url": "file:\\\\\\Users\\erik\\var\\repo" },  
    { "@type": "svnRepositoryData",  
      "enabled": true,  
      "name": "Local2",  
      "type": "svn",  
      "path": "",  
      "url": "file:\\\\\\Users\\erik\\var\\repo" } ]  
}}
```

Get Repository By Name

Method: GET

URL:

```
/repositories-v1/<repositoryName>
```

Description:

Returns the repository of which the `name` attribute equals `repositoryName`.

Status Code:

200 (OK) on success

Example XML:

```
<svnRepositoryData>
  <enabled>true</enabled>
  <name>local</name>
  <type>svn</type>
  <path></path>
  <url>file:///Users/tomd/dev/svn/</url>
</svnRepositoryData>
```

Example JSON:

```
{ "svnRepositoryData": {
  "enabled": true,
  "name": "Local",
  "type": "svn",
  "path": "",
  "url": "file:///Users/ervzijst/var/repo"
}}
```

Review Service

The Review Service allows you to list, examine, create and modify reviews.

See the [Data Types](#) Page for the structure of the `reviewData` and `detailedReviewData` tags.

Click an item in the list below to see full details, all options and example code.

- [Review Service - Reviews](#)
 - [Add Changeset To Review](#)
 - [Add Patch Revisions To Review](#)
 - [Create Review](#)
 - [Delete Review](#)
 - [Get All Reviews](#)
 - [Get All Reviews \(limited\)](#)
 - [Get Allowed Review Actions](#)
 - [Get Allowed Review Transitions](#)
 - [Get Review](#)
 - [Get Review Details](#)
 - [Get Review Details by Path](#)
 - [Get Review Details through Custom Filter Criteria](#)
 - [Get Reviews by Filter](#)
 - [Get Reviews by Path](#)
 - [Get Reviews through Custom Filter Criteria](#)
 - [Get Single Review Details](#)

- Get Version Info
- Review Service - Reviewers
 - Add Reviewers
 - Get Finished Reviewers
 - Get Incomplete Reviewers
 - Get Reviewers
 - Remove Single Reviewer
- Review Service - Review Items
 - Add Revision to Review
 - Get Review Items
 - Get Single Revision Details
 - Remove Revision from Review
- Review Service - Workflow
 - Close a Review
 - Complete a Review
 - Move a Review to a New State
 - Uncomplete a Review
- Review Service - Comments
 - Add a Reply to a Comment
 - Add Comment to a Review Item
 - Add General Comment to a Review
 - Delete a Comment
 - Delete a Reply
 - Get a Comment
 - Get Comments on a Review Item
 - Get Comments on Files
 - Get General Comments
 - Get Review Comments
 - Get the Replies to a Comment
 - Mark a Comment as Read
 - Mark a Comment as Leave Unread
 - Mark All Comments as Read
 - Publish a Draft Comment
 - Publish All Draft Comments
 - Update a Comment
- Review Service - Miscellaneous
 - Get Metrics

Review Service - Comments

On this page:

- Comments
 - Add a Reply to a Comment
 - Add Comment to a Review Item
 - Add General Comment to a Review
 - Delete a Comment
 - Delete a Reply
 - Get a Comment
 - Get Comments on a Review Item
 - Get Comments on Files
 - Get General Comments
 - Get Review Comments
 - Mark a Comment as Read
 - Mark a Comment as Leave Unread
 - Mark all Comments as Read
 - Get the Replies to a Comment
 - Publish a Draft Comment
 - Publish All Draft Comments
 - Update a Comment

Comments

Add a Reply to a Comment

URL:

```
POST /reviews-v1/<review id>/comments/<comment id>/replies
```

Description:

Add a reply to an existing comment.

The POST data is a `generalCommentData` structure.

Status Code:

201 (Created) on success. The response will contain the `Location` response header with the URL of the newly created resource.

Add Comment to a Review Item**URL:**

```
POST /reviews-v1/<review id>/reviewitems/<review item id>/comments
```

Description:

Add a comment to a review. Returns the completed `versionedLineCommentData` structure.

The POST data is a `versionedLineCommentData` structure.

Status Code:

201 (Created) on success. The response will contain the `Location` response header with the URL of the newly created resource.

Add General Comment to a Review**URL:**

```
POST /reviews-v1/<review id>/comments
```

Description:

Add a general comment to a review. Returns the completed `generalCommentData` structure.

The POST data is a `generalCommentData` structure.

Status Code:

201 (Created) on success. The response will contain the `Location` response header with the URL of the newly created resource.

Delete a Comment**URL:**

```
DELETE /reviews-v1/<review id>/comments/<comment id>
```

Description:

Remove an existing comment.

Status Code:

204 (No Content) on success.

Delete a Reply**URL:**

```
DELETE /reviews-v1/<review id>/comments/<comment id>/replies/<reply id>
```

Description:

Delete a reply.

Status Code:

204 (No Content) on success.

Get a Comment

URL:

```
GET /reviews-v1/<review id>/comments/<comment id>
```

Description:

Retrieve an existing comment

Status Code:

200 (OK) on success.

Comment Read Status:

This attribute details the read status of the comment for the particular user that the request has been made for. There are three possible values:

- UNREAD: The comment has not been read.
- READ: The comment has been viewed.
- LEAVE_UNREAD: The comment has been read but marked by the user to be left for later referral.

Anonymous access requests and users who do not have permission to comment on a review always return the READ value.

Example XML Return Data:

```
<comments>
  <versionedLineCommentData>
    ...
  </versionedLineCommentData>
</comments>
```

Example JSON Return Data:

```
{ "comments": [
  { "versionedCommentData": ... }
]}
```

Get Comments on a Review Item**URL:**

```
GET /reviews-v1/<review id>/reviewitems/<review item id>/comments
```

Description:

Get all the comments made on a review item.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<comments>
  <versionedLineCommentData>
    ...
  </versionedLineCommentData>
  ...
</comments>
```

Example JSON Return Data:

```
{ "comments": [
  { "versionedCommentData": ... }
  { "versionedCommentData": ... }
]}
```

Get Comments on Files

URL:

```
GET /reviews-v1/<review id>/comments/versioned
```

Description:

Get all the versioned comments made on a review – that is, comments which are on a particular file in the review.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<comments>
  <versionedLineCommentData>
    ...
  </versionedLineCommentData>
  ...
</comments>
```

Example JSON Return Data:

```
{ "comments": [
  { "versionedCommentData": ... }
  { "versionedCommentData": ... }
]}
```

Get General Comments

URL:

```
GET /reviews-v1/<review id>/comments/general
```

Description:

Get all the general comments made on a review – that is, comments which are not attached to a particular file in the review.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<comments>
  <generalCommentData>
    ...
  </generalCommentData>
</comments>
```

Example JSON Return Data:

```
{
  "comments": [
    {
      "generalCommentData": {
        "createDate": "2008-10-30T16:32:51.032+1100",
        "defectApproved": false,
        "defectRaised": true,
        "deleted": false,
        "draft": false,
        "readStatus": "UNREAD",
        "message": "A general comment.",
        "metrics": [
          {
            "entry": {
              "key": "classification",
              "value": [
                {
                  "configVersion": 1,
                  "Not conforming to standards"
                }
              ]
            },
            {
              "key": "rank",
              "value": [
                {
                  "configVersion": 1,
                  "Minor"
                }
              ]
            }
          ]
        },
        "permIdAsString": "CMT:1",
        "replies": "",
        "user": {
          "displayName": "joe lowercase",
          "userName": "joe",
          "permId": {
            "id": "CMT:1"
          }
        },
        "generalCommentData": {
          "createDate": "2008-11-03T10:26:50.951+1100",
          "defectApproved": false,
          "defectRaised": false,
          "deleted": false,
          "draft": false,
          "message": "Another general comment.",
          "metrics": "",
          "permIdAsString": "CMT:4",
          "replies": "",
          "user": {
            "displayName": "joe lowercase",
            "userName": "joe",
            "permId": {
              "id": "CMT:4"
            }
          }
        }
      }
    }
  ]
}
```

Get Review Comments

URL:

```
GET /reviews-v1/<review id>/comments
```

Description:

Get all the comments made on a review. The `versionedLineCommentData` tag may contain `fromLineRange` and `toLineRange` tags, indicating that the comment was made against a specific range of lines.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<comments>
  <generalCommentData>
    <createDate>2008-09-16T16:28:02.833+1000</createDate>
    <defectApproved>>false</defectApproved>
    <defectRaised>>false</defectRaised>
    <deleted>>false</deleted>
    <draft>>false</draft>
    <readStatus>READ</readStatus>
    <message>This is a general comment.</message>
    <metrics/>
    <permIdAsString>CMT:1</permIdAsString>
    <replies/>
    <user>
      <displayName>Matt Quail</displayName>
      <userName>matt</userName>
    </user>
    <permId>
      <id>CMT:1</id>
    </permId>
  </generalCommentData>
  <versionedLineCommentData>
```

```
<createDate>2008-09-16T16:28:26.432+1000</createDate>
<defectApproved>false</defectApproved>
<defectRaised>true</defectRaised>
<deleted>false</deleted>
<draft>false</draft>
  <readStatus>LEAVE_READ</readStatus>
<message>This is a revision level defect.</message>
<metrics>
  <entry>
    <key>classification</key>
    <value>
      <configVersion>1</configVersion>
      <value>Inconsistent</value>
    </value>
  </entry>
  <entry>
    <key>rank</key>
    <value>
      <configVersion>1</configVersion>
      <value>Major</value>
    </value>
  </entry>
</metrics>
<permaIdAsString>CMT:2</permaIdAsString>
<replies/>
<user>
  <displayName>Matt Quail</displayName>
  <userName>matt</userName>
</user>
<permaId>
  <id>CMT:2</id>
</permaId>
<reviewItemId>
  <id>CFR-4</id>
</reviewItemId>
</versionedLineCommentData>
<versionedLineCommentData>
  <createDate>2008-09-16T16:28:54.604+1000</createDate>
  <defectApproved>false</defectApproved>
  <defectRaised>false</defectRaised>
  <deleted>false</deleted>
  <draft>false</draft>
    <readStatus>UNREAD</readStatus>
  <message>This is a comment covering two lines of a revision.</message>
  <metrics/>
  <permaIdAsString>CMT:3</permaIdAsString>
  <replies/>
  <user>
    <displayName>Matt Quail</displayName>
    <userName>matt</userName>
  </user>
  <permaId>
    <id>CMT:3</id>
  </permaId>
  <reviewItemId>
    <id>CFR-4</id>
  </reviewItemId>
  <fromLineRange>2-3</fromLineRange>
  <toLineRange>3-4</toLineRange>
```

```
</versionedLineCommentData>
</comments>
```

Example JSON Return Data:

```
{ "comments": {
  "generalCommentData": {
    "createDate": "2008-10-30T16:32:51.032+1100",
    "defectApproved": false,
    "defectRaised": true,
    "deleted": false,
    "draft": false,
    "readStatus": "READ",
    "message": "A general comment.",
    "metrics": [
      { "entry": { "key": "classification", "value": { "configVersion": 1, "Not conforming to standards" } },
      { "entry": { "key": "rank", "value": { "configVersion": 1, "Minor" } } },
    ],
    "permaIdAsString": "CMT:1",
    "replies": "",
    "user": { "displayName": "joe lowercase", "userName": "joe" },
    "permId": { "id": "CMT:1" },
    "versionedLineCommentData": [
      {
        "createDate": "2008-10-30T16:33:02.726+1100",
        "defectApproved": false,
        "defectRaised": false,
        "deleted": false,
        "draft": false,
        "readStatus": "LEAVE_READ",
        "message": "This is wrong",
        "metrics": "",
        "permaIdAsString": "CMT:2",
        "replies": "",
        "user": { "displayName": "joe lowercase", "userName": "joe" },
        "permId": { "id": "CMT:2" },
        "reviewItemId": { "id": "CFR-4" },
        "toLineRange": "1-2",
      },
      {
        "createDate": "2008-10-30T16:34:12.535+1100",
        "defectApproved": false,
        "defectRaised": false,
        "deleted": false,
        "draft": false,
        "readStatus": "UNREAD",
        "message": "This is a revision level defect.",
        "metrics": "",
        "permaIdAsString": "CMT:3",
        "replies": "",
        "user": { "displayName": "joe lowercase", "userName": "joe" },
        "permId": { "id": "CMT:3" },
        "reviewItemId": { "id": "CFR-5" }
      }
    ]
  }
}
```

Mark a Comment as Read

URL:

```
POST /reviews-v1/<review id>/comments/<comment id>/markAsRead
```

Description:

Marks a particular comment as read.

Status Code:

200 (OK) on success.

Mark a Comment as Leave Unread

URL:

```
POST /reviews-v1/<review id>/comments/<comment id>/markAsLeaveUnread
```

Description:

Marks a particular comment as leave unread.

Status Code:

200 (OK) on success.

Mark all Comments as Read

URL:

```
POST /reviews-v1/<review id>/comments/markAllAsRead
```

Description:

Marks all comments in the review which are in an *unread* state as *read*. Any comments which are in the *leave unread* state are not modified.

Status Code:

200 (OK) on success.

Get the Replies to a Comment

URL:

```
GET /reviews-v1/<review id>/comments/<comment id>/replies
```

Description:

Get the replies to an existing comment.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<comments>
  <generalCommentData>
    ...
  </generalCommentData>
  ...
</comments>
```

Example JSON Return Data:

```
{ "comments": [
  { "generalCommentData": ... }
  { "generalCommentData": ... }
]}
```

Publish a Draft Comment

```
POST /reviews-v1/<review id>/publish/<comment id>
```


Description:

Publish a draft comments on the review.

Status Code:

200 (OK) on success.

Publish All Draft Comments

```
POST /reviews-v1/<review id>/publish
```

Description:

Publish all the user's draft comments on the review.

Status Code:

200 (OK) on success.

Update a Comment**URL:**

```
POST /reviews-v1/<review id>/comments/<comment id>
```

Description:

Update an existing comment. The `readStatus` attribute is ignored when updating a comment.

The POST data is a `generalCommentData` structure.

Status Code:

200 (OK) on success.

Review Service - Miscellaneous

On this page:

- [Miscellaneous](#)
 - [Get Metrics](#)

Miscellaneous***Get Metrics*****URL:**

```
GET /reviews-v1/metrics/<version>
```

Description:

Get the replies to an existing comment.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<?xml version="1.0"?>
<metrics>
  <metricsData>
    <configVersion>1</configVersion>
    <defaultValue>
      <name>Minor</name>
      <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">0</value>
    </defaultValue>
```

```

<label>Ranking</label>
<name>rank</name>
<type>INTEGER</type>
<values>
  <name>Major</name>
  <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">1</value>
</values>
<values>
  <name>Minor</name>
  <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">0</value>
</values>
</metricsData>
<metricsData>
  <configVersion>1</configVersion>
  <defaultValue>
    <name>Improvement desirable</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">5</value>
  </defaultValue>
  <label>Classification</label>
  <name>classification</name>
  <type>INTEGER</type>
  <values>
    <name>Missing</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">1</value>
  </values>
  <values>
    <name>Extra (superfluous)</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">2</value>
  </values>
  <values>
    <name>Ambiguous</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">3</value>
  </values>
  <values>
    <name>Inconsistent</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">4</value>
  </values>
  <values>
    <name>Improvement desirable</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">5</value>
  </values>
  <values>
    <name>Not conforming to standards</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">6</value>
  </values>
  <values>
    <name>Risk-prone</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">7</value>
  </values>
  <values>
    <name>Factually incorrect</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">8</value>
  </values>
  <values>
    <name>Not implementable</name>
    <value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">9</value>
  </values>
</values>

```

```
<name>Editorial</name>
<value xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xs=
"http://www.w3.org/2001/XMLSchema" xsi:type="xs:int">10</value>
</values>
```

```
</metricsData>
</metrics>
```

Example JSON Return Data:

```
{ "metrics": { "metricsData": [
  { "configVersion": 1,
    "defaultValue": { "name": "Minor", "value": 0 },
    "label": "Ranking",
    "name": "rank",
    "type": "INTEGER",
    "values": [
      { "name": "Major", "value": 1 },
      { "name": "Minor", "value": 0 }
    ]
  },
  { "configVersion": 1,
    "defaultValue": { "name": "Improvement desirable", "value": 5 },
    "label": "Classification",
    "name": "classification",
    "type": "INTEGER",
    "values": [
      { "name": "Missing", "value": 1 },
      { "name": "Extra (superfluous)", "value": 2 },
      { "name": "Ambiguous", "value": 3 },
      { "name": "Inconsistent", "value": 4 },
      { "name": "Improvement desirable", "value": 5 },
      { "name": "Not conforming to standards", "value": 6 },
      { "name": "Risk-prone", "value": 7 },
      { "name": "Factually incorrect", "value": 8 },
      { "name": "Not implementable", "value": 9 },
      { "name": "Editorial", "value": 10 }
    ]
  }
]
}
```

Review Service - Reviewers

On this page:

- [Reviewers](#)
 - [Add Reviewers](#)
 - [Get Finished Reviewers](#)
 - [Get Incomplete Reviewers](#)
 - [Get Reviewers](#)
 - [Remove Single Reviewer](#)

Reviewers

Add Reviewers

URL:

```
POST /reviews-v1/<review id>/reviewers
```

Description:

Add new reviewers to the review. Send a string of comma separated user names.

Status Code:

200 (OK) on success.

Get Finished Reviewers

URL:

```
GET /reviews-v1/<review id>/reviewers/completed
```

Description:

Return a list of the reviewers who have completed the review.

Status Code:

200 (OK) on success.

Example Return Data:

Return value as `/reviews-v1/<review id>/reviewers`, but only completed reviewers are included.

Get Incomplete Reviewers**URL:**

```
GET /reviews-v1/<review id>/reviewers/uncompleted
```

Description:

Return a list of the reviewers who have not yet completed the review.

Status Code:

200 (OK) on success.

Example Return Data:

Return value as `/reviews-v1/<review id>/reviewers`, but only incomplete reviewers are included.

Get Reviewers**URL:**

```
GET /reviews-v1/<review id>/reviewers
```

Description:

Return a list of the reviewers participating in the review.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviewers>
  <reviewer>
    <displayName>Conor MacNeill</displayName>
    <userName>conor</userName>
    <completed>>false</completed>
  </reviewer>
  ... more reviewers ...
</reviewers>
```

Example JSON Return Data:

```
{ "reviewers": { "reviewer": [
  { "displayName": "Peter Moore", "userName": "pete", "completed": false },
  { "displayName": "Brendan Humphreys", "userName": "brendan", "completed": false } ]
}}
```

Remove Single Reviewer

URL:

```
DELETE /reviews-v1/<review id>/reviewers/<username>
```

Description:

Remove a reviewer from a review.

Status Code:

204 (No Content) on success.

Review Service - Review Items

On this page:

- [Review Items](#)
 - [Add Revision to Review](#)
 - [Get Review Items](#)
 - [Get Single Revision Details](#)
 - [Remove Revision from Review](#)

Review Items

Add Revision to Review

URL:

```
POST /reviews-v1/<review id>/reviewitems
```

Description:

Add a revision to a review. Send a `reviewItem` with the repository name, and from and to paths and revisions specified. Other values can be omitted. This returns the completed `reviewItem` structure.

Status Code:

201 (Created) on success. The response will contain the `Location` response header with the URL of the newly created resource and the `reviewItemData` document in the response body.

Get Review Items

URL:

```
GET /reviews-v1/<review id>/reviewitems
```

Description:

Get a list of the items in a review.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviewItems>
  <reviewItem>
    <permId>
      <id>CFR-1</id>
    </permId>
    <authorName>tomd</authorName>
    <commitDate>2008-01-29T14:41:43.202+1100</commitDate>
    <commitType>Modified</commitType>
    <fileType>File</fileType>
    <toContentUrl>/cru/CR-4/rawcontent/53/foo.txt</toContentUrl>
    <fromPath>foo.txt</fromPath>
    <fromRevision>21</fromRevision>
    <repositoryName>local</repositoryName>
    <toContentUrl>/cru/CR-4/rawcontent/51/foo.txt</toContentUrl>
    <toPath>foo.txt</toPath>
    <toRevision>22</toRevision>
  </reviewItem>
  ... more reviewItems ...
</reviewItems>
```

Example JSON Return Data:

```
{ "reviewItems": { "reviewItem": [
  { "permId": { "id": "CFR-4" },
    "authorName": "ervzijst",
    "commitDate": "2008-10-16T17:19:52.119+1000",
    "commitType": "Modified",
    "fileType": "File",
    "fromContentUrl": "\/cru\/CR-4\/rawcontent\/53\/foo.txt",
    "fromPath": "path\/to\/file.txt",
    "fromRevision": "3",
    "repositoryName": "Local",
    "toContentUrl": "\/cru\/CR-4\/rawcontent\/51\/foo.txt",
    "toPath": "path\/to\/file.txt",
    "toRevision": "13"
  },
  ... more reviewItems ...
]
}}
```

Get Single Revision Details

URL:

```
GET /reviews-v1/<review id>/reviewitems/<review item id>
```

Description:

Get the details of a single revision in a review. Returns the `reviewItem` structure for the item.

Status Code:

200 (OK) on success.

Remove Revision from Review

URL:

```
DELETE /reviews-v1/<review id>/reviewitems/<review item id>
```

Description:

Remove a revision from a review.

Status Code:

204 (No Content) on success.

Review Service - Reviews

On this page:

- [Reviews](#)
 - [Add Changeset To Review](#)
 - [Add Patch Revisions To Review](#)
 - [Upload Files To Review](#)
 - [Create Review](#)
 - [Delete Review](#)
 - [Get All Reviews](#)
 - [Get All Reviews \(limited\)](#)
 - [Get Allowed Review Actions](#)
 - [Get Allowed Review Transitions](#)
 - [Get Review](#)
 - [Get Review Details](#)
 - [Get Review Details by Path](#)
 - [Get Review Details through Custom Filter Criteria](#)
 - [Get Reviews by Filter](#)
 - [Get Reviews by Path](#)
 - [Get Reviews through Custom Filter Criteria](#)
 - [Get Single Review Details](#)
 - [Get Version Info](#)

Reviews**Add Changeset To Review****URL:**

```
POST /reviews-v1/<review id>/addChangeset
```

Description:

Add the revisions in a set of changesets to an existing review.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<addChangeset>
  <repository>aRepositoryName</repository>
  <changesets>
    <changesetData>
      <id>...the id...</id>
    </changesetData>
    ... more change sets ...
  </changesets>
</addChangeset>
```

Example JSON Return Data:

```
{ "addChangeset": {
  "repository": "Local",
  "changesets": {
    "changesetData": [
      { "id": 234 }, { "id": 237 }, ...more change sets...
    ]
  }
}
```

The response is the reviewData structure of the review.

Add Patch Revisions To Review

URL:

```
POST /reviews-v1/<review id>/addPatch
```

Description:

Add the revisions in a patch to an existing review.

Status Code:

200 (OK) on success.

Example XML Request Data:

```
<addPatch>
  <repository>aRepositoryName</repository>
  <patch>
    <![CDATA[
... text of patch goes here ...
]]>
  </patch>
</addPatch>
```

Example JSON Request Data:

```
{ "addPatch": { "repository": "aRepositoryName", "patch": "... text of patch goes here ..." } }
```

The response is the `reviewData` structure of the review.

Upload Files To Review

URL:

```
POST /reviews-v1/<review id>/addFile
```

Description:

Uploads a local file to the review. In contrast to a patch, files can be either binary or text. Depending on the filetype, size and contents, Crucible may be able to display either parts, or the entire file in the review. It is possible to upload two versions of the file, in which case Crucible will display a diff and report that the file was modified. When only a single file is uploaded, Crucible treats the file as newly added.

This action returns the `ReviewData` document on success.

This resources uses [multipart form-data](#) to receive the file(s), character set indication and optional comments (it does not expect an XML document with embedded files, as that would require the client to first encode the files in Base64). Making a multipart form-data request can be done manually, but you will probably want to use a library. During testing it is inconvenient to let your browser generate the requests using the test html form below:

Example HTML Form for Testing REST-Based File Uploads

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head><title>Rest File Upload Test</title></head>
<body>
  <!-- Point the URL to the review you want to upload to. -->
  <form action="http://hostname:port/rest-service/reviews-v1/<review id>/addFile" enctype=
"multipart/form-data" method="POST">
    <table>
      <tr>
        <td>File (required):</td>
        <td><input name="file" type="file"/></td>
      </tr>
      <tr>
        <td>Diff to (optional):</td>
        <td><input name="diffFile" type="file"/></td>
      </tr>
      <tr>
        <td>Character Set (optional):</td>
        <td><input name="charset" type="text" value="UTF-8"/></td>
      </tr>
      <tr>
        <td>Comments (optional):</td>
        <td><input type="text" name="comments"/></td>
      </tr>
      <tr><td><input type="submit" value="Upload"/></td></tr>
    </table>
  </form>
</body>
</html>
```

The form requests understands the following 4 fields:

- file - the file to add to the review (required)
- diffFile - if supplied, Crucible will use this file as the base for a diff with file
- charset - if supplied, specifies the character set of the (text)file (when omitted, the server's default character set will be used)
- comments - optional user string that is stored along with the file

When uploading files, make sure you (or your http client library) supplies the proper Content-Type header. For text files, use "text/plain". Crucible will preserve the original file name.

Status Code:

201 (Created) on success.

The response is the `reviewItemData` structure describing the new item. Also, the `Location` response header is present and contains the `PermlId` URL of the new item.

Create Review

URL:

```
POST /reviews-v1
```

Description:

Create a review. A review can be created in one of three ways(see examples).

Status Code:

201 (Created) on success. The response will contain the `Location` response header with the URL of the newly created resource.

Example XML Request Data:

1. An empty review, i.e. no revisions in the review. This uses the request below:

XML
<pre> <createReview> <reviewData> ... </reviewData> </createReview> </pre>
JSON
<pre> {createReview: {"reviewData": { "allowReviewersToJoin":false, "author":{"userName":"joe"}, "creator":{"userName":"joe"}, "moderator":{"userName":"matt"}, "description":"Review objectives", "metricsVersion":1, "name":"Stuff to review", "projectKey":"CR" }}} </pre>

2. A patch review, containing diffs from a patch file, e.g. created by `svn diff >patch.txt` (note that the text of the patch must be properly escaped in the JSON object):

XML
<pre> <createReview> <reviewData> ... </reviewData> <patch> <![CDATA[... text of patch goes here ...]]> </patch> </createReview> </pre>
JSON
<pre> {createReview: {"reviewData": { "allowReviewersToJoin":false, "author":{"userName":"joe"}, "creator":{"userName":"joe"}, "moderator":{"userName":"matt"}, "description":""," "metricsVersion":1, "name":"readme ", "projectKey":"CR"}, "patch":"... text of patch goes here ..."} } </pre>

3. A review containing revisions from a set of changesets. XML/JSON:

XML

```
<createReview>
  <reviewData>
    ...
  </reviewData>
  <changesets>
    <changesetData>
      <id>...the id...</id>
    </changesetData>
    ... more changesets ...
  </changesets>
</createReview>
```

JSON

```
{createReview: {"reviewData": {
  "allowReviewersToJoin":false,
  "author":{"userName":"joe"},
  "creator":{"userName":"joe"},
  "moderator":{"userName":"matt"},
  "description":"",
  "metricsVersion":1,
  "name":"readme ",
  "projectKey":"CR"},
  "changesets": {
    "changesetData":[{"id":"234"},
    {"id":"237"}], "repository":"Local"
  }
}}
```

In all these cases, the reviewData structure shouldn't have the permaId or state attributes set. The response is a reviewData structure fully populated.]

Delete Review

URL:

```
DELETE /reviews-v1/<id>
```

Description:

Delete a review. The review must have been abandoned.

Status Code:

204 (No Content) on success.

Get All Reviews

URL:

```
GET /reviews-v1?state=<states>
```

Description:

Get all reviews as a list of ReviewData structures. Note that this may return a lot of data, so using /reviews-v1/filter/<filter> (see below) is usually better. The state parameter is a comma separated list of state names from the set Draft, Approval, Review, Summarize, Closed, Dead, Rejected, Unknown.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviews>
  <reviewData>
    ...
  </reviewData>
  ...
</reviews>
```

Example JSON Return Data:

```
{ "reviews": {
  "reviewData": [
    { "allowReviewersToJoin": true,
      ...
    }, ... ]
  }
}
```

Get All Reviews (limited)

URL:

```
GET /reviews-v1/details?state=<states>
```

Description:

Get all reviews as a list of DetailedReviewData structures. The `state` parameter is a comma separated list of state names from the set Draft, Approval, Review, Summarize, Closed, Dead, Rejected, Unknown.

Note that the `reviewItems` list in the `detailedReviewData` elements will not appear because this URL retrieves multiple reviews.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<detailedReviews>
  <detailedReviewData>
    ...
  </detailedReviewData>
  ...
</detailedReviews>
```

Example JSON Return Data:

```
{ "detailedReviews": {
  "detailedReviewData": [
    { "allowReviewersToJoin": true,
      ...
    }, ... ]
  }
}
```

Get Allowed Review Actions

URL:

```
GET /reviews-v1/<id>/actions
```

Description:

Get a list of the actions which the current user is allowed to perform on the review. This shows actions the user has *permission* to perform - the review may not be in a suitable state for all these actions to be performed.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<actions>
  <actionData>
    <name>action:summarizeReview</name>
  </actionData>
  <actionData>
    <name>action:viewReview</name>
  </actionData>
  <actionData>
    <name>action:approveReview</name>
  </actionData>
  <actionData>
    <name>action:closeReview</name>
  </actionData>
  <actionData>
    <name>action:modifyReviewFiles</name>
  </actionData>
  <actionData>
    <name>action:rejectReview</name>
  </actionData>
  <actionData>
    <name>action:deleteReview</name>
  </actionData>
  <actionData>
    <name>action:createReview</name>
  </actionData>
  <actionData>
    <name>action:recoverReview</name>
  </actionData>
  <actionData>
    <name>action:commentOnReview</name>
  </actionData>
  <actionData>
    <name>action:reopenReview</name>
  </actionData>
  <actionData>
    <name>action:abandonReview</name>
  </actionData>
  <actionData>
    <name>action:submitReview</name>
  </actionData>
</actions>
```

Example JSON Return Data:

```
{
  "actions": {
    "actionData": [
      { "name": "action:modifyReviewFiles" },
      { "name": "action:recoverReview" },
      { "name": "action:createReview" },
      { "name": "action:rejectReview" },
      { "name": "action:deleteReview" },
      { "name": "action:abandonReview" },
      { "name": "action:closeReview" },
      { "name": "action:reopenReview" },
      { "name": "action:approveReview" },
      { "name": "action:submitReview" },
      { "name": "action:summarizeReview" },
      { "name": "action:viewReview" },
      { "name": "action:commentOnReview" }
    ]
  }
}
```

Get Allowed Review Transitions

URL:

```
GET /reviews-v1/<id>/transitions
```

Description:

Get a list of the actions which the current user can perform on this review, given its current state and the user's permissions.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<transitions>
  <transitionData>
    <name>action:summarizeReview</name>
  </transitionData>
  <transitionData>
    <name>action:abandonReview</name>
  </transitionData>
</transitions>
```

Example JSON Return Data:

```
{
  "transitions": {
    "transitionData": [
      { "name": "action:approveReview" },
      { "name": "action:abandonReview" }
    ]
  }
}
```

Get Review

URL:

```
GET /reviews-v1/<id>
```

Description:

Get a single review.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviewData>
  ...
</reviewData>
```

Example JSON Return Data:

```
{ "reviewData": {
  "allowReviewersToJoin": false,
  ...
}}
```

Get Review Details**URL:**

```
GET /reviews-v1/filter/<filter>/details
```

Description:

Get details of all the reviews which match the given filter. See above for filter names.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<detailedReviews>
  <detailedReviewData>
    ...
  </detailedReviewData>
  ...
</detailedReviews>
```

Example JSON Return Data:

```
{ "detailedReviews": {
  "detailedReviewData": [
    { "allowReviewersToJoin": true,
      ...
    }, ... ]
  }
}
```

Get Review Details by Path**URL:**


```
GET /reviews-v1/search/<repository>/details?path=<path>
```

Description:

Return a list of Review details which include a particular file. The `path` parameter must be the full path name of a file in `repository`, with **no leading slash**.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<detailedReviews>
  <detailedReviewData>
    ...
  </detailedReviewData>
  ...
</detailedReviews>
```

Example JSON Return Data:

```
{ "detailedReviews": {
  "detailedReviewData": [
    { "allowReviewersToJoin": true,
      ...
    }, ... ]
  }
}
```

Get Review Details through Custom Filter Criteria**URL:**

```
GET /reviews-v1/filter/details?title=..&author=..&moderator=..&creator=..&reviewer=..&orRoles=true|false&complete=true|false&allReviewersComplete=true|false&project=..
```

Description:

Get details of all the reviews which match the specified filter criteria. Criteria are supplied as normal query parameters in the URL.

Filter criteria are:

- `title` - Reviews whose title contain this substring.
- `author` - Reviews authored by this user.
- `moderator` - Reviews moderated by this user.
- `creator` - Reviews created by this user.
- `reviewer` - Reviews reviewed by this user.
- `orRoles` - Whether to the value for `author`, `creator`, `moderator` and `reviewer` should be combined using OR (`orRoles=true`) or AND (`orRoles=false`).
- `complete` - Reviews that the specified reviewer has completed.
- `allReviewersComplete` - Reviews that all reviewers have completed.
- `project` - Reviews for the specified project.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviews>
  <reviewData>
    ...
  </reviewData>
  ...
</reviews>
```

Example JSON Return Data:

```
{ "reviews": {
  "reviewData": [
    { "allowReviewersToJoin": true,
      ...
    }, ... ]
  }
}
```

Get Reviews by Filter

URL:

```
GET /reviews-v1/filter/<filter>
```

Description:

Get all the reviews which match the given filter, for the current user.

Filter names are:

- allReviews - All reviews for everyone.
- allOpenReviews - Open reviews for everyone.
- allClosedReviews - Closed reviews for everyone.
- draftReviews - Draft reviews for everyone.
- toReview - Reviews on which the current user is an uncompleted reviewer.
- requireMyApproval - Reviews waiting to be approved by the current user.
- toSummarize - Completed reviews which are ready for the current user to summarize.
- outForReview - Reviews with uncompleted reviewers, on which the current reviewer is the moderator.
- drafts - Draft reviews created by the current user.
- open - Open reviews created by the current user.
- closed - Closed reviews created by the current user.
- trash - Abandoned reviews created by the current user.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviews>
  <reviewData>
    ...
  </reviewData>
  ...
</reviews>
```

Example JSON Return Data:

```
{
  "reviews": {
    "reviewData": [
      { "allowReviewersToJoin": true,
        ...
      }, ... ]
    }
  }
}
```

Get Reviews by Path

URL:

```
GET /reviews-v1/search/<repository>?path=<path>
```

Description:

Return a list of Reviews which include a particular file. The `path` parameter must be the full path name of a file in `repository`, with **no leading slash**.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviews>
  <reviewData>
    ...
  </reviewData>
  ...
</reviews>
```

Example JSON Return Data:

```
{
  "reviews": {
    "reviewData": [
      { "allowReviewersToJoin": true,
        ...
      }, ... ]
    }
  }
}
```

Get Reviews through Custom Filter Criteria

URL:

```
GET /reviews-v1/filter?title=..&author=..&moderator=..&creator=..&reviewer=..&orRoles=true|false
&complete=true|false&allReviewersComplete=true|false&project=..
```

Description:

Get all the reviews which match the specified filter criteria. Criteria are supplied as normal query parameters in the URL.

Filter criteria are:

- `title` - Reviews whose title contain this substring.
- `author` - Reviews authored by this user.
- `moderator` - Reviews moderated by this user.
- `creator` - Reviews created by this user.

- reviewer - Reviews reviewed by this user.
- orRoles - Whether to the value for author, creator, moderator and reviewer should be combined using OR (orRoles=true) or AND (orRoles=false).
- complete - Reviews that the specified reviewer has completed.
- allReviewersComplete - Reviews that all reviewers have completed.
- project - Reviews for the specified project.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<reviews>
  <reviewData>
    ...
  </reviewData>
  ...
</reviews>
```

Example JSON Return Data:

```
{ "reviews": {
  "reviewData": [
    { "allowReviewersToJoin": true,
      ...
    }, ... ]
  }
}
```

Get Single Review Details

URL:

```
GET /reviews-v1/<id>/details
```

Description:

Get details of a single review.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<detailedReviewData>
  ...
</detailedReviewData>
```

Example JSON Return Data:

```
{ "detailedReviewData": {
  "allowReviewersToJoin": false,
  ...
}}
```

Get Version Info

URL:

```
GET /reviews-v1/versionInfo
```

Description:

Returns the release number and build date of Crucible.

Status Code:

200 (OK) on success.

Example XML Return Data:

```
<versionInfo>
  <buildDate>2008-10-28</buildDate>
  <releaseNumber>1.6.3</releaseNumber>
</versionInfo>
```

Example JSON Return Data:

```
{ "versionInfo": { "buildDate": "2008-10-28", "releaseNumber": "1.6.3" } }
```

Review Service - Workflow

On this page:

- [Workflow](#)
 - [Close a Review](#)
 - [Complete a Review](#)
 - [Move a Review to a New State](#)
 - [Uncomplete a Review](#)

Workflow

Close a Review

```
POST /reviews-v1/<review id>/close
```

Description:

Close the review.

Status Code:

200 (OK) on success.

Complete a Review

```
POST /reviews-v1/<review id>/complete
```

Description:

Indicate that the current user has completed the review.

Status Code:

200 (OK) on success.

Move a Review to a New State

```
POST /reviews-v1/<review id>/transition?action=<action>
```

Description:

Change the state of the review.

Status Code:

200 (OK) on success.

Valid actions are:

```
action:abandonReview
action:deleteReview
action:submitReview
action:approveReview
action:rejectReview
action:summarizeReview
action:closeReview
action:reopenReview
action:recoverReview
action:completeReview
action:uncompleteReview
```

Uncomplete a Review

```
POST /reviews-v1/<review id>/uncomplete
```

Description:

Indicate that the current user has **not** completed the review.

Status Code:

200 (OK) on success.

Crucible Plugin Types

Crucible plugins come in a variety of flavours, read on to see how the plugin technology interacts with the core of Crucible and what rules can be bent, or possibly *broken* in this world.

Source Code Management (SCM) Plugins
<ul style="list-style-type: none">• Crucible SCM Plugins
Servlets
<ul style="list-style-type: none">• Servlet Modules
Event Listeners
<ul style="list-style-type: none">• Event Listener Plugins

Crucible Web Items

Web UI plugin modules allow you to add links, interactive elements and page segments to the Crucible user interface. By adding a link to a servlet plugin you can add your own pages to the UI. Your pages will need to ask Crucible to provide standard headers and footers by specifying a [Decorator](#). There are also [FishEye Web Items](#).

On this page:

- [Web Items Listing and Reference](#)
- [Web Item Conditions](#)

- [Condition Parameters](#)
- [Example of a Web Item Condition in Use](#)
- [Visual Locations of Crucible Web Items](#)

For an example of the code syntax, see [FishEye Web Items](#).

Web Items Listing and Reference

Key	Description	Helpers available
system.admin	Links on the admin menu. Sections: repositories, global, system.	application, user
system.crucible.dashboard	After 'My Dashboard' in the dropdown project/dashboard selector menu	application, user
system.crucible.review	Actions which can be performed on a review. These appear both as buttons on the review page and as links on an expanded review in a review list.	application, user, project, review
system.crucible.review.comment	Actions which can be performed on a review comment. These appear as buttons in the comment header bar, left of the reply, edit and delete buttons.	application, user, project, review, reviewItem, repository, comment
system.crucible.review.fileitem	Actions which can be performed on a revision in a review. Displayed next to the 'Remove', 'Change Diff' buttons.	application, user, project, review, reviewItem, repository
system.header.application	Links like 'Crucible' and 'FishEye' in the header. Sections: fisheye, crucible (sections are shown when a particular application is selected)	application, user
system.header.item	Links in the header, separated by a pipe.	application, user
system.main	Links on the Crucible or FishEye main page. These appear at the bottom of the Crucible or FishEye boxes on the main page. Sections: fisheye, crucible	application, user
system.userprofile.tab	The user profile tabs.	application, user

Web Item Conditions

Conditions control whether a given web item will be displayed.

com.atlassian.fisheye.plugin.web.conditions.HasCrucible

This condition measures whether the product runs with a Crucible license.

com.atlassian.fisheye.plugin.web.conditions.HasFishEye

This condition measures whether the product runs with a FishEye license. This is useful to prevent a Crucible plugin from rendering in an instance that only has a FishEye license.

com.atlassian.fisheye.plugin.web.conditions.HasProjectPermission

This condition measures whether the user has project permission. Takes parameters.

com.atlassian.fisheye.plugin.web.conditions.HasReviewPermission

This condition measures whether the user has review permission (i.e. is able to take part in the review). Takes parameters.

com.atlassian.fisheye.plugin.web.conditions.IsFile

This condition passes if there is a context repository and path, and that path references a repository file.

com.atlassian.fisheye.plugin.web.conditions.IsReviewInState

This condition measures whether the review is in a given state. Takes parameters.

com.atlassian.fisheye.plugin.web.conditions.IsRootOrDirectory

This condition passes if there is either: a context repository and no repository context path; or a repository path is present, and that path references a directory.

com.atlassian.fisheye.plugin.web.conditions.IsSystemAdministrator

This condition measures whether the user has system administrator permissions.

com.atlassian.fisheye.plugin.web.conditions.UserCanAccessCrucible

This condition measures whether the user can access Crucible.

com.atlassian.fisheye.plugin.web.conditions.UserLoggedInCondition

This condition measures whether the user is logged in.

Condition Parameters

The following conditions take parameters:

- HasProjectPermission
- HasReviewPermission
- IsReviewInState

The usage and conditions that these parameters apply to are tabled below.

Parameter Value	Parameter Name	Description	Applies to
action:abandonReview	actionName	Causes the current review to be abandoned.	HasProjectPermission, HasReviewPermission
action:approveReview	actionName	Causes the current review to be approved.	HasProjectPermission, HasReviewPermission
action:closeReview	actionName	Causes the current review to be closed.	HasProjectPermission, HasReviewPermission
action:recoverReview	actionName	Causes the current review to be recovered.	HasProjectPermission, HasReviewPermission
action:reopenReview	actionName	Causes the current review to be re-opened.	HasProjectPermission, HasReviewPermission
action:rejectReview	actionName	Causes the current review to be rejected.	HasProjectPermission, HasReviewPermission
action:submitReview	actionName	Causes the current review to be submitted.	HasProjectPermission, HasReviewPermission
action:summarizeReview	actionName	Causes the current review to be summarised.	HasProjectPermission, HasReviewPermission
Approval	stateName	Measures whether the current review is in the approval state.	IsReviewInState
Closed	stateName	Measures whether the current review is in the closed state.	IsReviewInState
Dead	stateName	Measures whether the current review is in the dead state.	IsReviewInState
Draft	stateName	Measures whether the current review is in the draft state.	IsReviewInState
Review	stateName	Measures whether the current review is in the review state.	IsReviewInState
Rejected	stateName	Measures whether the current review is in the rejected state.	IsReviewInState
Summarize	stateName	Measures whether the current review is in the summarize state.	IsReviewInState
Unknown	stateName	Measures whether the current review is in the unknown state.	IsReviewInState

Applying these values will cause the action to be enacted on the currently logged-in user.

Example of Condition Parameters in Use

```
<condition class="com.atlassian.fisheye.plugin.web.conditions.HasReviewPermission">
  <param name="actionName" value="action:approveReview"/>
</condition>
```

Example of a Web Item Condition in Use


```

<web-item key="hello-file2" section="system.crucible.review.fileitem">
  <link>#set($x =
'test')/plugins/servlet/${x}-servlet?name=${helper.global.user.displayName}</link>
  <label key="Id: {0}">
    <param name="param0">${helper.review.permaId.id}</param>
  </label>
  <condition class="com.atlassian.fisheye.plugin.web.conditions.IsReviewInState">
    <param name="stateName" value="Draft"/>
  </condition>
</web-item>

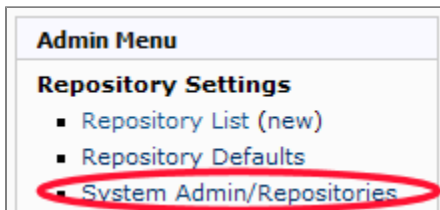
```

Visual Locations of Crucible Web Items

system.admin

This item relates to links in the left navigation bar, in the Crucible admin menu.

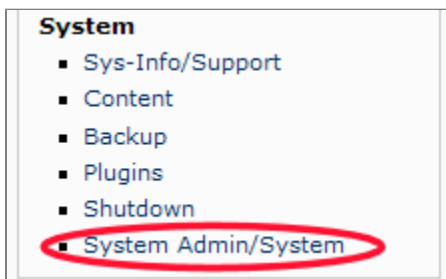
Screenshot: Crucible's system.admin/repositories Web Item



Screenshot: Crucible's system.admin/global Web Item



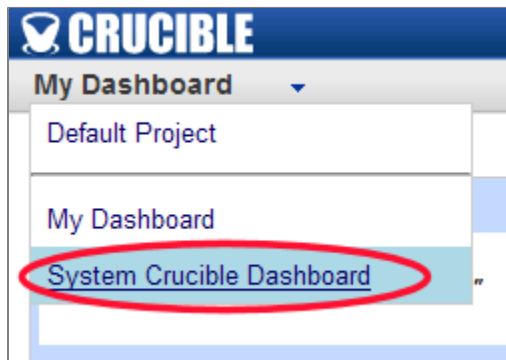
Screenshot: Crucible's system.admin/system Web Item



system.crucible.dashboard

This item relates to dashboard links in the Crucible dashboard/project drop-down menu.

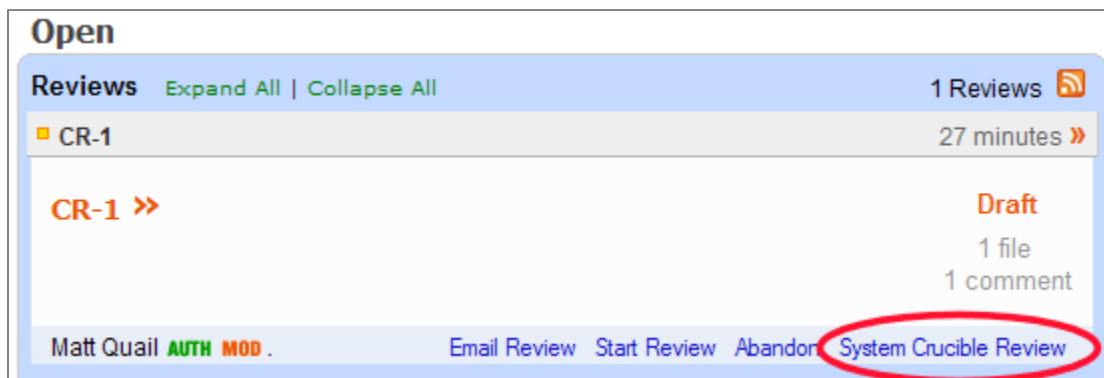
Screenshot: Crucible's system.crucible.dashboard Web Item



system.crucible.review

This item relates to actions that can be performed on a review, appearing in various places inside the Crucible UI.

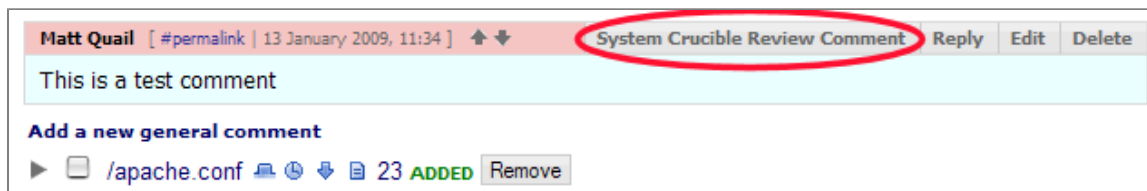
Screenshot: Crucible's system.crucible.review Web Item



system.crucible.review.comment

This item relates to actions that can be performed on a review, appearing in various places inside the Crucible UI.

Screenshot: Crucible's system.crucible.review.comment Web Item



system.crucible.review.fileitem

This item relates to actions which can be performed on a revision in a review, in the Crucible UI.

Screenshot: Crucible's system.crucible.review.fileitem Web Item



system.header.application

This item relates to product name links in the Crucible header.

i Note that a system.header.application item must go into a section of either Crucible or FishEye. In this case, we have put it into the Crucible section.

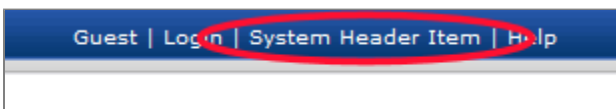
Screenshot: Crucible's system.header.application Web Item



system.header.item

This item relates to links in the Crucible header, at the top right of the Crucible screen.

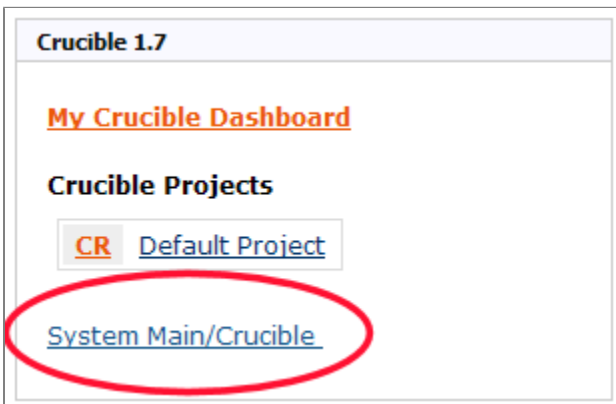
Screenshot: Crucible's system.header.item Web Item



system.main/crucible

This item relates to links at the bottom of the Crucible main page.

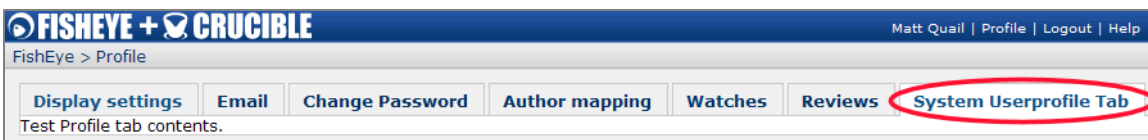
Screenshot: Crucible's system.main/crucible Web Item




system.userprofile.tab

This item relates to user profile tabs in the Crucible UI.

Screenshot: Crucible's system.userprofile.tab Web Item



 Looking for the FishEye web items? [Click here](#).

Live Code Examples for Crucible Development

On this page is a list of real-world plugin examples that showcase the various sides of Crucible development. The following items are an excellent resource for the Atlassian developer community. Feel free to investigate these examples, hack them to pieces, or use them as inspiration to really innovate.

SCM Plugin Examples
<ul style="list-style-type: none">• Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)• Crucible ClearCase plugin
Servlet Examples
<ul style="list-style-type: none">• Basic Servlet Example• Crucible Reporting plugin

Bundled Plugins from Crucible

There are a number of Crucible features that ship as plugins with the product. See below for listings and more information.

- [Confluence SCM Plugin](#)
- [File System SCM Plugin](#)
- [Perforce SCM Plugin](#)
- [Subversion SCM Plugin](#)

Confluence SCM Plugin

Name	Crucible Confluence SCM plugin
Version	1.0-SNAPSHOT
Product Versions	1.6.2 and later
Author(s)	Atlassian
Homepage	http://confluence.atlassian.com/display/CRUCIBLE/
Price	Bundled with Crucible
License	BSD
JavaDocs	Not Applicable
IssueTracking	Jira
Subversion URL	Repository: https://svn.atlassian.com/svn/public/contrib/crucible/crucible-confluence-scm-plugin/trunk FishEye: http://svn.atlassian.com/fisheye/browse/public/contrib/crucible/crucible-confluence-scm-plugin
Download JAR	https://maven.atlassian.com/repository/public/com/atlassian/crucible/plugins/crucible-confluence-scm-plugin/1.0-SNAPSHOT/crucible-confluence-scm-plugin-1.0-SNAPSHOT-jar-with-dependencies.jar

Description/Features

This page provides links to resources for Crucible plugin developers. Access the source code and distributed packages from the links in the table above.

In conjunction with the [Crucible Plugin](#) this allows the creation of review of changes to Confluence pages.

Usage

To set up this plugin, see the [documentation](#).

Installation

This plugin ships with the Crucible distribution.

Configuration

Follow the Crucible [configuration documentation](#).

Related Links

- [How to build a Crucible Plugin](#)

File System SCM Plugin

Name	Crucible File System SCM plugin
Version	1.2
Product Versions	1.6.0+
Author(s)	Atlassian
Homepage	http://confluence.atlassian.com/display/CRUCIBLE/
Price	Bundled with Crucible
License	BSD
JavaDocs	Not Applicable
IssueTracking	http://developer.atlassian.com/jira/browse/CRU
Subversion URL	Repository: https://svn.atlassian.com/svn/public/contrib/crucible/crucible-filessystem-scm-plugin/trunk FishEye: http://svn.atlassian.com/fisheye/browse/public/contrib/crucible/crucible-filessystem-scm-plugin

Description/Features

This page provides links to resources for Crucible plugin developers. Access the source code from the links in the table above.

This plugin allows Crucible to review files directly on the server file system.

Usage

To use this plugin, see the [documentation](#).

Installation

This plugin ships with the Crucible distribution.

Related Links

- [How to build a Crucible Plugin](#)

Perforce SCM Plugin

Name	Crucible Perforce SCM plugin
Version	1.2
Product Versions	1.6.4+
Author(s)	Atlassian
Homepage	http://confluence.atlassian.com/display/CRUCIBLE/
Price	Bundled with Crucible
License	BSD

JavaDocs	Not Applicable
IssueTracking	http://developer.atlassian.com/jira/browse/CRU
Subversion URL	Repository: https://svn.atlassian.com/svn/public/contrib/crucible/crucible-p4-scm-plugin/trunk FishEye: http://svn.atlassian.com/fisheye/browse/public/contrib/crucible/crucible-p4-scm-plugin

Description/Features

This page provides links to resources for Crucible plugin developers. Access the source code from the links in the table above.

This plugin allows Crucible to access Perforce repositories without requiring FishEye.

Usage

To use this plugin, see the [documentation](#).

Installation

This plugin ships with the Crucible distribution.

Related Links

- [How to build a Crucible Plugin](#)

Subversion SCM Plugin

Name	Crucible Subversion SCM plugin
Version	1.2
Product Versions	1.6.0+
Author(s)	Atlassian
Homepage	http://confluence.atlassian.com/display/CRUCIBLE/
Price	Bundled with Crucible
License	BSD
JavaDocs	Not Applicable
IssueTracking	http://developer.atlassian.com/jira/browse/CRU
Subversion URL	Repository: https://svn.atlassian.com/svn/public/contrib/crucible/crucible-svn-scm-plugin/trunk FishEye: http://svn.atlassian.com/fisheye/browse/public/contrib/crucible/crucible-svn-scm-plugin

Description/Features

This page provides links to resources for Crucible plugin developers. Access the source code from the links in the table above.

This plugin allows Crucible to access Subversion repositories without requiring FishEye.

Usage

To use this plugin, see the [documentation](#).

Installation

This plugin ships with the Crucible distribution.

Related Links

- [How to build a Crucible Plugin](#)

SCM Plugin Examples

This page contains examples of SCM plugins that can be added to Crucible.

Crucible SCM plugin listing:

- [Crucible ClearCase plugin](#)
- [Crucible Git Plugin](#)
- [Example Crucible SCM Plugin for JSR-170 \(Apache JackRabbit\)](#)

Crucible ClearCase plugin

Name	Crucible ClearCase plugin
Version	0.1.0
Product Versions	1.6.1 to 2.0
Author(s)	Ross Rowe
Homepage	http://confluence.atlassian.com/display/CRUCIBLE/Crucible+ClearCase+plugin
Price	Free
License	BSD
JavaDocs	TBA
IssueTracking	Jira
Subversion URL	https://svn.atlassian.com/svn/public/contrib/crucible/crucible-clearcase-plugin or browse via fisheye
Download JAR	crucible-clearcase-0.0.1.jar (supports 1.6.1 and 1.6.2) crucible-clearcase-0.0.3.jar (supports 1.6.3 to 1.6.5) crucible-clearcase-0.0.5.jar (supports 1.6.6) crucible-clearcase-0.1.0.jar (supports 2.0)
Download Source	TBA

Description/Features

A plugin for [Crucible](#) that facilitates the usage of ClearCase UCM source code repositories.

Usage



This plugin requires Crucible 1.6.1 or higher. In addition, at the moment this plugin only supports the LiteSCM module provided by Crucible and ClearCase UCM.

Installation

The plugin can be installed by copying the [crucible-clearcase-0.0.3.jar](#) file into the `CRUCIBLE_HOME/var/plugins/user` directory and restarting Crucible. Detailed instructions on the plugin installation steps can be found at the [Managing Plugins](#) page.

Configuring the plugin

Once the plugin has been installed, under the 'Administration' - 'Repository List' option, there should be a 'Plugin Repository List: ClearCase' entry. Select 'Configure Plugin', then 'Add a repository'. The fields required are:

Field	Description
Name	The name for the repository eg. <i>Project</i>
Main Component	The component containing the source files eg. <i>project_main@pvob</i>
Integration Stream	The integration stream for your project eg. <i>stream:Project_Integration@pvob</i>
View Location	The location of the main component for the view (snapshot or dynamic) for the project eg. <i>c:/projects/project/project_main</i>

Reviews can be driven from change sets, which are populated from the list baselines that exist for the ClearCase project, or from the file contents of the view location.

Version History

Version	Date	Description
0.1.0	7 Jul 2009	CCCCP-3 Updated plugin to support Crucible 2.0
0.0.5	4 Mar 2009	Updated fix for CCCCP-1
0.0.4	10 Feb 2009	Recompiled plugin to support Crucible 1.6.6
0.0.3	4 Feb 2009	Included fix for CCCCP-1
0.0.2	11 Nov 2008	Updated plugin to support Crucible 1.6.3
0.0.1	1 Oct 2008	Initial version of plugin

Screenshots

Screenshots

There are no images attached to this page.

Crucible Git Plugin

Name	Crucible Git plugin
Version	1.0
Product Versions	1.6.6
Author(s)	Atlassian
Homepage	http://confluence.atlassian.com/display/CRUCIBLE/Crucible+Git+plugin
Price	Free
License	BSD
JavaDocs	TBA
IssueTracking	JIRA
Subversion URL	Subversion FishEye
Download JAR	Atlassian Maven Repository (supports Crucible 1.6.6)
Download Source	TBA

Description/Features

A plugin for [Crucible](#) that facilitates the usage of [Git](#) source code repositories.

Usage

The Git plugin is an early-access implementation of a Crucible SCM plugin for Git. It allows users to perform code reviews on a local Git repository (local to the Crucible server). The plugin does not 'pull' updates from a remote master repository. Synchronising with the master repository needs to be executed manually ([via the command line](#)), for the changes to appear in the plugin.

Installation

The plugin is installed by placing the .JAR file in the `FISHEYE_INST/var/plugins/user` directory of your Crucible install. Once installed, you need to enable the plugin in the Crucible Admin interface. Detailed instructions on the plugin installation steps can be found at the [Managing Plugins](#) page.

The plugin requires the Git command to be available in the system path when starting Crucible.

Configuring the plugin

Once the plugin has been installed, under the 'Administration' - 'Repository List' option, there should be a 'Plugin Repository List: Git' entry. Select 'Configure Plugin', then 'Add a repository'. The fields required are:

Field	Description
Name	The name for the repository eg. <i>Project</i>
Repository Path	The location of the local Git repository clone

Once configured, the Git repository can be selected as the review source when creating a new review, whereupon reviews can be created either using changesets or by selecting files in the repository view.

Feedback

If you have any feedback on this plugin and its operation, we would appreciate users posting feedback in the [Crucible Forums](#).

Version History

Version	Date	Description
1.0	10 Feb 2009	Early Access as part of Crucible 1.6.6

Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)

Name	Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)
Version	1.0-SNAPSHOT
Product Versions	1.6.3
Author(s)	Erik van Zijst, Atlassian
Homepage	http://confluence.atlassian.com/display/CRUCIBLE/
Price	Free
License	BSD
JavaDocs	Crucible SCM API JavaDoc
Browse Source	FishEye
Download Source	Subversion

Description/Features

This plugin contains an implementation of Crucible SCM for Java Content Repositories ([JSR-170](#)), using the [Apache JackRabbit](#) implementation. Note that although this code can be used without modification, it is NOT supported and intended as example code only. It should not be used in a production environment, because of several known issues discussed below. Having said that, feel free to improve on this code 😊

Usage

Check out the source and build the plugin jar file using `mvn package`. Note that this plugin depends on `com.sun.jdmk:jmxtools:1.2.1` and `com.sun.jmx:jmxrmi:1.2.1`. These libraries are provided by Sun, but may not be present in your maven repository, as Sun requires each user to agree to its license terms. If you get the following build error:

```

[INFO] -----
[ERROR] BUILD ERROR
[INFO] -----
[INFO] Failed to resolve artifact.

Missing:
-----
1) com.sun.jdmk:jmxtools:jar:1.2.1

Try downloading the file manually from:
http://java.sun.com/products/JavaManagement/download.html

Then, install it using the command:
mvn install:install-file -DgroupId=com.sun.jdmk -DartifactId=jmxtools -Dversion=1.2.1
-Dpackaging=jar -Dfile=/path/to/file

Alternatively, if you host your own repository you can deploy the file there:
mvn deploy:deploy-file -DgroupId=com.sun.jdmk -DartifactId=jmxtools -Dversion=1.2.1
-Dpackaging=jar -Dfile=/path/to/file -Durl=[url] -DrepositoryId=[id]

Path to dependency:
1) com.atlassian.crucible.example.scm:jackrabbit-scm-plugin:atlassian-plugin:1.0-SNAPSHOT
2) com.sun.jdmk:jmxtools:jar:1.2.1

2) com.sun.jmx:jmxri:jar:1.2.1

Try downloading the file manually from the project website.

Then, install it using the command:
mvn install:install-file -DgroupId=com.sun.jmx -DartifactId=jmxri -Dversion=1.2.1
-Dpackaging=jar -Dfile=/path/to/file

Alternatively, if you host your own repository you can deploy the file there:
mvn deploy:deploy-file -DgroupId=com.sun.jmx -DartifactId=jmxri -Dversion=1.2.1 -Dpackaging=jar
-Dfile=/path/to/file -Durl=[url] -DrepositoryId=[id]

Path to dependency:
1) com.atlassian.crucible.example.scm:jackrabbit-scm-plugin:atlassian-plugin:1.0-SNAPSHOT
2) log4j:log4j:jar:1.2.15
3) com.sun.jmx:jmxri:jar:1.2.1

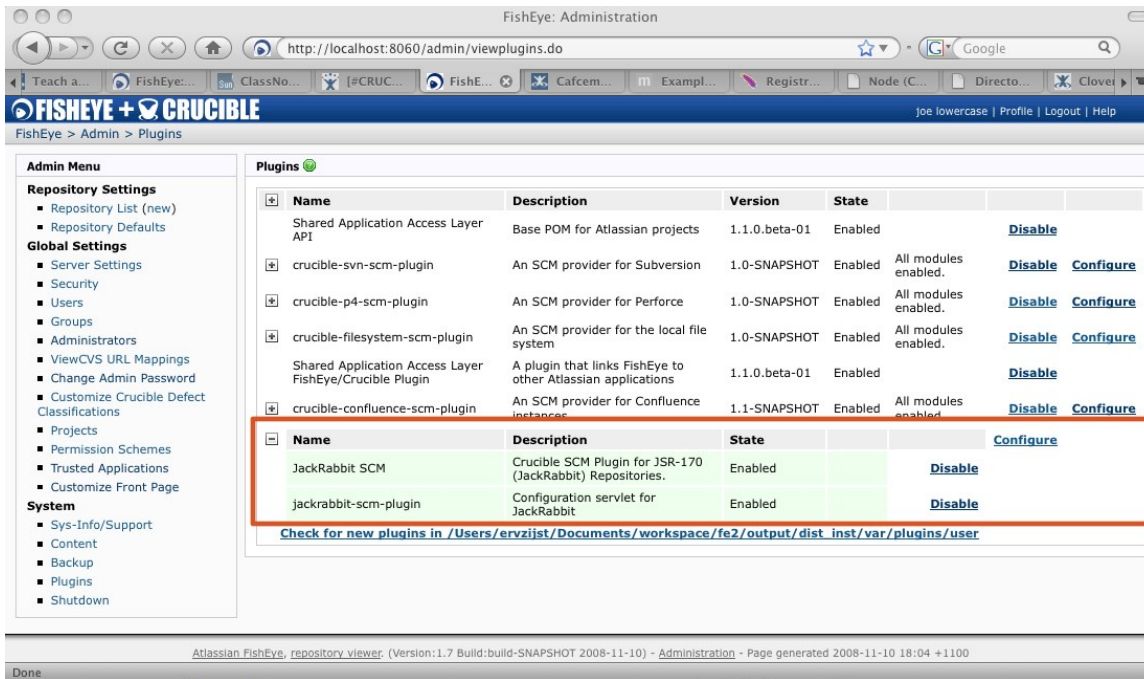
-----
2 required artifacts are missing.

```

Then download the two libraries from [Sun](#), extract the jar files from the zip files and follow the instructions above to manually install the artifact in your local repository. You should then be able to complete the build.

Installation

Copy the jackrabbit-scm-plugin-1.0-SNAPSHOT.jar file from the 'target/' directory to the var/plugins/user directory in your Crucible installation. Then login to the administration section, go to Plugins and click the link "Check for new plugins in...". This should detect your plugin:



Configuring the plugin

Next, click "Configure" and add a repository. The current version of this plugin can only read JackRabbit repositories directly from the local file system (as opposed to connecting to a remote JackRabbit server), so when configuring a repository, specify the location of the repository's XML file and the repository's home directory (the directory containing `workspaces/`, `repository/` and `version/`).

The plugin comes with a ready to use, pre-populated JackRabbit repository in `testrepo.zip` (the unit tests run against this repository). When trying things out, unzip this file somewhere on the file system and point the plugin to it.

Known Limitations

As this plugin is intended as example material for those interested in building Crucible SCM Plugins, readability of the source code is more important than features, flexibility or performance and as result there are quite a number of known limitations:

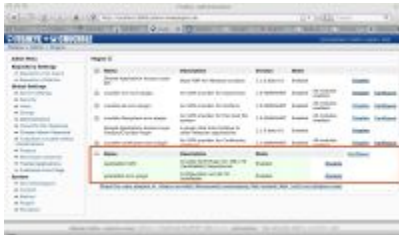
- Only file based JackRabbit 1.x repositories on the local file system are currently supported. This makes this plugin impossible to use with "active" repositories, because the JackRabbit acquires an exclusive lock on the repository.
- The plugin only recognizes nodes of JCR types `"nt:file"` and `"nt:folder"`. All other nodes are ignored.
- All file and folder nodes must have the mixin type `"mix:versionable"`.
- Every file or folder node must have at least one checked-in version (hence: at least one version of each resource must have been committed).
- JSR-170 does not support ChangeSets** where a changeset represents a collection of related changes to multiple entities at once. As a consequence, the plugin represents every individual change as a ChangeSet containing one file. The UUID of the version node is used as the changeset ID.
- This plugin does not detect files that are deleted.
- Due to file locking issues in JackRabbit, the plugin uses a single JCR Session instance per repository. Access is synchronized.
- The implementation for listing a set of changesets traverses the entire repository and does not cache results, which kills scalability.

Version History

Version	Date	Description
1.0-SNAPSHOT	10-November-2008	Initial release

Screenshots

Screenshots



Servlet Examples

This page lists the Servlet code examples for Crucible plugin developers.

- [Basic Servlet Example](#)
- [Crucible Reporting plugin](#)

Basic Servlet Example

Name	Example FishEye Servlet Plugin
Version	0.1
Product Versions	1.7
Author(s)	Anna Buttfield, Atlassian
Homepage	http://confluence.atlassian.com/display/FISHEYE/
Price	Free
License	BSD
JavaDocs	Data Package Summary Services Package Summary
IssueTracking	N/A
Subversion URL	FishEye svn
Download JAR	Attached to this page: compiled jar sources jar
Download Source	svn

Description/Features

Basic plugin showing the use of the FishEye API in a servlet. This can be used as the basis for more advanced FishEye plugins.

Usage

If compiling from source, follow the instructions listed in the '[readme](#)' file.



Requires a version 1.7 (or later) [FishEye development build](#), will not work with released versions.

Installation

1. Copy the plugin .jar file from the 'target' directory to the var/plugins/user directory in your FishEye installation.
2. Run FishEye and point your browser at this location:

```
FISHEYE_HOME/plugins/servlet/[servlet url]
```

to view the servlet (where the servlet url is set in 'url-pattern' in **atlassian-plugin.xml** and is set to `example-servlet` by default).

Configuring the plugin

No configuration is required, just start FishEye and point a browser at this URL:

```
FISHEYE_HOME/plugins/servlet/example_servlet
```

Version History

Version	Date	Description
0.1	7-November-2008	Initial release

Screenshots

Screenshots



Crucible Reporting plugin

Name	Crucible Reporting plugin
Version	1.0.0
Product Versions	1.5.x, 1.6.x
Author(s)	Ross Rowe
Homepage	http://confluence.atlassian.com/display/CODEGEIST/Crucible+Reporting+plugin
Price	Free
License	BSD
JavaDocs	crucible-export-plugin-javadoc.zip
IssueTracking	N/A
Subversion URL	https://svn.atlassian.com/svn/public/contrib/crucible/crucible-export-plugin or browse via fisheye
Download JAR	crucible-export-1.0.0.jar (for Crucible 1.5.x) / crucible-export-1.6.2.jar (for Crucible 1.6.x)
Download Source	crucible-export.zip

Description/Features

A plugin for [Crucible](#) that facilitates the generation of a consolidated report for a specific review. This is especially useful if you are required to keep hard copies of your code review (like in the case of an audit 😊)

Usage



This plugin requires Crucible 1.5 or 1.6

Crucible 1.5 installation

The plugin can be installed by copying the [crucible-export-1.0.0.jar](#) file into the CRUCIBLE_HOME/var/plugins directory. You will also need to copy the [iText](#) jar file (available from <http://www.lowagie.com/iText/download.html>) into the CRUCIBLE_HOME/lib directory.

Crucible 1.6 installation

The plugin can be installed by copying the [crucible-export-1.6.2.jar](#) file into the CRUCIBLE_HOME/var/plugins/user directory.

Running the plugin

As Crucible does not currently have a mechanism to include user interface components via it's plugin api, the export mechanism can be opened by visiting <http://YourCrucibleHost/plugins/servlet/export>. From this page, the user must enter their username, password and the review id they wish to export.

The screenshot shows a web browser window titled 'Crucible 1.5' with the address bar displaying 'http://localhost:8060/plugins/servlet/export'. The page header includes the 'FISHEYE + CRUCIBLE' logo and 'Login | Help' links. The main content area is titled 'Generate a PDF export of a review' and contains three input fields: 'Review Id:' with the value 'CR-1', 'Username:' with the value 'ross', and 'Password:' with masked characters '....'. A 'Run' button is located below the password field. At the bottom of the page, a footer line reads: 'Atlassian Crucible, painless code review. (Version:1.5 Build:build-293 2008-04-15) - Administration - Page generated 2008-04-17 19:34 +1000'.

Once the details are entered and the 'Run' button is clicked, a PDF including the Crucible Review details is generated. This report includes the summary information of the review, as well as any general and specific file comments.

Version History

Crucible 1.6 support

Version	Date	Description
1.6.2	22 Mar 2009	CRPT-2 Sort versioned comments for a specific file (thanks Soren!)
1.6.1	10 Dec 2008	Added ability to include defect and revision information in report (thanks Soren!)
1.6.0	22 Sep 2008	Updated plugin to support Crucible 1.6.0 beta

Crucible 1.5 support

Version	Date	Description
1.0.0	9 May 2008	Updated plugin to support Crucible 1.5.1
0.0.4	4 May 2008	Added i18n support
0.0.3	29 April 2008	Updated unit tests
0.0.2	21 April 2008	Updated plugin to support Crucible 1.5
0.0.1	23 Mar 2008	Initial plugin version

Screenshots

Screenshots



Input page for the Crucible Export plugin
Sample report layout

Developing Crucible Plugins

Introduction

Crucible uses the standard [Atlassian Plugins framework](#), so many of the tasks involved in developing a plugin for Crucible are the same as for other Atlassian products.

The differences are:

- The set of plugin types available.
- And, of course, the API available for plugins to interact with the Crucible application.

Building a Crucible Plugin

The simplest way to build a Crucible plugin is via Maven.

Atlassian provides an Archetype for Fisheye/Crucible plugins.

You can create a maven 2 project containing a sample Servlet Plugin Module with the following command:

```
mvn org.apache.maven.plugins:maven-archetype-plugin:1.0-alpha-7:create \
  -DarchetypeGroupId=com.atlassian.maven.archetypes \
  -DarchetypeArtifactId=crucible-plugin-archetype \
  -DarchetypeVersion=1-SNAPSHOT \
  -DremoteRepositories=https://maven.atlassian.com/repository/public/ \
  -DgroupId=com.foo -DartifactId=foo-crucible-plugin
```

This will create your project in a subdirectory of your current directory named `foo-crucible-plugin`. Change into that directory (`cd foo-crucible-plugin`). You can create the plugin jar with the command `mvn package`, and install it in a running Fisheye or Crucible instance by copying `target/foo-crucible-plugin-1.0-SNAPSHOT.jar` to the `var/plugins/user` directory of your Fisheye/Crucible instance.

Crucible Plugin Module Types

Servlet Modules

Create a servlet which is deployed to the same web application context as Fisheye/Crucible. See [Servlet Plugin Modules](#) for more details.

Crucible SCM Plugins

An SCM plugin module lets Crucible create reviews based on files stored in another source code management system. See [Crucible SCM Plugins](#) for details.

Event Listener Plugins

An event listener plugin module will be called when certain events occur inside Crucible. See [Crucible Event Listener Plugins](#) for details.

The Crucible API

Your plugin will need to use [The Crucible API](#) to retrieve data from Crucible and to perform operations on it, such as changing the state of reviews.

Debugging your plugin

You can start Crucible in debug mode with the environment variable setting:

```
export FISHEYE_OPTS="-Xdebug -Xrunjdpw:transport=dt_socket,server=y,suspend=n,address=5005"
```

This allows you to connect your IDE to the debugger listening on port 5005.

Crucible Event Listener Plugins

An event listener plugin module is an object which is notified when certain internal Crucible or FishEye events occur.

To include an event listener module add a `listener` element to your `atlassian-plugins.xml` file:

```
<listener key="example-listener" class=
"com.atlassian.crucible.example.plugin.spring.ExampleListener"/>
```

and create a class which implements `com.atlassian.event.EventListener`. See the [FishEye event](#) and [Crucible event](#) javadoc for specific event types. See the [javadoc for EventListener](#) to understand the general details regarding events.

For example, if we want to listen for all events, and print a message to standard output we would write:

```
public class ExampleListener implements EventListener {
    public void handleEvent(Event event) {
        System.out.println("Got event: " + event);
    }

    public Class[] getHandledEventClasses() {
        return new Class[0];
    }
}
```

Event listeners may implement [StateAware](#) if they need to be notified when the module is enabled or disabled.

A plugin containing an event listener module needs to declare a dependency on `atlassian-events` in its `pom.xml`:


```
<dependency>
  <groupId>com.atlassian.event</groupId>
  <artifactId>atlassian-event</artifactId>
  <version>0.5</version>
  <scope>provided</scope>
</dependency>
```

Note that this is a *provided* dependency – the plugin does not need to include the `atlassian-events` classes.

Crucible SCM Plugins

On this page:

- [Crucible SCM Plugins](#)
 - [Creating a Project](#)
 - [Crucible SCM Plugin API](#)
 - [Servlet Based Administration Pane](#)
 - [Packaging, Deploying and Running](#)

Crucible SCM Plugins

Crucible SCM modules are plugins that make version control systems accessible to Crucible. An SCM plugin can be used to give Crucible the ability to work with a custom version control system that is not supported out of the box. SCM plugins are independent from FishEye's version control integrations and allow Crucible to run standalone. Crucible ships with a number of built-in SCM plugins, including Subversion and Perforce.

In this section we will implement a new Crucible SCM Plugin and explore Crucible's public SCM API. The example builds a module that exposes the underlying file system as the "repository", so that users can perform reviews of files on the server file system.

Creating a Project

To start, we use the Crucible Plugin archetype to create a new empty Maven2 project:

```
mvn org.apache.maven.plugins:maven-archetype-plugin:1.0-alpha-7:create \
-DarchetypeGroupId=com.atlassian.maven.archetypes \
-DarchetypeArtifactId=crucible-plugin-archetype \
-DarchetypeVersion=1-SNAPSHOT \
-DremoteRepositories=https://maven.atlassian.com/repository/public/ \
-DgroupId=com.atlassian.crucible.example.scm \
-DartifactId=example-scm-plugin
```

This creates a new project that has a dependency on `atlassian-fisheye-api`. This library contains the basic API components required by plugins. However, as we are building an SCM plugin that can be configured through a servlet, we need to add a dependency on `atlassian-crucible-scmutils` as well as `atlassian-plugins-core` by editing the generated `pom.xml`:

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

  <parent>
    <groupId>com.atlassian.crucible.plugin.base</groupId>
    <artifactId>crucible-plugin-base</artifactId>
    <version>1-SNAPSHOT</version>
  </parent>

  <modelVersion>4.0.0</modelVersion>
  <groupId>com.atlassian.crucible.example.scm</groupId>
  <artifactId>example-scm-plugin</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>Example plugin that offers file system based repositories.</name>
  <packaging>atlassian-plugin</packaging>

  <properties>
    <atlassian.plugin.key>com.atlassian.crucible.example.scm.example-scm-plugin
  </atlassian.plugin.key>
  <atlassian.pdk.server.url>${atlassian.product.url}</atlassian.pdk.server.url>
  <atlassian.product.version>1.6.3-SNAPSHOT</atlassian.product.version>
</properties>

  <dependencies>
    <dependency>
      <groupId>com.atlassian.crucible</groupId>
      <artifactId>atlassian-crucible-scmutils</artifactId>
      <version>${atlassian.product.version}</version>
    </dependency>
    <dependency>
      <groupId>com.atlassian.plugins</groupId>
      <artifactId>atlassian-plugins-core</artifactId>
      <version>2.0.4</version>
    </dependency>
  </dependencies>
</project>
```



IDEA Users

If you are using IntelliJ for development, be sure to run `mvn idea:idea` to generate the project files. Opening the pom file directly is known to miss the parent dependencies.

Crucible SCM Plugin API

Crucible's public API can be [browsed online](#) and contains the functionality needed to develop a custom SCM plugin in the package `com.atlassian.crucible.scm`. It consists of a set of interfaces, some of which are optional, for browsing a repository, accessing its directories, retrieving file contents and exploring changes between revisions.

At the very least, your SCM plugin should implement the `com.atlassian.crucible.scm.SCMModule` interface that defines the new plugin. The module is then used to create one or more repository instances:

```

package com.atlassian.scm;

import com.atlassian.crucible.scm.SCModule;
import com.atlassian.crucible.scm.SCMRepository;
import com.atlassian.plugin.ModuleDescriptor;

import java.util.Collection;
import java.util.Collections;

public class ExampleSCModule implements SCModule {

    private ModuleDescriptor moduleDescriptor;
    private List<SCMRepository> repos = Collections.emptyList();

    public String getName() {
        return "Example File System SCM.";
    }

    public Collection<? extends SCMRepository> getRepositories() {
        return repos;
    }

    public void setModuleDescriptor(ModuleDescriptor moduleDescriptor) {
        this.moduleDescriptor = moduleDescriptor;
    }

    public ModuleDescriptor getModuleDescriptor() {
        return moduleDescriptor;
    }
}

```

When your module is instantiated, Crucible passes a `ModuleDescriptor` instance to it containing information about the plugin. The `getRepositories()` method returns the repositories offered by this plugin. Currently we're returning an empty collection.

To be able to use the Crucible administration console to configure our plugin and specify the locations of the repositories we want to use, we will also implement the `Configurable` interface that allows for the injection of a custom configuration bean (by implementing `SimpleConfiguration`) whose properties can be manipulated through the administration interface for which we will write a small servlet. In our custom configuration bean we'll add a property for the base path or root directory of the file system based repositories we want to offer.

The plugin configuration is written to disk and fed to our `SCModule` when Crucible starts up. Our plugin is responsible for generating and parsing that data, so we're free to choose the format. The `ModuleConfigurationStore` provides persistent storage and will automatically be injected into our plugin if we create a constructor that takes it as an argument. For the serialization, let's use simple XML serialization through [XStream](#) (using XStream is convenient as it is one of the dependencies for `atlassian-crucible-scmutils`):

```

package com.atlassian.scm;

import com.atlassian.fisheye.plugins.scm.utils.SimpleConfiguration;

public class ExampleConfiguration implements SimpleConfiguration {

    private String name;
    private String basePath;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getBasePath() {
        return basePath;
    }

    public void setBasePath(String basePath) {
        this.basePath = basePath;
    }
}

```

Now we make the required changes to our SCModule to read and write the configuration:

```

public class ExampleSCModule implements SCModule, Configurable<List<ExampleConfiguration>> {

    private ModuleDescriptor moduleDescriptor;
    private ModuleConfigurationStore store;

    public ExampleSCModule(ModuleConfigurationStore store) {
        this.store = store;
    }

    [...]

    public List<ExampleConfiguration> getConfiguration() {
        byte[] configData = store.getConfiguration(moduleDescriptor);
        if (configData != null) {
            try {
                return (List<ExampleConfiguration>)getXStream().fromXML(new String(configData, "UTF8"));
            } catch (Exception e) {
                throw new RuntimeException("Error reading configuration:" + configData, e);
            }
        }
        return new ArrayList<ExampleConfiguration>();
    }

    public void setConfiguration(List<ExampleConfiguration> config) {
        try {
            store.putConfiguration(moduleDescriptor, getXStream().toXML(config).getBytes("UTF8"));
        } catch (UnsupportedEncodingException e) {
            throw new RuntimeException("UTF8 encoding not supported", e);
        }
    }

    private XStream getXStream() {
        XStream xstream = new XStream();
        xstream.setClassLoader(moduleDescriptor.getPlugin().getClassLoader());
        return xstream;
    }

    [...]
}

```

Now that we have access to the configuration data, which describes the repositories, we can go ahead and implement our file system based repository class.

The `SCMRepository` interface offers basic functionality for retrieving file contents of specific file revisions. It is queried by Crucible when a user adds files to a review. Depending on the optional interfaces you implement in addition to `SCMRepository`, your implementation could also have the ability to browse the repository and to explore different versions of each file. Because a standard file system does not store version information, we'll only offer directory browsing in this example. As a revision key or version number we shall simply use the last modification date that is stored by the file system.

```
package com.atlassian.scm;

import com.atlassian.crucible.scm.SCMRepository;
import com.atlassian.crucible.scm.RevisionData;
import com.atlassian.crucible.scm.RevisionKey;
import com.atlassian.crucible.scm.DetailConstants;
import com.cenqua.crucible.model.Principal;

import java.io.OutputStream;
import java.io.IOException;
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.Date;
import java.net.MalformedURLException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

import org.apache.commons.io.IOUtils;

public class ExampleSCMRepository implements SCMRepository {

    private final ExampleConfiguration config;

    public ExampleSCMRepository(ExampleConfiguration config) {
        this.config = config;
    }

    public boolean isAvailable(Principal principal) {
        return true;
    }

    public String getName() {
        return config.getName();
    }

    public String getDescription() {
        return getName() + " file system repo at: " + config.getBasePath();
    }

    public String getStateDescription() {
        return "Available";
    }

    public RevisionData getRevisionData(Principal principal,
        RevisionKey revisionKey) {
        if (revisionKey.equals(currentKey(revisionKey.getPath()))) {
            File f = getFile(revisionKey.getPath());

            RevisionData data = new RevisionData();
            data.setDetail(DetailConstants.COMMIT_DATE, new Date(f.lastModified()));
            data.setDetail(DetailConstants.FILE_TYPE, f.isDirectory() ? "dir" : "file");
            data.setDetail(DetailConstants.ADDED, true);
            data.setDetail(DetailConstants.DELETED, false);
            try {
                data.setDetail(DetailConstants.REVISION_LINK, f.toURL().toString());
            } catch (MalformedURLException e) {
            }
            return data;
        } else {

```

```

        throw new RuntimeException("Revision " + revisionKey.getRevision() + " of file " +
revisionKey.getPath() + " is no longer available.");
    }
}

public void streamContents(Principal principal, RevisionKey revisionKey,
    OutputStream outputStream) throws IOException {
    if (revisionKey.equals(currentKey(revisionKey.getPath()))) {
        InputStream is = new FileInputStream(getFile(revisionKey.getPath()));
        try {
            IOUtils.copy(is, outputStream);
        } finally {
            IOUtils.closeQuietly(is);
        }
    } else {
        throw new RuntimeException("Revision " + revisionKey.getRevision() + " of file " +
revisionKey.getPath() + " is no longer available.");
    }
}

public RevisionKey getDiffRevisionKey(Principal principal,
    RevisionKey revisionKey) {
    // diffs are not supported in this example
return null;
}

/**
 * Returns a {@link RevisionKey} instance for the specified file. Because we
 * do not support versioning, the revision string will be set to the file's
 * last modification date.
 *
 * @param path
 * @return
 */
private RevisionKey currentKey(String path) {
    File f = getFile(path);
    return new RevisionKey(path, createDateFormat().format(new Date(f.lastModified())));
}

/**
 * Takes the name of a file in the repository and returns a file handle to the
 * file on disk.
 *
 * @param path
 * @return
 */
private File getFile(String path) {
    return new File(config.getBasePath() + File.separator + path);
}

private DateFormat createDateFormat() {
    return new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss.SSSZ");
}

```

```

    }
}

```

In the above code, the `getRevisionData()` method is used by Crucible to retrieve versioning properties for a specific revision of a file in the repository. Although the file system does not keep track of older versions, we can provide some of the properties. Most important are the predefined constants `DetailConstants.FILE_TYPE`, `DetailConstants.ADDED`, `DetailConstants.DELETED` (the last two indicate whether the file was newly created (ADDED), or has been removed from the repository (DELETED) as part of the revision) and `DetailConstants.REVISION_LINK`. In addition to the predefined constants, a repository implementation is free to add custom properties.

We are not able to implement `getDiffRevisionKey()` due to the lack of version information on the file system.

Before we continue to extend the functionality of the `ExampleSCMRepository`, we should go back to `ExampleSCMModule` and implement `getRepositories()`:

```

[...]
```

```

    // initialize at null to trigger loading from the configuration
    private List<SCMRepository> repos = null;

    public synchronized Collection<SCMRepository> getRepositories() {
        if (repos == null) {
            repos = new ArrayList<SCMRepository>();
            for (ExampleConfiguration config : getConfiguration()) {
                repos.add(new ExampleSCMRepository(config));
            }
        }
        return repos;
    }

    public void setConfiguration(List<ExampleConfiguration> config) {
        try {
            store.putConfiguration(moduleDescriptor, xstream.toXML(config).getBytes("UTF8"));
            // we're given a new configuration, so reset our repositories:
            repos = null;
        } catch (UnsupportedEncodingException e) {
            throw new RuntimeException("UTF8 encoding not supported", e);
        }
    }

[...]
```

Our `SCMModule` now properly creates the repository instances according to the configuration.

The above code gives us a very simple Crucible SCM plugin. However you would normally also want to implement the `com.atlassian.crucible.scm.DirectoryBrowser` and `com.atlassian.crucible.scm.HasDirectoryBrowser` interfaces. The `DirectoryBrowser` gives Crucible the ability to let the user interactively browse the repository and select files to review. If you do not provide a `DirectoryBrowser`, the only way to create a review for files in your repository is when the required files and file revisions are known up front.

In this example, we'll implement `DirectoryBrowser`:

```

public class FileSystemSCMRepository implements HasDirectoryBrowser, DirectoryBrowser {

    [...]

    public DirectoryBrowser getDirectoryBrowser() {
        return this;
    }

    public List<FileSummary> listFiles(Principal principal, String path) {
        List<FileSummary> files = new ArrayList<FileSummary>();
        for (String p : list(path, true)) {
            files.add(new FileSummary(currentKey(p)));
        }
        return files;
    }

    public List<DirectorySummary> listDirectories(Principal principal, String path) {
        List<DirectorySummary> files = new ArrayList<DirectorySummary>();
        for (String p : list(path, false)) {
            files.add(new DirectorySummary(p));
        }
        return files;
    }

    public FileHistory getFileHistory(Principal principal, String path) {
        return new FileHistory(Collections.singletonList(currentKey(path)));
    }

    private List<String> list(String path, boolean returnFiles) {
        File parent = getFile(path);
        List<String> files = new ArrayList<String>();
        if (parent.isDirectory()) {
            File[] children = parent.listFiles();
            // this may be null if we can't read the directory, for instance.
            if (children != null) {
                for (File f : children) {
                    if (f.isFile() && returnFiles || f.isDirectory() && !returnFiles) {
                        files.add(getPath(f));
                    }
                }
            }
        }
        return files;
    }

    /**
     * @return the path for a given File relative to the base configured for this
     *         repository -- the path doesn't include the base component.
     */
    private String getPath(File file) {
        String s = file.getAbsolutePath();
        if (!s.startsWith(config.getBasePath())) {
            throw new RuntimeException("Invalid file with path " + s + " is not under base " +
config.getBasePath());
        }
        return s.substring(config.getBasePath().length() + 1);
    }

    [...]
}

```

This is as far as we can go with the file system. In most cases you will be integrating version control systems that keep track of all previous revisions of the resources in the repository and you would expose this to Crucible by also implementing `HasChangelogBrowser` and `ChangelogBrowser`.

Servlet Based Administration Pane

With the code for the module and the repository in place, we can focus on our servlet that provide plugin administration in Crucible's administration section. The easiest way to do this is to subclass

`com.atlassian.fisheye.plugins.scm.utils.SimpleConfigurationServlet` and implement the three abstract methods:

```
package com.atlassian.crucible.example.scm;

import com.atlassian.fisheye.plugins.scm.utils.SimpleConfigurationServlet;
import com.atlassian.plugin.PluginAccessor;
import com.atlassian.crucible.spi.FisheyePluginUtilities;

public class ExampleSCMConfigServlet extends SimpleConfigurationServlet<ExampleConfiguration> {

    public ExampleSCMConfigServlet(PluginAccessor pluginAccessor,
        FisheyePluginUtilities fisheyePluginUtilities) {
        super(pluginAccessor, fisheyePluginUtilities);
    }

    protected ExampleConfiguration defaultConfig() {
        return new ExampleConfiguration();
    }

    protected String getProviderPluginModuleKey() {
        return "com.atlassian.crucible.example.scm.example-scm-plugin:scmprovider";
    }

    protected String getTemplatePackage() {
        return "/examplescm-templates";
    }
}
```

The `getTemplatePackage()` method returns the name of the resource directory that contains the [velocity templates](#) that determine how the configuration pane will be rendered. The template directory must be in `src/main/resources` so Crucible can find them. We'll create three different pages: one that lists the current configuration `list.vm`, one to edit a repository's configuration `edit.vm` and one that is displayed when the user tries to manipulate a non-existing repository instance (`nosuchrepo.vm`):

src/main/resource/examplescm-templates/list.vm

```
<html>
<head>
    <link rel="stylesheet" href="$request.contextPath/$STATICDIR/main.css" type="text/css" />
</head>
<body class="plugin">
<div class="box formPane">
<table class="adminTable">
#if ($configs.empty)
    <tr><td>No File System repositories are configured.</td></tr>
#else
    <tr>
        <th>Name</th>
        <th>Base Path</th>
        <th><!-- for edit link --></th>
        <th><!-- for delete link --></th>
    </tr>
    #foreach ($config in $configs)
    <tr>
        <td>$config.name</td>
        <td>$config.basePath</td>
        <td><a href="/examplescm?name=$config.name">Edit</a></td>
        <td><a href="/examplescm?name=$config.name&delete=true">Delete</a></td>
    </tr>
    #end
    #end
    <tr>
        <td class="verb"><a href="/examplescm?name=_new">Add a repository.</a></td>
    </tr>
</table>
</div>
</body>
</html>
```

src/main/resource/examplescm-templates/edit.vm

```
<html>
<head>
  <link rel="stylesheet" href="$request.contextPath/$STATICDIR/main.css" type="text/css" />
</head>
<body class="plugin">
<div class="box formPane">
<form action="./examplescm" method="POST">
  #if ($config.name)
    <input type="hidden" name="name" value="$!config.name"/>
  #end
  <table class="adminTable">
    #if ($errorMessage)
      <tr><td colspan="2"><span class="errorMessage">$errorMessage</span></td></tr>
    #end
    <tr>
      <td class="tdLabel"><label class="label">Name:</label></td> <td><input
        #if ($config.name)
          disabled="true"
        #else
          name="name"
        #end
        type="text" value="$!config.name"/> </td>
      </tr>
      <tr>
        <td class="tdLabel"><label class="label">Base Path:</label></td> <td><input type="text"
name="basePath" value="$!config.basePath"/> </td>
      </tr>
      <tr>
        <td colspan="2">
          <input type="submit" value="Save"/>
        </td>
      </tr>
    </table>
  </form>
</div>
</body>
</html>
```

src/main/resource/examplescm-templates/nosuchrepo.vm

```
<html>
<head>
  <link rel="stylesheet" href="$request.contextPath/$STATICDIR/main.css" type="text/css" />
</head>
<body class="plugin">
<p>
There is no repository named '$name'.
</p>
</body>
</html>
```

Finally we tie everything together in the mandatory `atlassian-plugin.xml` file that describes the new plugin, contains its name, location of the servlet and the classnames Crucible uses to instantiate the components. Because this is an SCM plugin, we must add the `<scm/>` element:

src/main/resources/atlassian-plugin.xml

```
<atlassian-plugin key="${atlassian.plugin.key}" name="example-scm-plugin" plugins-version="2">
  <plugin-info>
    <description>An example SCM provider for the local file system</description>
    <vendor name="Atlassian" url="http://www.atlassian.com"/>
    <version>1.0-SNAPSHOT</version>
    <param name="configure.url">/plugins/servlet/example SCM</param>
  </plugin-info>

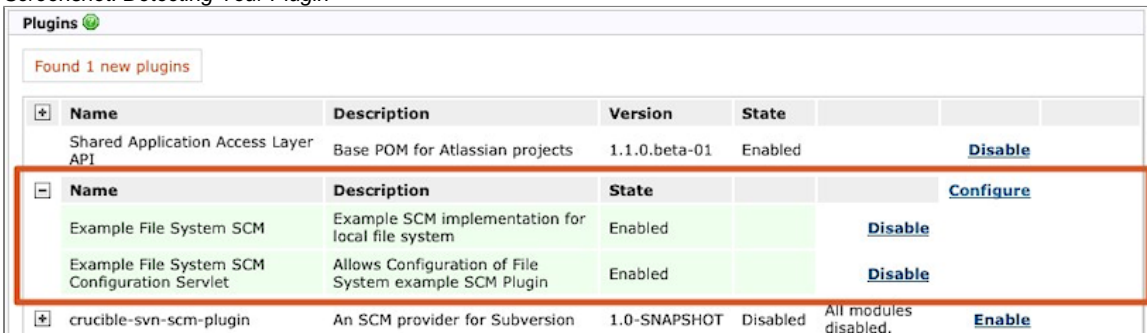
  <scm name="Example File System SCM" key="scmprovider" class=
"com.atlassian.crucible.example.scm.ExampleSCModule">
    <description>Example SCM implementation for local file system</description>
  </scm>

  <servlet name="Example File System SCM Configuration Servlet" key="configservlet" class=
"com.atlassian.crucible.example.scm.ExampleSCMConfigServlet" adminLevel="system">
    <description>Allows Configuration of File System example SCM Plugin</description>
    <url-pattern>/example SCM</url-pattern>
  </servlet>
</atlassian-plugin>
```

Packaging, Deploying and Running

Now we can package everything up using `mvn package` and you should end up with `target/example-scm-plugin-1.0-SNAPSHOT.jar` that can be deployed in Crucible by copying the jar file to the `CRUCIBLE_HOME/var/plugins/user` directory. Then login to the administration section, go to **Plugins** and click the link "Check for new plugins in...". This should detect your plugin and add it to the list in "disabled" state as illustrated below:

Screenshot: Detecting Your Plugin



Name	Description	Version	State	
Shared Application Access Layer API	Base POM for Atlassian projects	1.1.0.beta-01	Enabled	Disable
Example File System SCM	Example SCM implementation for local file system	Enabled		Disable
Example File System SCM Configuration Servlet	Allows Configuration of File System example SCM Plugin	Enabled		Disable
crucible-svn-scm-plugin	An SCM provider for Subversion	1.0-SNAPSHOT	Disabled	All modules disabled. Enable

Next, click "Configure" to create a file system based repository:

Screenshot: Creating a File-System Based Repository

[FishEye](#) > [Admin](#) > [Plugins](#) > [Configure Plugin](#)

Admin Menu

Repository Settings

- Repository List (new)
- Repository Defaults

Global Settings

- Server Settings
- Security
- Users
- Groups
- Administrators
- ViewCVS URL Mappings
- Change Admin Password
- Customize Crucible Defect Classifications
- Projects
- Permission Schemes
- Trusted Applications
- Customize Front Page

Configure Plugin

Name:

ExampleRepo

Base Path:

/tmp/exemplerepo

Save

When the repository is created, navigate to "Repository List". Our custom Crucible SCM Plugin will now show up in the list and is ready to use:

Screenshot: The Custom SCM Plugin in Crucible

[FishEye](#) > [Admin](#) > [Repository List](#)

Admin Menu

Repository Settings

- Repository List (new)
- Repository Defaults

Global Settings

- Server Settings
- Security
- Users
- Groups
- Administrators
- ViewCVS URL Mappings
- Change Admin Password
- Customize Crucible Defect Classifications
- Projects
- Permission Schemes

Repository List

Name	Description	Repository location	Status	Last Update	Operations
Local	Local, file-based svn test repo	file:///Users/ervzjst/var/repo/	Enabled: Running	26 seconds ago	View Browse Stop Restart Disable Delete

Add repository

Repository Plugin: Example File System SCM.

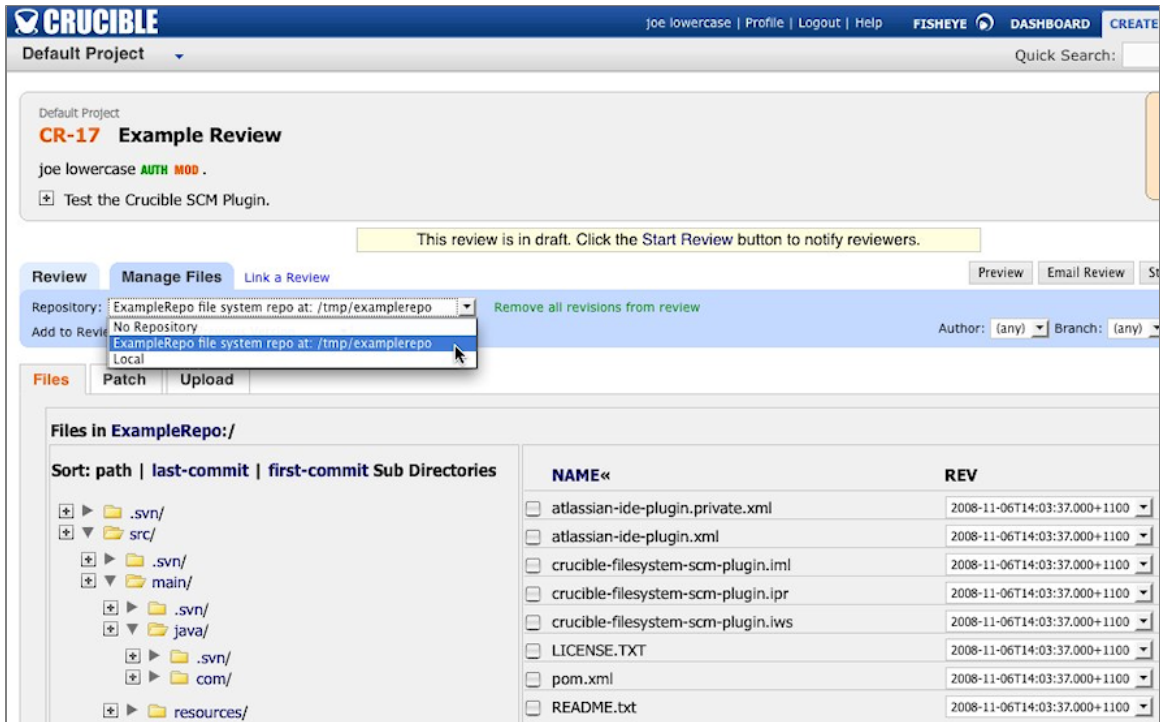
Name	Description
ExampleRepo	ExampleRepo file system repo at: /tmp/exemplerepo

Configure Plugin

More Plugins

When reviewing files from the plugin repository, click on the "Manage Files" tab in a new or existing review and then select the repository from the pull down list and select the files and revisions you want to review:

Screenshot: Selecting Files and Revisions for Review



The Crucible API

This page covers the Crucible API and how you can use service interfaces that are exposed to plugins.

On this page:

- [Services](#)
 - [Overview](#)
 - [Using a Service in your Plugin](#)
 - [Maven dependencies for Spring](#)

Services

Crucible exposes a set of service interfaces to plugins. The parameters and return types of the methods on these interfaces are 'plain old Java objects'. Having an API specifically designed to be used by plugins protects plugins from internal changes in Crucible's implementation, and presents plugins with a simpler API.

Overview

The service interfaces are in the package `com.atlassian.crucible.spi.services`.

The types they use as parameters are in the package `com.atlassian.crucible.spi.data`.

Refer to the [Crucible API javadoc](#) for details of the services.

Using a Service in your Plugin

The services are all available in your plugin's Spring context.

We can inject a spring bean using constructor injection, e.g.:

```

public class ExampleServlet extends HttpServlet {
    private ProjectService projectService;

    @Autowired
    public ExampleServlet(ProjectService projectService) {
        this.projectService = projectService;
    }
    ...
}

```

You can also use setter injection on your plugin class:

```

public void setReviewService(ReviewService reviewService) {
    this.reviewService = reviewService;
}

```

Note that you **cannot** mix constructor and setter injection in the same class – if you mark a constructor with `@Autowired`, no setters will be used for injection.

All plugin module classes which are created by the plugin system are injected by Spring. That is, the `HttpServlet` subclass of a servlet plugin module, the `SCModule` implementation of a Light SCM plugin module and the `EventListener` implementation of an event listener plugin module.

Maven dependencies for Spring

If you are using Spring annotations in your plugin you will need the following dependencies in your `pom.xml`:

```

<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring</artifactId>
    <version>2.5.5</version>
    <scope>provided</scope>
  </dependency>
  <dependency>
    <groupId>com.sun.xml.bind</groupId>
    <artifactId>jaxb-impl</artifactId>
    <version>2.1</version>
    <scope>provided</scope>
  </dependency>
</dependencies>

```

Crucible FAQ

Crucible FAQ

Answers to frequently asked questions about configuring and using Crucible.

- Top Evaluator Questions
 - Can Crucible add support for new repositories?
 - Can I purchase Crucible on it's own?
 - Can I trial Crucible without FishEye?
 - How can I do reviews from the file system?
 - How does Crucible help enforce compliance and auditability?
 - How do I convince my team of the benefits of code review?
 - How do I do pre-commit reviews?
 - How do I raise defects in JIRA?
 - How do I review patch diffs?
 - What user permissions and review security is available?
- Can Crucible be run as a Windows Service?
- Do I need a FishEye licence to run Crucible?
- How to Automate Daily Crucible Backups
- Troubleshooting
- Fix 'Out of Memory' errors by increasing available memory



Most setup issues are likely to be related to the FishEye component of Crucible. Refer to the FishEye documentation:

- [FishEye documentation](#)
- [FishEye knowledge base](#)

Do you still have a question, or need help with Crucible? Please [create a support request](#).

Can Crucible be run as a Windows Service?

To run Crucible as a service you can either use [SRVANY](#) and [INSTSRV](#) to run `java.exe` or [create a Java Service Wrapper](#). A mechanism to run Crucible as a service will be incorporated at a later stage. In the meantime, example wrapper files written by users can be found [here](#).

To install on Windows:

1. Unzip the wrapper zip file into your `FISHEYE_HOME` directory (Note, the end structure should be `FISHEYE_HOME/wrapper`, `FISHEYE_HOME/wrapper/bin`, etc and NOT `FISHEYE_HOME/wrapper/wrapper`, `FISHEYE_HOME/wrapper/wrapper/bin`. The location of the wrapper directory is important).
2. Run `Fisheye-Install-NTService.bat`, found in `FISHEYE_HOME/wrapper/bin`.
3. Start the Fisheye service under the Windows Control Panel.
4. Set your `FISHEYE_INST` within your `FISHEYE_HOME/wrapper/conf/wrapper.conf` as per the instructions below:

Please note, that if you do run as a service, then any [Environment Variables](#) that you want to set, need to be set in your `FISHEYE_HOME/wrapper/conf/wrapper.conf` file.

If there are other java parameters you wish to add, then you will need to add them under the additional parameters, e.g.

```
# JDK 1.5 Additional Parameters for jmx
wrapper.java.additional.4=-Dcom.sun.management.jmxremote
wrapper.java.additional.5=-Dcom.sun.management.jmxremote.port=4242
wrapper.java.additional.6=-Dcom.sun.management.jmxremote.authenticate=false
wrapper.java.additional.7=-Dcom.sun.management.jmxremote.ssl=false
wrapper.java.additional.8=-Dcom.sun.management.jmxremote.authenticate=false
wrapper.java.additional.9=-Dcom.sun.management.jmxremote.password.file=./wrapper/jmxremote.password
wrapper.java.additional.10=-Dwrapper.mbean.name="wrapper:type=Java Service Wrapper Control"
```

For example if you wish to add a `FISHEYE_INST` environment variable or add the java parameter "MaxPermSize", or the `-Xrs` options (should be used if running FishEye as a service under Windows, to prevent the JVM closing when an interactive user logs out) then it would be something like:

```
wrapper.java.additional.11=-Dfisheye.inst="c:/path/to/FISHEYE_INST"
wrapper.java.additional.12=-XX:MaxPermSize=128m
wrapper.java.additional.13=-Xrs
```

Your memory settings can also be found in this file:

```
# Initial Java Heap Size (in MB)
wrapper.java.initmemory=32

# Maximum Java Heap Size (in MB)
wrapper.java.maxmemory=256
```

Increase these values if you have a large repository or expect to use more memory (init of 256, and a max of 1024 would be reasonable).

Do I need a FishEye licence to run Crucible?

FishEye and Crucible are separate products. They can be run separately, and they can also be run together.

You can run Crucible standalone, but you will not have access to some of FishEye's advanced features, such as pre-caching repository content (for maximum performance), advanced searching and FishEye's activity graphs.

Crucible stand-alone can be used with Subversion, CVS, Perforce, Git and Clearcase via SCM plugins. In this situation, access to the repositories is strictly on-demand.

For the best combination of features and performance, run both Crucible and FishEye.

How to Automate Daily Crucible Backups

To set daily Crucible backups, open the administration page, click the '**Backup**' link under '**System**' on the left navigation bar, and simply follow the instructions set out on the [Backing Up and Restoring Crucible Data](#) page.

Troubleshooting

The most common cause of FishEye/Crucible issues is an incorrect [symbolic setup](#) (trunk/branch/tag) for Subversion repositories. If you are using Subversion and your initial index is taking forever, double-check that your symbolic setup matches your repository.

FishEye runs with the default Java heap of 64 megabytes. This is sometimes problematic for FishEye, especially for Subversion repositories during the initial scan. You can give FishEye's JVM more memory by setting the FISHEYE_OPTS [environment variable](#).

Starting Crucible with the [command line options](#) `--debug --debug-perf` will print a lot of information to Crucible's logs. This can give you an insight into what is happening and possibly where you are stuck. Attach these logs along with your `config.xml` to an Atlassian support ticket, to speed up your [support request](#).


Crucible Installation & Upgrade Guide

- [Crucible Installation Guide](#)
 - [System Requirements](#)
 - [Installing Crucible](#)
 - [Configuring Crucible](#)
 - [Configuring Repositories](#)
 - [Best Practices for Crucible Configuration](#)
- [Crucible Upgrade Guide](#)
 - [Upgrading to a New Version of Crucible](#)
 - [Upgrading from FishEye to Crucible](#)

Crucible Installation Guide

This guide explains how to get Crucible installed and running as easily as possible. Many references are made to the [FishEye documentation](#).

This document assumes you have extracted your Crucible zip file into a directory called `/FISHEYE_HOME/`.

 Refer to our explanation of how [Crucible works with FishEye](#).








Knowledge Base

You may find some useful information in the [Knowledge Base](#) too.

- [System Requirements](#)
- [Installing Crucible](#)
- [Configuring Crucible](#)
- [Configuring Repositories](#)
- [Best Practices for Crucible Configuration](#)

System Requirements

These are the requirements for Crucible and FishEye. See the [FishEye requirements](#) for more specific details, including hardware requirements.

Java Runtime	<p>A JDK or JRE version 1.5 or greater.</p> <p> There is a known issue with Crucible 1.6.x and the JDK/JRE versions 1.6.0 through 1.6.0u3(update 3). Crucible requires version 2.1 of a component called JAXB, but these versions of the JDK/JRE include JAXB2.0. To avoid this issue, upgrade your JDK/JRE to version 1.6.u4(update 4) or later.</p> <p> You can download a Java Runtime for Windows/Linux/Solaris here.</p> <p> On Mac OS X, the JDK is bundled with the operating system.</p> <p> We strongly recommend the use of a 32-bit JDK/JRE rather than a 64-bit JDK/JRE. 64-bit JDK/JREs will consume the available RAM more rapidly, and this may result in poor performance.</p>
Source code Repository	Crucible stand-alone supports use of Subversion, Confluence and the server file system as a repository. It can also store files in its own database, removing the need for any kind of repository. When Crucible is used with FishEye, Subversion, CVS and Perforce are supported.
Web Browser	<p>Crucible has been tested with Firefox 3, Internet Explorer 7, and Safari 4. IE6 is NOT supported. Crucible should work with most modern browsers.</p> <div>  Font size (Especially for Linux users.) For best results you may want to tweak your default monospace font and font-size. The default browser font is usually Courier New which can be hard to read in some browsers. We recommend choosing the same font you use in your IDE and selecting a font size approximately 2 points larger than your variable width font. Firefox 3, Internet Explorer 7 and Safari all have excellent font rendering. It is worth taking some time to tweak your fonts for the best experience. </div>
Operating System	Crucible is a pure Java application and should run on any platform provided the above requirements are satisfied.

Installing Crucible

This page contains instructions for the initial installation of Crucible.

On this page:

- [Installing the Crucible Binary Files](#)
- [Setting up a Repository for use with Stand-alone Crucible](#)
- [Setting up a Repository for use with FishEye and Crucible](#)
- [Next: Configuration](#)

Installing the Crucible Binary Files

Follow these steps to install Crucible:

1. [Download](#) the Crucible zip file and extract it. This document assumes you have extracted your Crucible zip file into a directory called `/FISHEYE_HOME/`.
2. Ensure you have installed an appropriate Java runtime - see [System Requirements](#). Ensure that `java` is in the `PATH`, or that the `JAVA_HOME` [environment variable](#) is set.

Setting up a Repository for use with Stand-alone Crucible

When accessing repositories from stand-alone Crucible, the SCM client interface will access the repositories on demand. This requires that you configure the plugin.

For complete instructions, see [Configuring Repositories](#).

Setting up a Repository for use with FishEye and Crucible

- If you intend to use Crucible and FishEye with [Subversion](#), please ensure you read the [System Requirements](#), [Subversion client setup](#), and [granting permission to FishEye](#) to scan your repository.
- If you intend to use Crucible and FishEye with [Git](#), please ensure you read the [System Requirements](#) and [Git Client setup](#).
- If you intend to use Crucible and FishEye with [Perforce](#), please ensure you read the [System Requirements](#) and [Perforce Client setup](#).
- If you intend to use Crucible and FishEye with [CVS](#), please ensure you read the [System Requirements](#) and [CVS Client setup](#).

Next: Configuration

To go on with configuration, see the page [configuring Crucible](#).

Configuring Crucible

On this page:

- [Running Crucible](#)
- [Supplying Administration Password and License Key](#)
- [Accessing the Administration Pages](#)
- [Setting Up a Perforce Repository in Stand-Alone Crucible](#)
- [Setting Up a Subversion Repository in Stand-Alone Crucible](#)
- [Setting Up Reviewing of Confluence Pages in Crucible](#)
- [Setting Up the File System as a Code Repository For Crucible](#)
- [Setting Up a Repository via FishEye](#)
- [Setting Up Users](#)
- [Setting Up SMTP](#)
- [Using Crucible](#)
- [Stopping Crucible](#)
- [Information on FishEye integration](#)

Running Crucible

 This document assumes you have extracted your Crucible zip file into a directory called `/FISHEYE_HOME/`.

To run Crucible for the first time, simply do the following:


- On Windows:

```
C:\> cd FISHEYE_HOME\bin
C:\FISHEYE_HOME\bin> run.bat
```

- On Unix-based systems:

```
$ cd /FISHEYE_HOME/bin
$ ./run.sh
```

Once started, Crucible will run its own HTTP web server on port 8060. You can access Crucible immediately by going to <http://HOSTNAME:8060/> in a browser.

 By default, Crucible will listen on port 8060 for HTTP requests. It also listens on 127.0.0.1:8059 as a control port. You can configure both of these in the Administration screens, or by editing `/FISHEYE_HOME/config.xml` and restarting Crucible.

Supplying Administration Password and License Key

The first time you access the Crucible web server (<http://HOSTNAME:8060/>) you will see a screen like [this](#), and here you will be asked for two things:

1. An administrator password. This password controls access to the Administration screens.
2. A license key. Please note your [server ID](#). You can then get a Crucible evaluation license key [here](#).

Accessing the Administration Pages

Once you have set up an administrator password (as described above), you can access the Administration screens at <http://HOSTNAME:8060/admin/>.

One of your first steps will be to set up access to a source-control repository, or an alternative form of code storage such as the Local File System or [Atlassian Confluence](#).

Setting Up a Perforce Repository in Stand-Alone Crucible

To set up Perforce in stand-alone Crucible,

1. Ensure that the Perforce executable file is on the system path, in the Crucible server's [Environment Variables](#)
2. Start Crucible then open the 'Admin' menu by clicking the *Administration* link in the footer of the page.
3. Under the 'Repository Settings' heading, click 'Repository List' in the left-hand navigation bar.
4. The 'Repository List' screen opens.
5. Find the **Perforce** repository plugin and click its **Configure Plugin** link.
6. The 'Configure Plugin' screen opens. Click 'Add Repository'.
7. The 'Add Repository' screen opens. Fill in the fields.

Configure Plugin

Name:

Example Server

Repository Server:

example.com:666

Repository Path:

//depot/code/example/main/


Perforce Username:

admin


Perforce Password:


Save

|| Field || What to enter ||

Name	Choose a unique name for the repository.
Repository Server	Enter the base URL and port for the repository, for example: example.com:666.
Repository Path	Add the path to your Perforce repository. For example: //depot/code/example/main.
Perforce Username	Enter the username of the Perforce account that Crucible will use. (optional)  Note that this account should only have read-only access to the repository.
Perforce Password	Enter the password of the Perforce account that Crucible will use. (optional)

8. Click 'Save'. The view will return to the list of repositories.
9. Your Perforce repository is now set up for Crucible. You will be able to select changesets from it when creating reviews.

 There is no 'initial scanning' required in this process, as stand-alone Crucible's access to Perforce is strictly on-demand. Data is not indexed, hence there is no scanning.

 For those who may be interested, Crucible executes the Perforce command-line tool to enable this functionality.

Setting Up a Subversion Repository in Stand-Alone Crucible


To set up Subversion in stand-alone Crucible,

1. Start Crucible then open the 'Admin' menu by clicking the *Administration* link in the footer of the page.
2. Under the 'Repository Settings' heading, click 'Repository List' in the left-hand navigation bar.
3. The 'Repository List' screen opens.
4. Find the SVN repository plugin and click its **Configure Plugin** link.
5. The 'Configure Plugin' screen opens. Click 'Add Repository'.
6. The 'Add Repository' screen opens. Fill in the fields.


Configure Plugin

Name:	<input type="text" value="subversion5"/>
Repository Root:	<input type="text" value="http://svn.example.com"/>
Repository Path:	<input type="text" value="svn5"/>
SVN Username:	<input type="text" value="subversion-admin"/>
SVN Password:	<input type="password" value=""/>
<input type="button" value="Save"/>	

|| Field || What to enter ||

Name	Choose a unique name for the repository.
Repository Root	Enter the repository root URL for the repository. If you are not sure what the repository root is, please see the instructions below under "Finding your Repository Root".
Repository Path	Add the path on the base URL where your repository. For example, if you used the root URL above, and the full path to your Subversion instance is 'http://svn.example.com/svn5/', you would enter 'svn5' into this field.
SVN Username	Enter the username of the Subversion account that Crucible will use.  Note that this account should only have read-only access to the repository.
SVN Password	Enter the password of the Subversion account that Crucible will use.

7. Click 'Save'. The view will return to the list of repositories.
8. Your Subversion repository is now set up for Crucible. You will be able to select changesets from it when creating reviews.

 There is no 'initial scanning' required in this process, as stand-alone Crucible's access to Subversion is strictly on-demand. Data is not cached, hence scanning is not required.

Setting Up Reviewing of Confluence Pages in Crucible

To set up Confluence as a Code Repository in Crucible,

1. Start Crucible then open the **'Admin'** menu by clicking the *Administration* link in the footer of the page.
2. Under the **'Repository Settings'** heading, click **'Repository List'** in the left-hand navigation bar.
3. The **'Repository List'** screen opens.
4. Find the **Confluence** repository plugin and click its **Configure Plugin** link.
5. The **'Configure Plugin'** screen opens. Click **'Add Repository'**.
6. The **'Add Repository'** screen opens. Fill in the fields.

Configure Plugin

Name:

URL:

Space Key:

To restrict the repository to a single space, set the Space Key.

Field	What to enter
Name	Choose a unique name for the repository.
URL	Enter the URL of your Confluence instance.
Space Key	You may optionally enter a space key here to restrict Crucible's view to that key only. If there are many spaces in your Confluence instance you will find that this improves performance. You can set up several Confluence repositories in Crucible, each using the same Confluence instance but covering a different Space.

7. Click **'Save'**. The view will return to the list of repositories.
8. Now, access your Confluence instance. Open the **'Confluence Administration Console'**, then select **'Trusted Applications'**. The Confluence **'Trusted Applications Details'** dialog opens.
9. In the **'Trusted Applications Details'** dialog, enter the URL of your Crucible instance into the **'Name'** field and click **'Send Request'**. The **'Application Alias'** will be automatically retrieved from Crucible. Save your changes.
10. Confluence is now set up as a code repository for Crucible. You will be able to select your Confluence from the list of repositories, then select files from the Confluence wiki and add them to reviews.

Setting Up the File System as a Code Repository For Crucible

The page Setting Up the File System as a Code Repository For Crucible does not exist.

Setting Up a Repository via FishEye

 This section requires a working [FishEye](#) installation as well as Crucible.

To use FishEye to access the source control repositories CVS, Subversion or Perforce for Crucible, see the FishEye documentation for how to [add a repository](#).

Once you have [added a repository](#), you can view it through FishEye at <http://HOSTNAME:8060/>.



Building index and cache

FishEye needs to build an index and cache of the contents of your repository, so some information will not appear in FishEye until this is complete. This may take some time to complete, depending on the size of the repositories.



We recommend you access the repository with a user that has only **read** access to the repository.

Setting Up Users

On initial setup of Crucible, there are no users. Adding user accounts is done via the Administration screens or by configuring Crucible/FishEye to use external authentication.

To add users:

1. Open the FishEye Administration screens at <http://HOSTNAME:8060/admin/>.
2. Click '**Users/Security**' under '**Global Settings**' in the '**Admin Menu**'.

Read more details about the different ways of [creating users](#).

Setting Up SMTP

Crucible can email each review participant on a range of changes. Each user can then set up their own preferences. This is described in the [User Profile guide](#).

First, you must [set up the SMTP Server](#).

Using Crucible

You can access Crucible immediately by going to <http://HOSTNAME:8060/> in a browser

Or you can go directly into the Crucible homepage at <http://HOSTNAME:8060/cru>

Stopping Crucible

To stop the Crucible server:

- On Windows:

```
C:\> cd FISHEYE_HOME\bin
C:\FISHEYE_HOME\bin> stop.bat
```

- On Unix-based systems:

```
$ cd /FISHEYE_HOME/bin
$ ./stop.sh
```

Information on FishEye integration

If you want to know more about how Crucible and FishEye interact, refer to our explanation of how [Crucible works with FishEye](#).

Configuring Repositories

This page contains links to the instructions for configuring the different kinds of repositories that Crucible supports. Click one of the links below to see the desired page.

- [Enabling Reviews from the Server File System in Crucible](#)
- [Setting Up a Git Repository in Stand-Alone Crucible](#)
- [Setting Up a Perforce Repository in Stand-Alone Crucible](#)
- [Setting Up a Repository via FishEye](#)
- [Setting Up a Subversion Repository in Stand-Alone Crucible](#)
- [Setting Up Reviewing of Confluence Pages in Crucible](#)

Enabling Reviews from the Server File System in Crucible

Enabling Reviews from the Server File System in Crucible

[To set up the File System as a Code Repository in stand-alone Crucible,](#)

1. Start Crucible then open the **'Admin'** menu.
2. Under the **'System'** heading, click **'Plugins'** in the left-hand navigation bar.
3. The **'Plugins'** screen opens.
4. Next to **'File System SCM'**, click **'Enable'**.
5. New options appear next to **'File System SCM'**: **'Disable'** and **'Configure'**. Click **'Configure'**.
6. The **'Configure Plugin'** screen opens. Click **'Add Repository'**.
7. The **'Add Repository'** screen opens. Fill in the fields.

|| Field || What to enter ||

Name	Choose a unique name for the repository.
Base Path	Choose the lowest level of directory that Crucible will access.

8. Click **'Save'**. The view will return to the list of repositories.
9. The server file system is now set up as a code repository for Crucible. You will be able to select files from it when creating reviews, with the ability to browse the files and directories on the hard drive.

Setting Up a Git Repository in Stand-Alone Crucible

This page contains instructions on how to configure the Crucible Git plugin to access Git repositories.

Usage

The Git plugin is an early-access implementation of a Crucible SCM plugin for Git. It allows users to perform code reviews on a local Git repository (local to the Crucible server). The plugin does not 'pull' updates from a remote master repository. Synchronising with the master repository needs to be executed manually ([via the command line](#)), for the changes to appear in the plugin.

Installation

The plugin is installed by placing the .JAR file in the `FISHEYE_INST/var/plugins/user` directory of your Crucible install. Once installed, you need to enable the plugin in the Crucible Admin interface. Detailed instructions on the plugin installation steps can be found at the [Managing Plugins](#) page.

The plugin requires the Git command to be available in the system path when starting Crucible.

Configuring the plugin

Once the plugin has been installed, under the **'Administration'** - **'Repository List'** option, there should be a **'Plugin Repository List: Git'** entry. Select **'Configure Plugin'**, then **'Add a repository'**. The fields required are:

Field	Description
Name	The name for the repository eg. <i>Project</i>
Repository Path	The location of the local Git repository clone

Once configured, the Git repository can be selected as the review source when creating a new review, whereupon reviews can be created either using changesets or by selecting files in the repository view.

Feedback

If you have any feedback on this plugin and its operation, we would appreciate users posting feedback in the [Crucible Forums](#).

For more information, see the page on the [Crucible Git Plugin](#).

Setting Up a Perforce Repository in Stand-Alone Crucible

Setting Up a Perforce Repository in Stand-Alone Crucible

To set up Perforce in stand-alone Crucible,

1. Ensure that the Perforce executable file is on the system path, in the Crucible server's [Environment Variables](#)
2. Start Crucible then open the 'Admin' menu by clicking the *Administration* link in the footer of the page.
3. Under the 'Repository Settings' heading, click 'Repository List' in the left-hand navigation bar.
4. The 'Repository List' screen opens.
5. Find the **Perforce** repository plugin and click its **Configure Plugin** link.
6. The 'Configure Plugin' screen opens. Click 'Add Repository'.
7. The 'Add Repository' screen opens. Fill in the fields.

Configure Plugin

Name:

Example Server

Repository Server:

example.com:666

Repository Path:

//depot/code/example/main/

Perforce Username:


admin

Perforce Password:


.....


Save

|| Field || What to enter ||

Name	Choose a unique name for the repository.
Repository Server	Enter the base URL and port for the repository, for example: example.com:666.
Repository Path	Add the path to your Perforce repository. For example: //depot/code/example/main.
Perforce Username	Enter the username of the Perforce account that Crucible will use. (optional)  Note that this account should only have read-only access to the repository.
Perforce Password	Enter the password of the Perforce account that Crucible will use. (optional)

8. Click 'Save'. The view will return to the list of repositories.
9. Your Perforce repository is now set up for Crucible. You will be able to select changesets from it when creating reviews.

 There is no 'initial scanning' required in this process, as stand-alone Crucible's access to Perforce is strictly on-demand. Data is not indexed, hence there is no scanning.

 For those who may be interested, Crucible executes the Perforce command-line tool to enable this functionality.

Setting Up a Repository via FishEye

Setting Up a Repository in Crucible via FishEye

 This section requires a working [FishEye](#) installation as well as Crucible.

To use FishEye to access the source control repositories CVS, Subversion or Perforce for Crucible, see the FishEye documentation for how to [add a repository](#).

Once you have [added a repository](#), you can view it through FishEye at <http://HOSTNAME:8060/>.



Building index and cache

FishEye needs to build an index and cache of the contents of your repository, so some information will not appear in FishEye until this is complete. This may take some time to complete, depending on the size of the repositories.



We recommend you access the repository with a user that has only **read** access to the repository.

Setting Up a Subversion Repository in Stand-Alone Crucible

Setting Up a Subversion Repository in Stand-Alone Crucible

To set up Subversion in stand-alone Crucible,

1. Start Crucible then open the '**Admin**' menu by clicking the *Administration* link in the footer of the page.
2. Under the '**Repository Settings**' heading, click '**Repository List**' in the left-hand navigation bar.
3. The '**Repository List**' screen opens.
4. Find the **SVN** repository plugin and click its **Configure Plugin** link.
5. The '**Configure Plugin**' screen opens. Click '**Add Repository**'.
6. The '**Add Repository**' screen opens. Fill in the fields.

Configure Plugin

Name:

subversion5

Repository Root:

http://svn.example.com

Repository Path:

svn5


SVN Username:

subversion-admin

SVN Password:

Save

|| Field || What to enter ||

Name	Choose a unique name for the repository.
Repository Root	Enter the repository root URL for the repository. If you are not sure what the repository root is, please see the instructions below under "Finding your Repository Root".
Repository Path	Add the path on the base URL where your repository. For example, if you used the root URL above, and the full path to your Subversion instance is 'http://svn.example.com/svn5/', you would enter 'svn5' into this field.
SVN Username	Enter the username of the Subversion account that Crucible will use.  Note that this account should only have read-only access to the repository.
SVN Password	Enter the password of the Subversion account that Crucible will use.

7. Click '**Save**'. The view will return to the list of repositories.
8. Your Subversion repository is now set up for Crucible. You will be able to select changesets from it when creating reviews.



There is no 'initial scanning' required in this process, as stand-alone Crucible's access to Subversion is strictly on-demand. Data is not cached, hence scanning is not required.

Finding your Repository Root.

Run the following command:

```
svn info SVN_URL
```

Where SVN_URL is the complete URL of the repository you want to add.


You will get something like the following:

```
>svn info http://svn.example.com/svn5/  
  
Path: svn5  
URL: http://svn.example.com/svn5/  
Repository Root: http://svn.example.com/  
Repository UUID: ce062a09-193b-427a-a7b3-a85007076e5d  
Revision: 83  
Node Kind: directory  
Last Changed Author: ryan  
Last Changed Rev: 83  
Last Changed Date: 2009-05-07 10:48:41 +1000 (Thu, 07 May 2009)
```

Next to "Repository Root" is the URL you should define as your repository root. The path will be whatever is remaining.

Setting Up Reviewing of Confluence Pages in Crucible

Setting Up Reviewing of Confluence Pages in Crucible

 You need to first install the [Confluence Crucible Plugin](#) in the Confluence instance (see [instructions on installing Confluence plugins](#)), and you must [set up a trust relationship](#) so that your Crucible instance trusts your Confluence instance.

[To set up Confluence as a Code Repository in Crucible,](#)

1. Start Crucible then open the '**Admin**' menu by clicking the *Administration* link in the footer of the page.
2. Under the '**Repository Settings**' heading, click '**Repository List**' in the left-hand navigation bar.
3. The '**Repository List**' screen opens.
4. Find the **Confluence** repository plugin and click its **Configure Plugin** link.
5. The '**Configure Plugin**' screen opens. Click '**Add Repository**'.
6. The '**Add Repository**' screen opens. Fill in the fields.

Configure Plugin

Name:

URL:

Space Key:

To restrict the repository to a single space, set the Space Key.

Field	What to enter
Name	Choose a unique name for the repository.
URL	Enter the URL of your Confluence instance.
Space Key	You may optionally enter a space key here to restrict Crucible's view to that key only. If there are many spaces in your Confluence instance you will find that this improves performance. You can set up several Confluence repositories in Crucible, each using the same Confluence instance but covering a different Space.

7. Click '**Save**'. The view will return to the list of repositories.
8. Now, access your Confluence instance. Open the '**Confluence Administration Console**', then select '**Trusted Applications**'. The Confluence '**Trusted Applications Details**' dialog opens.
9. In the '**Trusted Applications Details**' dialog, enter the URL of your Crucible instance into the '**Name**' field and click '**Send Request**'. The '**Application Alias**' will be automatically retrieved from Crucible. Save your changes.
10. Confluence is now set up as a code repository for Crucible. You will be able to select your Confluence from the list of repositories, then select files from the Confluence wiki and add them to reviews.

Best Practices for Crucible Configuration

1. **Set up a separate FISHEYE_INST folder location on the same system for Crucible's data.**

This will allow for easy upgrades of the core program and neatly separated data backup.

2. **Run Crucible on a dedicated machine, accessing its data on the local file system.**

This is the best environment for swift Crucible performance. Avoid running Crucible in a virtual environment.

3. **Do not give Crucible projects the same key as your JIRA projects.**

When naming projects, take care to ensure that the key you assign to them is not the same as any of your JIRA projects. The reason for this is, if one of your Crucible projects has the same key as one of your projects in JIRA, then all links with that key will lead back to Crucible, rather than leading to JIRA, removing the ability to navigate between the two applications.

To avoid this, name your Crucible project keys differently. For example, you could place the following text at the beginning of each project key: CR- to distinguish it. So, for this case, if you have an existing JIRA key of 'RHUBARB', you would create a Crucible key called 'CR-RHUBARB' so that they do not conflict.

Crucible Upgrade Guide

- [Upgrading to a New Version of Crucible](#)
- [Upgrading from FishEye to Crucible](#)
- [Upgrading from 1.6.x to 1.6.3.](#)

Upgrading to a New Version of Crucible

 Read about how your [Crucible installation works with FishEye](#).

- [Before you Start](#)
- [Crucible 2.0 Upgrade Notes](#)
- [Crucible 1.6 Upgrade Notes](#)
- [Upgrade Procedure](#)
 - [Method 1 - Using a FISHEYE_INST Directory](#)
 - [Method 2 - Without a FISHEYE_INST Directory](#)
 - [Method 3 - Without a FISHEYE_INST Directory, but would like to set one up](#)

Before you Start

- Before upgrading you should always read the [Release Notes](#) for the version you are upgrading to, as well as any versions you are skipping.
- **We strongly recommend you make a backup of your data before upgrading Crucible.** Refer to the documentation on [making a backup](#).
- [Download](#) the Crucible zip file.

Crucible 2.0 Upgrade Notes

- Supported browsers are: Safari 3+, FireFox 3+ and Internet Explorer 7+ (not IE6).

Crucible 1.6 Upgrade Notes


- Please see [Crucible 1.6.3 Upgrade Guide](#)

Upgrade Procedure

Method 1 - Using a FISHEYE_INST Directory

If you have Crucible configured to use a FISHEYE_INST directory, then simply:

1. Shutdown your existing fisheye server
2. Make a backup of your FISHEYE_INST directory
3. Extract the new Crucible version to a directory.
4. Leave your FISHEYE_INST environment variable set to its existing location.
5. Start Crucible from the new installation.
6. Follow any version-specific instructions found in the [Release Notes](#).

 Read more about the [FISHEYE_INST environment variable](#).

Method 2 - Without a FISHEYE_INST Directory

If you are not using FISHEYE_INST, you will need to copy some files from your old Crucible installation to your new one.

1. Extract the new Crucible archive into a directory such as /NEW_FISHEYE/.
2. Delete the /NEW_FISHEYE/var directory.
3. Shut down the old Crucible instance if it is running.
4. Copy /OLD_FISHEYE/config.xml to /NEW_FISHEYE/.
5. Copy (or move) the /OLD_FISHEYE/var directory to /NEW_FISHEYE/var.
6. If you have a Cenqua-issued Crucible license, copy all /OLD_FISHEYE/*.license files to /NEW_FISHEYE/. (Atlassian-issued licenses are included within config.xml.)
7. Follow any version-specific instructions found in the [Release Notes](#).

Method 3 - Without a FISHEYE_INST Directory, but would like to set one up

1. Shut down the old FishEye instance if it is running.
2. Set up the [FISHEYE_INST environment variable](#), then create the FISHEYE_INST directory on your filesystem.
3. Copy the /OLD_FISHEYE/config.xml to /FISHEYE_INST.
4. Copy the /OLD_FISHEYE/var directory to /FISHEYE_INST.
5. [Download](#) Crucible.
6. Extract the new Crucible archive into a directory such as /NEW_FISHEYE/.
7. Start Crucible from the new installation by running NEW_FISHEYE/bin/run.sh. (Use run.bat on Windows).
8. Follow the initial configuration steps outlined below.
9. If your configuration is not automatically picked up and you cannot see your existing repositories, check your Administration > Sys-Info page, where you will see information about FISHEYE_HOME and FISHEYE_INST. Check your FISHEYE_INST is pointing to the right

directory.

Upgrading from FishEye to Crucible

If you have been using FishEye and now want to move to Crucible, you can do this without losing your FishEye repositories.


 Read about how your [Crucible installation works with FishEye](#).

Before you Start

We strongly recommend you make a backup of your data before following the steps below. Refer to the documentation on [making a backup](#).

Upgrade Procedure

Follow **method 1** below if you have FishEye configured to use a `FISHEYE_INST` directory. Follow **method 2** if you are not using a `FISHEYE_INST` directory. Follow **method 3** if you are not using a `FISHEYE_INST` directory but would now like to start using one.

 Read more about the `FISHEYE_INST` [environment variable](#).

Method 1 - Using a FISHEYE_INST Directory

1. Shutdown your existing fisheye server
2. Make a backup of your `FISHEYE_INST` directory
3. [Download](#)Crucible and unzip the archive into a folder. This document assumes you have extracted your Crucible zip file into a directory called `/NEW_FISHEYE/`.
4. Leave your `FISHEYE_INST` environment variable set to its existing location.
5. Start Crucible from the new installation by running `NEW_FISHEYE/bin/run.sh`. (Use `run.bat` on Windows).
6. Follow the initial configuration steps outlined below.

Method 2 - Without a FISHEYE_INST Directory

1. [Download](#) Crucible.
2. Extract the new Crucible archive into a directory such as `/NEW_FISHEYE/`.
3. Delete the `/NEW_FISHEYE/var` directory.
4. Shut down the old FishEye instance if it is running.
5. Copy `/OLD_FISHEYE/config.xml` to `/NEW_FISHEYE/`.
6. Copy the `/OLD_FISHEYE/var` directory to `/NEW_FISHEYE/var`.
7. If you have a Cenqua-issued FishEye license, copy `/OLD_FISHEYE/fisheye.license` to `/NEW_FISHEYE/`. (Atlassian-issued licenses are included within `config.xml`.)
8. Follow any version-specific instructions found in the [Release Notes](#).
9. Start Crucible from the new installation by running `NEW_FISHEYE/bin/run.sh`. (Use `run.bat` on Windows).
10. Follow the initial configuration steps outlined below.

Method 3 - Without a FISHEYE_INST Directory, but would like to set one up

1. Shut down the old FishEye instance if it is running.
2. Set up the `FISHEYE_INST` [environment variable](#), then create the `FISHEYE_INST` directory on your filesystem.
3. Copy the `/OLD_FISHEYE/config.xml` to `/FISHEYE_INST`.
4. Copy the `/OLD_FISHEYE/var` directory to `/FISHEYE_INST`.
5. [Download](#) Crucible.
6. Extract the new Crucible archive into a directory such as `/NEW_FISHEYE/`.
7. Start Crucible from the new installation by running `NEW_FISHEYE/bin/run.sh`. (Use `run.bat` on Windows).
8. Follow the initial configuration steps outlined below.
9. If your configuration is not automatically picked up and you cannot see your existing repositories, check your Administration > Sys-Info page, where you will see information about `FISHEYE_HOME` and `FISHEYE_INST`. Check your `FISHEYE_INST` is pointing to the right directory.

Initial Configuration

1. You can access Crucible immediately by going to <http://HOSTNAME:8060/> in a browser.
2. The first time you run Crucible, you will be asked for a Crucible license key. You can view your license key [here](#).
3. If you do not already have user accounts configured, you will need to do this via the Administration screens or by configuring Crucible/FishEye to use external authentication.
To add users:
 - Open the [FishEye Administration screens](#) at <http://HOSTNAME:8060/admin/>.
 - Click 'Users/Security' under 'Global Settings' in the 'Admin Menu'.Read more details about the different ways of [creating users](#).
4. Crucible can email each review participant on a range of changes. Each user can then set up their own preferences. This is described in

the [User Profile](#) guide. First, you must set up the SMTP Server.

Crucible Release Notes



Crucible 2.0 has now been released. Read the [Release Notes](#).

Crucible Release Notes and Changelogs

- [Crucible Release Summary](#)
- [Crucible 2.0 Release Notes](#)
 - [Crucible 2.0 Changelog](#)
- [Crucible 2.0 Beta Release Notes](#)
 - [Upgrading to the Crucible 2.0 Beta](#)
 - [JIRA Integration in Crucible 2.0 Beta](#)
 - [Crucible 2.0 Beta Reviewer's Guide](#)
- [Crucible 1.6 Release Notes](#)
 - [Crucible 1.6 Changelog](#)
 - [Crucible 1.6.3 Upgrade Guide](#)
- [Crucible 1.5 Release Notes](#)
 - [Crucible 1.5 Changelog](#)
- [Crucible 1.2 Release Notes](#)
 - [Crucible 1.2 Upgrade Guide](#)
 - [Crucible 1.2 Changelog](#)
- [Crucible 1.1 Release Notes](#)
 - [Crucible 1.1 Upgrade Guide](#)
 - [Crucible 1.1 Changelog](#)
- For changes prior to 1.1, see the [1.0.x Changelog](#)

Installation

You can download Crucible from [here](#). Information on installing Crucible can be found [here](#).

If upgrading from a previous version, please follow the [Upgrade Guide](#).

- As of version 1.0, Crucible now requires a JVM version 1.5 or later. Previously, 1.4+ was required.
- Crucible 1.1.4 includes FishEye 1.3.8.
- Upgrading from 1.0.4 (or earlier) will force a complete re-index of P4 repositories.

Crucible Release Summary

Crucible 2.0 (30-Jun-09)

- Support for iterative reviews
- New User Interface
- Indicators for read/unread comments
- Enhanced JIRA integration
- More in [release notes](#).

Crucible 1.6 (23-Sep-08)

- Support for non-FishEye repositories
- Confluence page reviews
- Shared file system repositories
- Enhanced pre-commit reviews & image support
- Multiple admin users
- Expanded API
- More in [release notes](#).

Crucible 1.5 (14-Apr-08)

- Project Dashboard
- Filtered comments & defects search, with statistical summary

- Customisable email templates
- Improvements to Crucible Plugin API beta
- More in [release notes](#).

Crucible 1.2 (5-Dec-07)

- Reviews grouped into projects
- Customisable permission schemes
- Plugin API
- Enhancements to user management
- JIRA integration
- Crucible 1.2 includes FishEye 1.4
- More in [release notes](#).

Crucible 1.1 (18-Sep-07)

- Pre-commit review (patch review)
- Review participants can keep track of their progress through a review by marking each file as "done"
- Side-by-side diff mode within the Review display
- Syntax highlighting when displaying a diff
- More in [release notes](#).

Crucible 2.0 Release Notes

30 June 2009

Atlassian presents Crucible 2.0

Crucible 2.0 adds the all-new Iterative Reviews feature, enhanced JIRA integration and a brand new user interface.

Highlights of this release:

- [Iterative Reviews](#)
- [New User Interface](#)
- [Read/Unread comments](#)
- [Enhanced JIRA Integration](#)
- [Keyboard Shortcuts](#)
- [Review Activity](#)
- [External Databases and Backup](#)
- Plus numerous improvements and bug fixes

Thank you for your interest in Crucible 2.0.



See the documentation on [Upgrading to this version](#).

Installing Crucible 2.0

You can now download the Crucible 2.0 from [here](#). See the documentation on [Upgrading to this version](#).

Highlights of Crucible 2.0



Iterative Reviews

Crucible now allows you to review several revisions of a file within one review, seamlessly switching between them. Comments are linked and relative to a specific revision. This allows you to review every change that has occurred on a code file within a given period of time and hence visualise its evolution in the context of the review.

Screenshot: Iterative Reviews

The screenshot displays the Atlassian FishEye repository viewer interface for a code review. The top bar shows the repository path and the file being reviewed: `ElasticFunctionalityFacadeImplTest.java`. Below this, a revision timeline indicates three revisions: 92218, 92370, and 92643. The main area shows a diff view of the code, with line numbers and changes highlighted. A comment from Mark Chaimungkalanont is visible, asking, "Are we calling it instance not agent now?". The sidebar on the left provides navigation options and indicates the number of comments for each revision.

2

New User Interface

Taking on board wide-ranging feedback from customers, the Crucible team has completely revamped the user interface of the product, adding more views on your work and allowing you to access controls from multiple locations and allowing for different work styles. Files in review are arranged in a tree for easy navigation. New viewing modes for reviews support full screen view, side-by-side diff view and single or multiple file views.

Screenshot: Reviewed files

Mark files as reviewed to keep track of what you have done

Unreviewed files are bold

Reviewed files are not bold

'Completing' reviews as a reviewer helps others know your progress

Unread comments

3

Read/Unread comments

As you move around a review, Crucible keeps track of which comments you've seen and marks them as read. When you see a comment that you want to come back to, check the 'leave as unread' box so you don't forget it. Unlike an email thread, new comments are rarely at the bottom. That's why it's especially useful to filter and highlight just the new comments when you come back to a review that's underway.

Screenshot: Unread comments

Total read and (unread) comments

Top level comments

Comments in this file

Comments per revision

Unread comments

Keep as unread

4

Enhanced JIRA Integration

Crucible now has better JIRA integration, allowing you to see regular JIRA updates in your Crucible dashboard and create JIRA sub-tasks inline, without leaving the Crucible interface. You can still click on issue names to visit the JIRA instance they belong to, also. See [instructions for JIRA configuration](#).

Screenshot: Enhanced JIRA Integration

The screenshot shows the Crucible interface for a code review of 'review.js'. On the left, a sidebar lists subtasks under the review's linked issue. The main area shows the code review with comments from Sebastian Ruiz and Nicolas Venegas. A JIRA issue 'CRUC-1481' is linked to the review. Annotations on the right side of the screenshot highlight the following features:

- You can filter the review contents to only show files with JIRA subtasks or unresolved JIRA subtasks.
- Create an issue from any comment resolving is also a single click.
- The linked issue's state is displayed in the review.
- The issue links back to the comment is was created by.

Annotations on the left side of the screenshot highlight the following features:

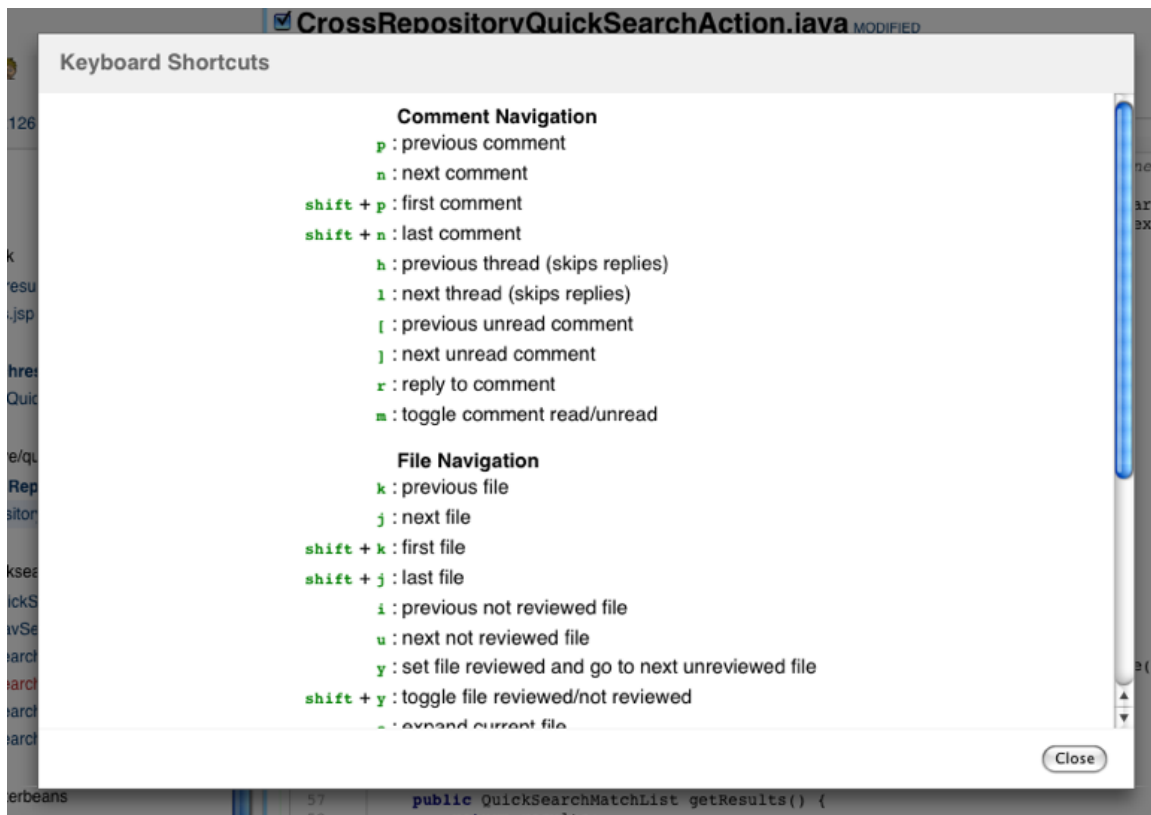
- Subtasks are created under a review's linked issue.
- The contents that do not have linked issues are grayed out.
- The JIRA issue's title and description are generated from the comment.

5

Keyboard Shortcuts

Crucible now has keyboard shortcuts for navigating around your reviews efficiently. No more repetitive strain injury from that mouse wheel.

Screenshot: Keyboard shortcuts



6

Review Activity

All the activity that happens in Crucible is available as an activity stream. Streams can be accessed per project as well as a personal stream that includes the activity from people, projects, reviews and even comments you favorite or are involved in.

Screenshot: Project review activity stream

CR-FE FishEye

https://extranet.atlassian.com/crucible/project/CR-FE?max=30&projectKey=CR-FE&view=cru&@

Dashboard Source Projects People Reviews Peter Moore Search

FishEye Tools

Activity Reviews

All Activity Commits Reviews Issues Show Revisions

Completions

Nicolas Venegas finished reviewing CR-FE-2137 12:40

commented on CR-FE-2137 12:40

Nice: this simplified a lot of the code.

Comments

Joe Xie commented on CR-FE-2147 12:40

looks good!

Nicolas Venegas commented on CR-FE-2137 12:26

Remove dead code?

`displayStyle` seems to be unused now.

New reviews

Tom Davies started review CR-FE-2147 12:19

FE-1823: use classes instead of ids, clean up code

Replies

Tim Pettersen replied to Erik van Zijst in CR-FE-2139 12:18

Ach, my bad - JIRA style is every brace on a new line, but I keep dropping back into FE/CRU (sun?) style.

Atlassian FishEye repository viewer with Crucible code review. (Version:2.0.0.RC3 Build:build-r38979 2009-06-29) - Administration - Page generated 2009-06-29 14:10 +1000

7

External Databases and Backup

Crucible now supports MySQL and Postgress in addition to the embeded HSQL database. Backup and restore capabilities have also been greatly enhanced.

8

Plus numerous improvements and bug fixes

Visit our [issue tracker](#) to see the full list of improvements and bug fixes.

Crucible 2.0 Changelog

On this page:








































- From 2.0 Beta3 to 2.0
- From 2.0 Beta2 to 2.0 Beta3
- From 1.6.6 to 2.0 Beta2
-


















































From 2.0 Beta3 to 2.0







30th June 2009


















































Full list of issues in this release:

JIRA Issues (190 issues)




































Key	Summary	Priority	Status
CRUC-1616	Tune polling period.		 Closed
CRUC-1615	Reset the delay when activetly using the review		 Closed
CRUC-1612	Move the warning to a Gmail-style overlay		 Closed
CRUC-59	External database		 Closed
CRUC-1656	RC3 - NPE when trying to create first project on empty CRU install		 Closed
CRUC-824	If we haven't already we should automatically create a backup of our CRUDB (and other configuration files) before upgrading		 Closed
CRUC-1661	File marked as unreviewed for defect author		 Closed
CRUC-1651	clicking [add] branch in SVN admin symbolic screen causes page-pop and 404		 Closed
CRUC-1649	toggling Display Preferences > Show Source in a review does nothing		 Closed
CRUC-1644	Some display preferences don't update when clicked		 Closed
CRUC-1641	error mapping JIRA servers		 Closed
CRUC-1623	Admin: Default reviewers are not re-saved		 Closed
CRUC-1611	Commenting doesn't work in Safari 4		 Closed
CRUC-1609	CAC documentation for iterative review info from CRUC-1536		 Closed
CRUC-1481	Warning dialogs and polling when new stuff added since last refresh		 Closed
CRUC-1404	Review throws 500 error NPE		 Closed
CRUC-1403	Rename "crucibledb" to "sql"		 Closed
CRUC-1399	Default Reviewer for project is not exposed through the SPI		 Closed
CRUC-1398	overdue but review has not been started. Setting a due date in the past causes this.		 Closed
CRUC-1393	Refactor Email Comments Page into a Dialog		 Resolved
CRUC-1392	fix remove file behaviour of current content panel		 Closed
CRUC-1391	Use newDirTreeBox in the files pane		 Closed
CRUC-1390	Fix Uploads		 Closed
			









CRUC-1388	Missing css files on Send Email page / Layout needs love		Closed
CRUC-1386	fix suggest reviewer ui		 Closed
CRUC-1383	suggest reviewers - turn blame calculation back on		 Closed
CRUC-1382	defect metrics charts are broken		 Closed
CRUC-1381	.png file not detected as binary		 Closed
CRUC-1380	Email Review button from the dashboard doesnt work / Email review in tols menu gives a 404 - permaid is undefined		 Closed
CRUC-1379	clicking HELP opens help twice / Help in Tools menu will open a new window and change your window		 Closed
CRUC-1378	page load error "The requested URL /atlaseye/static/kidi0j/2static/script/fe/fisheye-profile.js was not found on this server."		 Closed
CRUC-1377	wrong review actions (comments.txt and Email Review) are visible		 Closed
CRUC-1376	we need to handle multiple errors in a smarter way		 Closed
CRUC-1375	There is no jira quick link in the edit review dialog		 Closed
CRUC-1374	Review info popup on the dashboard shouldn't show when clicking on a link such as the review permaid		 Closed
CRUC-1373	Page load error "The requested URL /atlaseye/static/kidi0j/2static/images/sprite_arrows.png was not found on this server"		 Closed
CRUC-1371	External dialog 'View ...' links need to redirect to review and trigger filter		 Closed
CRUC-1370	Show current content panel in manage files		 Closed
CRUC-1369	make the side filters like the fisheye tabs		 Closed
CRUC-1368	bring back the custom filter		 Closed
CRUC-1367	make the pages consistent		 Closed
CRUC-1366	Add warnings for incomplete FRXs and unresolved JIRAs to dialogs		 Closed
CRUC-1365	Move review summary form into a dialog.		 Closed
CRUC-1364	Move creation-time review suggestions into a dialog		 Closed
CRUC-1363	Convert project selection screen to a dialog		 Closed
CRUC-1362	Backup broke Crucible Blitz		 Closed
CRUC-1361	General Issues		 Resolved
CRUC-1360	Manage Files		 Resolved









































CRUC-1359	Edit review form		 Resolved
CRUC-1358	Interstitials/dialogs		 Resolved
CRUC-1357	Dashboard Fixes		 Resolved
CRUC-1356	Remove driver url parameter from Restore and use new DBInfo properties		 Closed
CRUC-1354	Refactor programatic frx filtering to not invoke click events		 Resolved
CRUC-1351	Add path and base filename to backup UI		 Resolved
CRUC-1345	Extend QuartzManager to support persistent configuration data per job		 Closed
CRUC-1344	Rest Api returns the wrong URL for download Patch		 Closed
CRUC-1330	Put custom quartz job scheduling in a separate tab		 Closed
CRUC-1329	Dynamically update general comment counts		 Closed
CRUC-1328	Add general comments to the tree		 Closed
CRUC-1327	Add the BackupItem selection to the scheduled backup		 Closed
CRUC-1325	Add file type css place holders		 Closed
CRUC-1319	Update doco for admin backup page		 Resolved
CRUC-1317	Dynamically update comment counts		 Closed
CRUC-1316	Add comment counts to the frx tree		 Closed
CRUC-1305	Springify Backup Manager		 Closed
CRUC-1303	Write unit test		 Closed
CRUC-1294	Check start/stop no longer needed in restore		 Closed
CRUC-1293	Render maintenance page for non-admin pages when a backup is being made		 Closed
CRUC-1291	Add "Change Diff" dropdown to manipulate both fromRev and toRev independently		 Resolved
CRUC-1290	Collapse common directories		 Closed
CRUC-1289	Apply tree html structure and css styling to list		 Closed
CRUC-1288	Treeify FRX list pane		 Resolved
			

CRUC-1286	add other filters		Closed
CRUC-1285	add filter - with comments		 Closed
CRUC-1284	add UI for dropdown and expand/collapse		 Closed
CRUC-1283	Drop-down filter for frx tree		 Resolved
CRUC-1275	Convert stored dates to long		 Closed
CRUC-1274	Create FRX anchors and adjust the browser location as you navigate		 Resolved
CRUC-1273	FRXs should start expanded and maintain position when loading frxs above		 Closed
CRUC-1271	Move edit review box into AJS.Dialog		 Closed
CRUC-1270	Integrate the summarise and close pages		 Closed
CRUC-1254	Rearrange content		 Closed
CRUC-1253	Review meta information sub pane		 Closed
CRUC-1252	Move manage files tabs into AJS.Dialog		 Closed
CRUC-1251	scroll frx pane when clicking on frx list item		 Closed
CRUC-1250	create floating frx info subpane which updates on scroll		 Closed
CRUC-1248	Modify overview header		 Closed
CRUC-1247	3 Pane UI		 Resolved
CRUC-1242	Delete the old backup mechanism		 Resolved
CRUC-1241	Rewire the quartz job		 Closed
CRUC-1235	Revision slider hardening/tweaking		 Resolved
CRUC-1232	Delete comment refactor		 Closed
CRUC-1215	Quartz backup job must invoke the new xml-based backup		 Resolved
CRUC-1208	reorganise contents of static directory		 Closed
CRUC-1206	add new preference for new frxrevisions		 Closed
CRUC-1205	add new notification for adding frxRevisions		 Closed
CRUC-1179	Sort FRX - alpha server side sort		 Resolved

CRUC-1171	XML-RPC bollocks		 Closed
CRUC-1170	Delete a defect's linked JIRA subtask when deleting the defect		 Resolved
CRUC-1169	ui & action		 Closed
CRUC-1168	XML-RPC bollocks		 Closed
CRUC-1167	Resolve a defect's linked JIRA subtask from Crucible		 Resolved
CRUC-1166	XML-RPC bollocks		 Closed
CRUC-1165	UI display of jira link and status + post error creating		 Closed
CRUC-1164	schema		 Closed
CRUC-1163	Automatically add defects as subtasks to JIRA & link to them		 Resolved
CRUC-1162	guessing + jira validation		 Closed
CRUC-1161	edit details ui + actions		 Closed
CRUC-1160	schema		 Closed
CRUC-1159	Link a review to a single JIRA and enable auto defect creation		 Resolved
CRUC-1158	just do it		 Closed
CRUC-1157	Configure a JIRA instance for use by Crucible		 Resolved
CRUC-1156	admin UI + create review		 Closed
CRUC-1155	schema		 Closed
CRUC-1154	Configure a default review duration for a project in week days so the due date can be pre-populated		 Resolved
CRUC-1153	highlighting overdue		 Closed
CRUC-1152	sorting in the lists		 Closed
CRUC-1151	Order any list of reviews by due date & highlight any that are overdue		 Resolved
CRUC-1150	due date field in UI		 Closed
CRUC-1149	schema		 Closed
CRUC-1148	Set or change a due date on a review		 Resolved
			

CRUC-1143	just do it		Closed
CRUC-1142	Combine diff and annotation views		 Resolved
CRUC-1141	just do it		 Closed
CRUC-1140	show the reviews you can't update without buttons		 Resolved
CRUC-1139	Author colours		 Closed
CRUC-1138	Invisible markers (between lines)		 Closed
CRUC-1137	Mouseover		 Closed
CRUC-1136	Improve comment marker behaviour (tetris bar)		 Resolved
CRUC-1134	replace yahoo shite if we can		 Closed
CRUC-1133	event binding		 Closed
CRUC-1132	error handling		 Closed
CRUC-1131	namespacing & multiple files		 Closed
CRUC-1130	remove prototype		 Closed
CRUC-1129	client side js model		 Closed
CRUC-1128	refactor commentator.js and friends		 Resolved
CRUC-1127	just do it		 Closed
CRUC-1126	handle modifier keypress		 Resolved
CRUC-1125	just do it		 Closed
CRUC-1124	Popup a keyboard shortcut help overlay when i press '?'		 Resolved
CRUC-1123	add user pref		 Closed
CRUC-1122	add complete status resetter to the defect notification		 Closed
CRUC-1121	Reset my complete status when new defects are added to a review		 Resolved
CRUC-1119	Tweak comment scroll tracker behaviour		 Resolved
CRUC-1118	Improve revision slider behaviour		 Resolved
CRUC-1117	add unread comment counts per revision		 Closed





CRUC-1114	fix rendering/script issue in IE		 Closed
CRUC-1113	make the slider handles move in unison		 Closed
CRUC-1108	Mark comment read when it hits top of the viewport		 Closed
CRUC-1097	Remove the use of half closed TCP connections		 Resolved
CRUC-1083	render author + participant results		 Resolved
CRUC-1048	Change the admin backup page to use the new backup/restore implementation		 Resolved
CRUC-927	Add a warning in the product that re-sync in large LDAP servers can cause performance issue		 Closed
CRUC-922	Admin screens with config.xml write and create/migrate workflow		 Closed
CRUC-921	Set up maintenance Mode		 Closed
CRUC-920	set up tests to compare upgrades of each database type		 Closed
CRUC-918	Write tests to determine/compare expected results from database tables		 Closed
CRUC-915	Usage Documentation for web-based backup/restore feature		 Closed
CRUC-912	Integration Tests		 Closed
CRUC-910	Admin Backup Page		 Closed
CRUC-909	Backup Status Page		 Closed
CRUC-907	Online Backup/Restore		 Closed
CRUC-882	REST API return other people's drafts		 Closed
CRUC-861	Allow soft wrap on diff and annotated display		 Closed
CRUC-845	Describe this in the CAC docs.		 Closed
CRUC-823	Add an Expand Unchecked Files option		 Closed
CRUC-770	Support file upload via REST		 Closed
CRUC-603	Support Anonymous Users Properly		 Closed
CRUC-581	Include Light SCM Repos in information returned by REST API		 Closed
CRUC-466	Improve built-in group management		 Closed
			

CRUC-441	Revisit search result layout		 Closed
CRUC-440	Simplify workflow & terminology		 Closed
CRUC-420	Nested comments within review		 Closed
CRUC-416	Got fisheye and crucible problems.. not loading		 Closed
CRUC-403	XWork Action Plugin Modules		 Closed
CRUC-272	API for plug-ins responding to events like creation of a review and so on...		 Closed
CRUC-254	Very Slow Performance on large source files.		 Closed
CRUC-211	JIRA Integration		 Closed
CRUC-126	Better Multi-Commit Reviews		 Closed
CRUC-116	Iterative Review		 Closed
CRUC-114	Due date & Reminders		 Closed
CRUC-29	After page-ordering is implemented on CAC, remove 'manual' page numbers		 Closed
CRUC-858	"Lines are missing" divider graphics are no longer there		 Closed
CRUC-467	Poor performance when creating a review via Internet Explorer browser		 Closed
CRUC-431	Move review to another project - again		 Closed
CRUC-377	Integrate Jersey with Spring		 Closed
CRUC-277	Delete patches and uploaded files		 Closed
CRUC-1445	STUPID: old style error page when trying to access fisheye pages		 Closed
CRUC-1442	Crucible Standalone - Logout screen has FishEye icon & link		 Closed
CRUC-537	Summary Report & Searching by File		 Closed

From 2.0 Beta2 to 2.0 Beta3

5th June 2009

Full list of issues in this release:














JIRA Issues (35 issues)			
Key	Summary	Priority	Status
CRUC-1501	1.6 -> 2.0 beta upgrade truncates timestamps		 Closed
CRUC-1496	FRX reloads cause review model sort problems		 Closed

CRUC-1493	Show a message in the frx pane when all files are filtered		 Closed
CRUC-1491	I think it should be sticky. i.e. set your preference each time you toggle it.		 Closed
CRUC-1524	review actions from hovers sometimes aren't bound and dont work.		 Closed
CRUC-1515	File time stamps are not preserved in the backup		 Closed
CRUC-1513	Permalinks to hidden comments are broken		 Closed
CRUC-1512	Crucible user preferences should be saved automatically when changed		 Closed
CRUC-1508	Pressing 'y' in fullscreen mode skips an frx		 Closed
CRUC-1507	Next/Prev FRX buttons		 Closed
CRUC-1506	Pressing 'y' in review comments doesn't do anything.		 Closed
CRUC-1505	Marking a file as reviewed should mark all frx comments as read		 Closed
CRUC-1504	Error dialog box background css shows entire sprite		 Closed
CRUC-1503	Defect subtasks have their assignee set to default assignee when resolved from Crucible		 Closed
CRUC-1499	IDEA direct link port changed from 6666 to 51234		 Closed
CRUC-1492	Files from different repos breaks Manage Files		 Closed
CRUC-1487	Need to show initial sort order on the Review dashboard		 Closed
CRUC-1486	Adding a changeset to a review results in duplicate FRXs		 Closed
CRUC-1484	Linked subtask in comment doesn't have a JIRA hover		 Closed
CRUC-1480	Revision slider doesn't resize properly when you edit revisions		 Closed
CRUC-1477	Clicking on a source line when creating a file comment kills the file comment		 Closed
CRUC-1475	Moving slider on image diff causes NPE		 Closed
CRUC-1474	Suggested reviews dialog uses old style inline review dialog		 Closed
CRUC-1473	Review actions missing from inline review dialog		 Closed
CRUC-1469	"You do not have permission to see all the search results." Looks crap.		 Closed
CRUC-1461	Unable to add source file to review		 Closed
CRUC-1460	fix styling of crucible project side bar		 Closed
CRUC-1446	FishEye activity in activity stream when in CruOnly mode		 Closed
CRUC-1438	Review Layout Issues		 Closed
CRUC-1425	Folders show in review file list		 Closed
CRUC-1323	Bug in adding iterative revisions		 Closed
CRUC-1315	Suggestions are not accurate		 Closed
CRUC-1224	keep slider visible during frx reload		 Closed
CRUC-1434	Toolbar buttons don't hover and they dont show a cursor		 Closed
CRUC-1429	Tooltip over the top of Review hover		 Closed

From 1.6.6 to 2.0 Beta2


Full list of issues in this release:


--


JIRA Issues (13 issues)			
Key	Summary	Priority	Status
CRUC-1488	Improve user interface in the frx tree	↑	 Closed
CRUC-1478	Single frx visible mode	↑	 Closed
CRUC-1476	File revision comments aren't autosaved	↑	 Closed
CRUC-1465	Synchronise comment buttons when states are changed	↑	 Closed
CRUC-1464	Fix rev slider labels so that slider can snap	↑	 Closed
CRUC-1463	User has no display name	↑	 Closed
CRUC-1462	Selecting changeset in Manage Files fails	↑	 Closed
CRUC-1458	Account signup pages still use old styles	↑	 Closed
CRUC-1454	Timezone problem in review popup	↑	 Closed
CRUC-1439	Crucible Standalone - Manage Files links to FishEye	↑	 Closed
CRUC-1396	Empty reviews cannot be abandoned/closed/viewed	↑	 Closed
CRUC-1116	No notifications sent for comments created through REST	↑	 Closed
CRUC-1427	Old error page for disabled FishEye Urls	↓	 Closed

Crucible 2.0 Beta Release Notes

Crucible 2.0 Beta is a public development release leading up to **Crucible 2.0**. For all production use and testing of Crucible, please use [the latest official release](#).

 This page refers to an updated version of the Beta (Beta 3). We strongly recommend all beta users [upgrade to this release](#).

 Do not use in production.
Beta releases should not be used in production environments.

 Please also take note of the following information:

- Beta releases are not safe — Beta releases are snapshots of the ongoing Crucible development process. As such:
 - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
 - Features in development releases may be incomplete, may change or be removed before the next full release.
 - Firefox 3 and Safari are the only browsers supported.

5 June 2009

Atlassian presents Crucible 2.0 Beta

Crucible 2.0 adds the all-new Iterative Reviews feature, enhanced JIRA integration and a brand new user interface.

Highlights of this release:

- Iterative Reviews
- Enhanced JIRA Integration
- Stars Feature
- Unique Avatars
- People View
- New User Interface
- Plus numerous improvements and bug fixes

Thank you for your interest in Crucible 2.0 Beta.



See the documentation on [Upgrading to this version](#).

Installing Crucible 2.0 Beta

You can now download the Crucible 2.0 Beta from [here](#). See the documentation on [Upgrading to this version](#).

Highlights of Crucible 2.0 Beta

1

Iterative Reviews

Crucible now allows you to review several revisions of a file within one review, seamlessly switching between them. Comments are persistent and relative to a specific revision. This allows you to review every change that has occurred on a code file within a given period of time and hence visualise its evolution in the context of the review.

Screenshot: *Iterative Reviews*

The screenshot shows the Crucible web interface for a review titled "CR-CLOV-136 Test Review for Documentation". The review is under review for 8 days, with an overdue date of 15 days, 4 comments, and 1 reviewer. The review is managed by Edwin Dawson. The interface shows a timeline of revisions for the file "ElementInfo.java". The revisions are 32288, 35536, and 36696. The current revision (36696) is selected, showing the code for "ElementInfo.java". The code is a Java class that implements "CoverageDataReceiver" and "SourceRegion". It has a "private ContextSet context;" and a "private int relativeDataIndex;". The interface also shows a sidebar with a tree view of the project structure, including "trunk", "core/src/com/cenqua/clover/registry", and "idea7/src/com/cenqua/clover/idea".

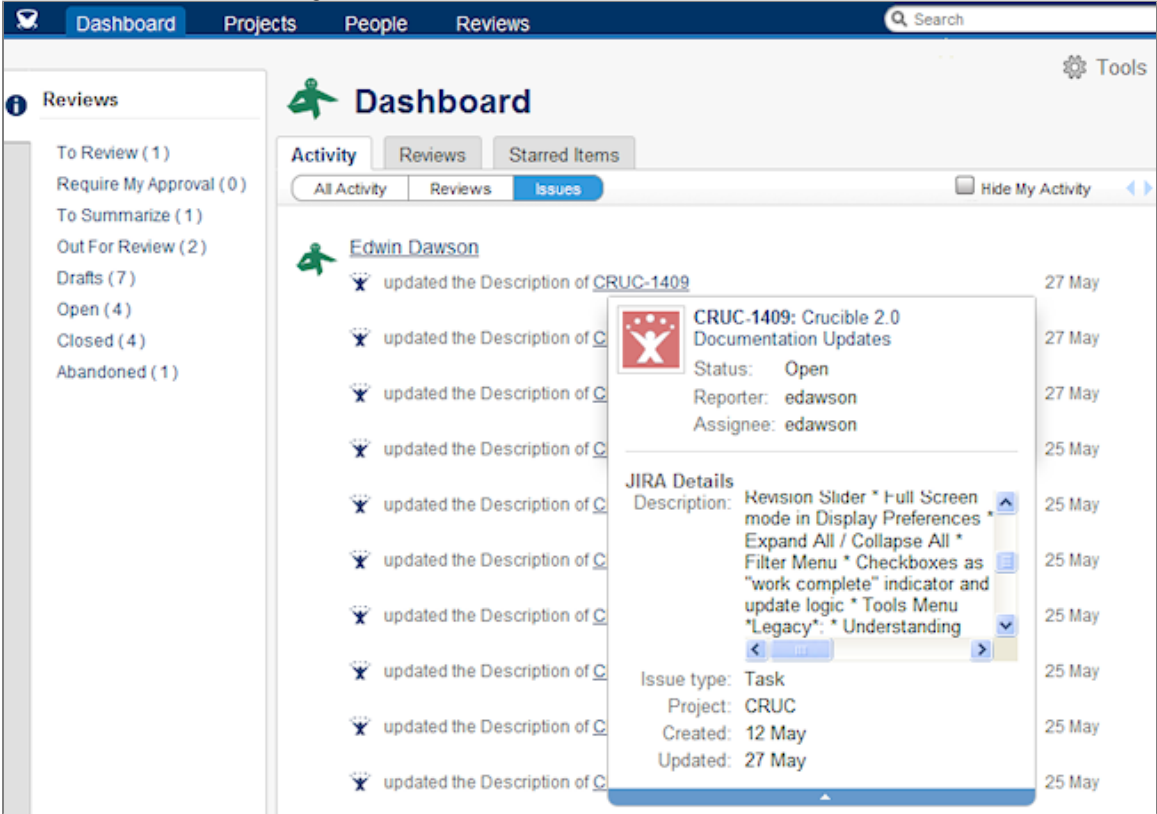
2

Enhanced JIRA Integration

Crucible now has better JIRA integration, allowing you to see regular JIRA updates in your Crucible dashboard and create JIRA sub-tasks inline,

without leaving the Crucible interface. You can still click on issue names to visit the JIRA instance they belong to, also. See [instructions for JIRA configuration](#).

Screenshot: Enhanced JIRA Integration

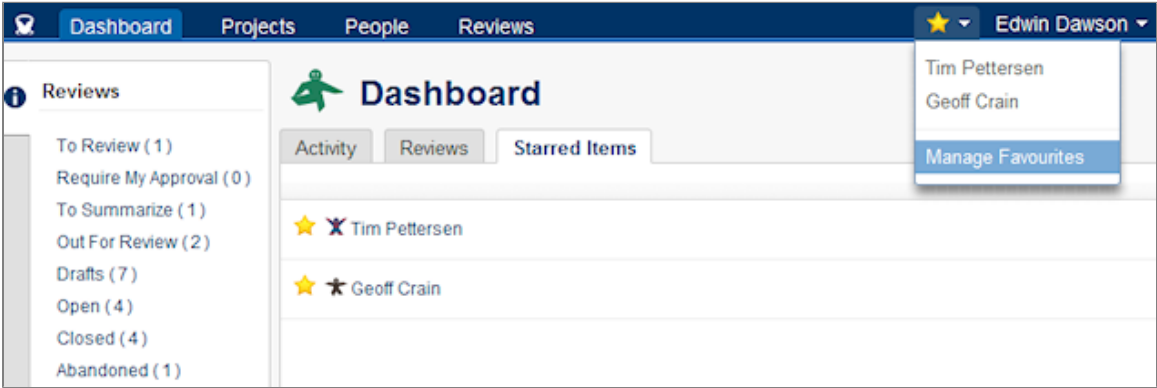


3

Stars Feature

Crucible now allows you to create a list of "Starred" favourite items that includes reviews, JIRA issues, colleagues and more. These favourites can be viewed together in aggregate as a stream of active work you are doing.

Screenshot: Stars




















4

Unique Avatars

Crucible will now generate a unique "Charlie" image that will be used as your avatar in the system. These avatars create a new visual linkage to the personalities working on various items and are used as a visual shorthand to show user involvement on items in menu screens and dialogs.

Screenshot: Unique Avatars
















Due	Reviewers
15 days ago	
11 days ago	
10 days ago	   
3 days ago	   
4 hours ago	  
3 hours ago	 
79 minutes ago	 



People View

You can now view detailed charts and activity statistics people who use your Crucible instance. You can compare number of reviews complete and other metrics charted over time.

Screenshot: People View

Dashboard Projects People Reviews <input type="text" value="Search"/>			
 All Users			
<div> <div></div> <div>1</div> <div>2</div> <div>3</div> <div>4</div> <div>5</div> <div>6</div> <div>7</div> <div>▶</div> </div>			
User	Latest Activity ^	Reviews	
 Geoff Crain	 on 29 May 2009	516	
 Erik van Zijst	 on 29 May 2009	459	
 Craig Sharkie	 on 29 May 2009	71	
 Conor MacNeill	 on 29 May 2009	461	
 Nick Pellow	 on 29 May 2009	195	
 Sebastian Ruiz	 on 29 May 2009	375	
 Joe Xie	 on 29 May 2009	231	
 Peter McNeil	 on 29 May 2009	517	

6

New User Interface

Taking on board wide-ranging feedback from customers, the Crucible team has completely revamped the user interface of the product, adding more views on your work and allowing you to access controls from multiple locations and allowing for different work styles. Files in review are arranged in a tree for easy navigation. New viewing modes for reviews support full screen view, side-by-side diff view and single or multiple file views.

Screenshot: New User Interface

The screenshot displays the Crucible 2.0 Beta web interface for a code review. At the top, there's a navigation bar with tabs for Dashboard, Projects, People, and Reviews. The current view is 'Reviews', showing a review titled 'CR-CLOV-136 Test Review for Documentation' by Edwin Dawson. The review is under review for 8 days, with 4 comments and 1 reviewer. The left sidebar shows a file tree with 'MethodInfo.java' selected. The main area shows the code for 'MethodInfo.java' with line numbers and a diff view. The code includes imports for List, Set, Serializable, ObjectInputStream, and IOException, followed by a class definition for MethodInfo that extends ElementInfo and implements HasMetric. The code is highlighted with green and red background colors to indicate changes.



Plus numerous improvements and bug fixes

Visit our [issue tracker](#) to see to see the full list of improvements and bug fixes between Beta 2 and Beta 3. We strongly recommend all beta users upgrade to the latest beta release.

See the [Beta Reviewer's Guide](#) for a list of known issues and guidance on the beta experience.

Upgrading to the Crucible 2.0 Beta

Crucible 2.0 Beta is a public development release leading up to **Crucible 2.0**. For all production use and testing of Crucible, please use the [latest official release](#).



Do not use in production.
Beta releases should not be used in production environments.



Please also take note of the following information:

- Beta releases are not safe — Beta releases are snapshots of the ongoing Crucible development process. As such:
 - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
 - Features in development releases may be incomplete, or may change or be removed before the next full release.
 - There will be an upgrade path from the 2.0 Beta to the final release.

This page contains instructions on how to upgrade your Crucible instance to the Crucible 2.0 Beta.

 Read about how your [Crucible installation works with FishEye](#).

Before you Start


- Before upgrading you should always read the [Release Notes](#) for the version you are upgrading to, as well as any versions you are skipping.
- **We strongly recommend you make a backup of your data before upgrading Crucible.** Simply make a copy of your `crucible_install_dir/var/data/` directory.
- [Download](#) the Crucible zip file.

Upgrade Procedure

Method 1 - Using a FISHEYE_INST Directory

If you have Crucible configured to use a `FISHEYE_INST` directory, then simply:

1. Shutdown your existing fisheye server
2. Make a backup of your `FISHEYE_INST` directory
3. Extract the new Crucible version to a directory.
4. Leave your `FISHEYE_INST` environment variable set to its existing location.
5. Start Crucible from the new installation.
6. Follow any version-specific instructions found in the [Release Notes](#).

 Read more about the `FISHEYE_INST` [environment variable](#).

Method 2 - Without a FISHEYE_INST Directory

If you are not using `FISHEYE_INST`, you will need to copy some files from your old Crucible installation to your new one.

1. Extract the new Crucible archive into a directory such as `/NEW_FISHEYE/`.
2. Delete the `/NEW_FISHEYE/var` directory.
3. Shut down the old Crucible instance if it is running.
4. Copy `/OLD_FISHEYE/config.xml` to `/NEW_FISHEYE/`.
5. Copy (or move) the `/OLD_FISHEYE/var` directory to `/NEW_FISHEYE/var`.
6. If you have a Cenqua-issued Crucible license, copy all `/OLD_FISHEYE/*.license` files to `/NEW_FISHEYE/`. (Atlassian-issued licenses are included within `config.xml`.)
7. Follow any version-specific instructions found in the [Release Notes](#).

Method 3 - Without a FISHEYE_INST Directory, but would like to set one up

1. Shut down the old FishEye instance if it is running.
2. Set up the `FISHEYE_INST` [environment variable](#), then create the `FISHEYE_INST` directory on your filesystem.
3. Copy the `/OLD_FISHEYE/config.xml` to `/FISHEYE_INST`.
4. Copy the `/OLD_FISHEYE/var` directory to `/FISHEYE_INST`.
5. [Download](#) Crucible.
6. Extract the new Crucible archive into a directory such as `/NEW_FISHEYE/`.
7. Start Crucible from the new installation by running `NEW_FISHEYE/bin/run.sh`. (Use `run.bat` on Windows).
8. Follow the initial configuration steps outlined below.
9. If your configuration is not automatically picked up and you cannot see your existing repositories, check your Administration > Sys-Info page, where you will see information about `FISHEYE_HOME` and `FISHEYE_INST`. Check your `FISHEYE_INST` is pointing to the right directory.

Crucible 2.0 Beta Reviewer's Guide

Crucible 2.0 Beta is a public development release leading up to **Crucible 2.0**. For all production use and testing of Crucible, please use [the latest official release](#).



Do not use in production.
Beta releases should not be used in production environments.



Please also take note of the following information:

- Beta releases are not safe — Beta releases are snapshots of the ongoing Crucible development process. As such:
 - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
 - Features in development releases may be incomplete, or may change or be removed before the next full release.
 - FireFox 3 and Safari are the only browsers supported.

Thank you for your interest in the Crucible 2.0 Beta. This page contains some direction on what is ready for testing, what the known issues are and how you can submit feedback.

Known Issues

This is a list of known issues with the Crucible 2.0 Beta; please do not raise requests related to these as solutions for them are already under way.

- REST API updates are not fully functional; you can't access various features from the IDE Connectors or other technology that depends on REST. The features affected include marking comments as read, Iterative Reviews and Activity Streams. Also, JSON support is alpha at this stage.
- JIRA sub-tasks integration will not work on JIRA instances where the JIRA General Setting "allow unassigned issues" is set to 'OFF'.
- Sorting functionality is only partially functional; when you are trying to sort content — such as review lists on the reviews page, content on the dashboard and project dashboard — the sorting of content by name, date and so on only applies to the current page of results. To mitigate this, narrow down your searches so that you only have one page of results. This will be revised to have server-side sorting.
- You can't create reviews with files or changesets from multiple repositories.

Features Ready For Testing

The following features in the Crucible 2.0 Beta are relatively hardened and using these thoroughly will help contribute to the final product.

- Iterative Reviews; create a review on multiple revisions of a changeset, to see the history and evolution of the code in-line.
- Activity Streams; see review activity, code commits and JIRA activity (when properly configured) on the Dashboard.
- External Database Support; You can now store Crucible's internal data in a MySQL or PostgreSQL database, as an alternative to the built-in HSQLDB.
- Crucible can assist you in updating reviews by suggesting changesets that are likely related to your reviews.
- Reviews can now have a Due Date. Reviews that are overdue will show up in red on the reviewer's dashboards to minimise the chance of reviews getting stale.
- Stars; add colleagues, reviews and files to your favourites list, then view updates related to them as a feed.
- Charlietars; the automatically generated Crucible avatars should work smoothly. Also, you can sign up to Globally Recognised Avatars (<http://www.gravatar.com>) to upload a profile image and use that instead of the Charlie image.
- Scheduled Backups; you can now easily set Crucible to backup data periodically using a simple calendar function in the user interface.

Submitting feedback

To submit feedback on the Crucible 2.0 Beta, please use the [Crucible Forums](#).

JIRA Integration in Crucible 2.0 Beta

Crucible 2.0 Beta is a public development release leading up to **Crucible 2.0**. For all production use and testing of Crucible, please use [the latest official release](#).




Do not use in production.
Beta releases should not be used in production environments.




Please also take note of the following information:

- Beta releases are not safe — Beta releases are snapshots of the ongoing Crucible development process. As such:
 - While we try to keep these releases stable, they have not undergone the same degree of testing as a full release.
 - Features in development releases may be incomplete, or may change or be removed before the next full release.

This page contains instructions for setting up JIRA integration in Crucible.

 JIRA is Atlassian's issue tracking product, which can be used to manage projects and associated work.

 Before you begin: Ensure that you configure your JIRA instance to [enable sub-tasks](#), [enable unassigned issues](#) and allow [Remote API access](#). The instructions on this page have been tested with JIRA 3.13.4.

On this page:


- [Opening the Administration Screen for JIRA Integration](#)
- [Adding a New JIRA Server](#)
 - [Obtaining the Subtask Type ID](#)
 - [Obtaining the Subtask Resolution ID and Subtask Resolution Action ID](#)
- [Editing Default JIRA Server Mappings](#)
- [Operations on Existing Servers](#)
 - [Edit settings for an existing JIRA server](#)
 - [Edit mappings for an existing JIRA server](#)
 - [Delete an existing JIRA server](#)

JIRA issues can be viewed in the main Dashboard view in Crucible. This requires you to enter details on the required JIRA server(s) via the Crucible administration screens.

Opening the Administration Screen for JIRA Integration

To set up JIRA integration, open the Administration screen and then click '**JIRA Servers**' under the '**Global Settings**' sub-menu on the left navigation bar. The '**View JIRA Servers**' administration page opens.

Screenshot: The View JIRA Servers Page

View JIRA Servers 								
Name	Url	Subtask ID	Subtask Resolution Action ID	Subtask Resolution ID	Allow Unassigned	Activity Stream Provider	Trusted Application	
jac	https://jira.atlassian.com	6	2	6	true	true	false	Edit Mappings Delete
Add a JIRA Server Edit default JIRA Server mappings								

On the View JIRA Servers page, you can carry out a number of operations as listed on this page.

Adding a New JIRA Server

To add a new JIRA server from the View JIRA Servers page, click '**Add JIRA Server**'.

The '**Add JIRA Server**' page opens.

Screenshot: The Add JIRA Server Page

Add JIRA Server

Server Details

Name:
URL:

Default Subtask Settings (leave blank to disable subtasks)

Subtask Type ID:
Subtask Resolution Action ID:
Subtask Resolution ID:

Allow Unassigned:
Yes No

Default Authentication

Username:
Password:

Options

☐ Include in Activity Streams
☐ Authenticate as Trusted Application


Test
Save
Cancel

A number of fields and options must be filled out or selected on this page. See the table below for information on each field.

Option	Type	Description	Required
Name	Text Field	A descriptive name for the JIRA server.	Yes
URL	Text Field	The Internet address of the JIRA server.	Yes
Subtask Type ID	Number	This is required to enable creating issues from a Crucible comment.	No
Subtask Resolution Action ID	Number	This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.	No
Subtask Resolution ID	Number	This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.	No
Allow Unassigned	True/False Button	Allow unassigned sub-tasks.	No
Username	Text Field	The username of an account on the JIRA instance (All activity that takes place will be attributed to this user, unless using the Trusted Application setting).	Yes
Password	Text Field	The password for the account on the JIRA instance.	Yes
Include in Activity Streams	Check Box	Allows JIRA information to appear on the Dashboard.	No
Authenticate as Trusted Application	Check Box	Allows the system to interface with JIRA and let users log on with their own accounts (and use their own accounts on the JIRA server. See complete FishEye documentation and complete JIRA documentation .	No

Once you've filled out the necessary fields, click **'Test'** to ensure that your details are correct. If you have a positive message return from the test, click **'Save'**.

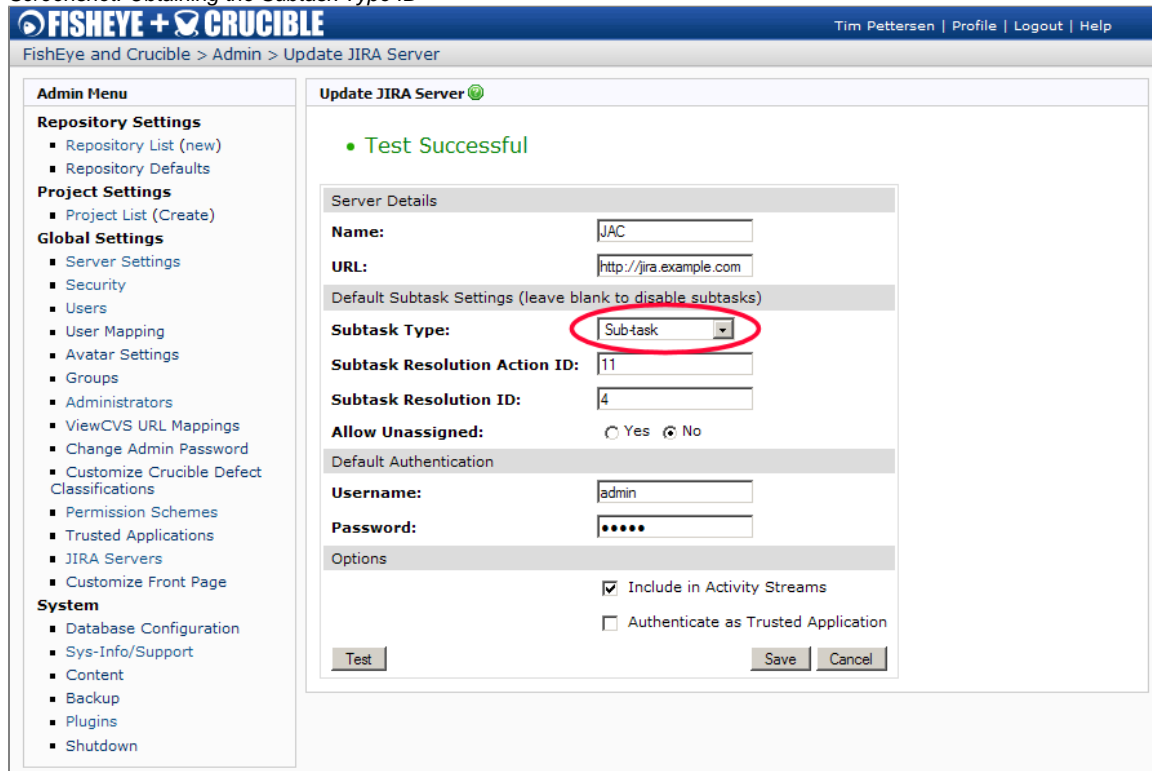
Obtaining the Subtask Type ID

 This value is required (along with the Subtask Resolution ID and Subtask Resolution Action ID) to enable creating issues from a Crucible comment. This is the subtask type that will be created when you create a JIRA subtask in Crucible.

To obtain this value, carry out the following steps.

1. Enable sub-tasks on your JIRA instance from the '**JIRA Administration**' > '**Sub-Tasks**' page. See the [JIRA documentation](#) for details on this step.
2. Return to the FishEye Administration screen and then click '**JIRA Servers**' under the '**Global Settings**' sub-menu on the left navigation bar. Click '**Edit**' next to the JIRA server you have configured.
3. Your JIRA server's basic details should appear. Click '**Test**' once again. The field for Subtask Type ID will change to a drop-down menu, showing the available subtask types. Choose the correct one.
4. Save your Crucible configuration settings.

Screenshot: Obtaining the Subtask Type ID



Obtaining the Subtask Resolution ID and Subtask Resolution Action ID

 These values are required (along with the Subtask Type ID) to enable creating issues from a Crucible comment.

To obtain these values, carry out the following steps.

1. Open your JIRA instance and go to '**Administration**' > '**Workflows**'. The '**Workflows**' screen opens. By default, the '**JIRA**' workflow is shown on screen in a table.
2. Click the '**Steps**' link in the far right table cell. The '**View Workflow Steps — JIRA**' page opens.
3. The '**Subtask Resolution Action ID**' is in the '**Open**' row, under the '**Transitions**' column. Look at the link in that cell named '**Resolve Issue**'. The ID number is shown in brackets next to that heading '**Resolve Issue**' (shown in the screenshot below as 5).
4. Save your Crucible configuration settings.
5. The '**Subtask Resolution ID**' is the '**Resolved ID**' on this page. The ID number is shown in brackets next to the heading '**Resolved**' (shown in the screenshot below as '4'). Note it down and enter it into the Crucible configuration screen.
6. Save your Crucible configuration settings.

Screenshot: Obtaining the Subtask Resolution ID & Subtask Resolution Action ID

View Workflow Steps — jira

This shows all of the steps for **jira**.

Not editable because workflow is **Active**.

☐ View all [workflows](#).
☐ View all [statuses](#).

Step Name (id)	Linked Status	Transitions (id)	Operations
Open (1)	Open	Start Progress (4) >> In Progress Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	View Properties
In Progress (3)	In Progress	Stop Progress (301) >> Open Resolve Issue (5) >> Resolved Close Issue (2) >> Closed	View Properties
Resolved (4)	Resolved	Close Issue (701) >> Closed Reopen Issue (3) >> Reopened	View Properties
Reopened (5)	Reopened	Resolve Issue (5) >> Resolved Close Issue (2) >> Closed Start Progress (4) >> In Progress	View Properties
Closed (6)	Closed	Reopen Issue (3) >> Reopened	View Properties

Editing Default JIRA Server Mappings

This setting enables the Crucible feature that shows JIRA information in a dynamic window when you hover the mouse over a JIRA issue key in Crucible. It will also turn every issue key into a hyperlink to that issue in Crucible.

To enable this feature, click **'Edit Default JIRA Server Mappings'** from the View JIRA Servers page. The **'Map JIRA Project Default'** page opens.

Screenshot: The Default JIRA Server Mappings Page

Map JIRA Project Default

Default Mappings for JIRA Projects

Default mappings for JIRA servers

Choose Fisheye Repository: CLOV

Selected Repositories:

- CLOV

Choose Crucible Project: TEST


Selected Crucible Projects:

- POTATO
- TEST

On this page, select the FishEye repositories or Crucible Projects that you wish to associate with all the JIRA servers you have configured for use

in Crucible. You can click **'add all'** to quickly include them all in this category. You can remove individual items by clicking the small 'X' marks.


Once you've finished, click **'Save'**.

 You should disable any existing Crucible [linkers](#) you have set up for JIRA, as they will override this feature and prevent the dynamic dialog box from appearing when you mouse over an issue.

Operations on Existing Servers

Once you have configured an existing JIRA server, there are three main operations you can carry out on it: **'Edit'**, **'Mappings'** and **'Delete'**. These options appear on the far right of the screen.

Screenshot: Operations in the JIRA Servers Page

View JIRA Servers 								
Name	Url	Subtask ID	Subtask Resolution Action ID	Subtask Resolution ID	Allow Unassigned	Activity Stream Provider	Trusted Application	
jac	https://jira.atlassian.com	6	2	6	true	true	false	Edit Mappings Delete
Add a JIRA Server Edit default JIRA Server mappings								

Edit settings for an existing JIRA server

When you click **'Edit'**, you can adjust any of the general settings you configured when you first added the server.

Edit mappings for an existing JIRA server

When you click **'Mappings'**, a page is loaded that is almost identical to the **'Default Mapping'** screen, but allows you to choose mappings only for that specific JIRA server.

Delete an existing JIRA server

Clicking **'Delete'** will remove the server from the list.

Crucible 1.6 Release Notes

23 September 2008

Atlassian presents Crucible 1.6

Crucible release 1.6 makes it easier to review content that is not in [FishEye](#). Furthermore, Crucible 1.6 can be deployed without FishEye for the first time. Through Crucible's new 'Light SCM' plugins, you can include content in reviews that are not associated with FishEye or even a source control repository. For example, you can review pages directly from [Confluence](#), files on any file system connected to the machine FishEye is running on, and Subversion repositories not connected to FishEye. Crucible now has better support for uploading files for pre-commit review, in addition to the existing support for patches.

Highlights of this release:

- Support for Non-FishEye Repositories
- Confluence Page Reviews
- Shared File System Repositories
- Enhanced Pre-commit Reviews & Image Support
- Multiple Admin Users
- Expanded API
- Plus numerous improvements and bug fixes



Upgrading to Crucible 1.6

You can now download Crucible from [here](#). If upgrading from a previous version, please follow the [Upgrade Guide](#).

Highlights of Crucible 1.6

1

Support for Non-FishEye Repositories

Crucible can now be deployed as a stand-alone application for the first time. With Crucible 1.6, you no longer need a FishEye license or even a source-control repository. Crucible's new *Light SCM plugin* infrastructure already supports [Confluence](#), [server file systems](#), and Subversion repository types. We will be adding GIT and ClearCase in the near future. The Light SCM interface is public and the shipped plugins are open source. As a result, you can extend these plugins or even write your own — great news for plugin developers.

2

Confluence Page Reviews

Crucible 1.6 allows you to select Atlassian Confluence as a source of material for reviews. In this way, you can use Crucible to easily review the Wiki Markup of pages created in Confluence. [Read more](#).

Screenshot: Confluence Page Reviews in Crucible

```
9 {color:#5b5bff}{*}To add a repository,{*color}
Edwin Dawson [ #permalink | (12 September 2008, 00:05) ] ↑ ↓ Reply Edit Delete
Is this the right terminology in this context?
10 {panel: borderStyle=dashed| borderColor=#ccc| titleBGColor=#F7D6C1}
11 # From the '*Admin Menu*', click '*Repository List*'.
12 # Select a '*Repository type*' from the dropdown list.
Edwin Dawson [ #permalink | (12 September 2008, 00:05) ] ↑ ↓ Reply Edit Delete
Should this list be manually numbered?
13 # Specific fields will appear on the '*Add Repository*' screen.
```

3

Shared File System Repositories

You can create a 'repository' for a local or remote directory on the server file system. Teams that are managing documents through a shared file system instead of a source control system can still benefit from peer reviews. [Read more](#).

Screenshot: Crucible Reviews on the File System

5

Multiple Admin Users

Crucible now allows the Administrator to grant other users administration privileges. Admin Users can be individually assigned or given privileges through local or remote directory groups. [Read more](#).





























6

Expanded API

The Crucible API now allows programmable review creation, along with a host of other additions. [Read more](#).

7

Plus numerous improvements and bug fixes

JIRA Issues (109 issues)				
Key	Summary	Priority	Status	
CRUC-661	On the Crucible/Fisheye dashboard Projects are displayed even if the user has no access to enter those projects (e.g. anonymous users).			Closed
CRUC-646	Disabling and reenabling a plugin gives 'javax.el.ELException: org.springframework.osgi.service.importer.ServiceProxyDestroyedException: service proxy has been destroyed'			Closed
CRUC-608	Show a list of LightSCM repositories in admin			Closed
CRUC-539	Serious: "Next comment" does not always work right			Closed
CRUC-227	Cannot delete projects			Closed
CRUC-668	We don't update lite SCM revision details if the project does not store revisions			Closed
CRUC-664	remove unnecessary transactions from read-only REST requests			Closed
CRUC-663	blank page when a non creator/author/moderator tries to preview a draft review			Closed
CRUC-659	Email notifications should be more easily threaded by mail clients			Closed
CRUC-654	improve webwork/jsp error handling			Closed
CRUC-647	improve ReviewManager.countStatesOn query			Closed
CRUC-645	Optimise DB queries			Closed
CRUC-640	Shutdown server documentation			Closed
CRUC-639	Admin password documentation			Closed

CRUC-635	Can't save changes to statement of objectives when at .../{id}/confirmApprove		 Closed
CRUC-634	ReviewData should not include repoName		 Closed
CRUC-632	Help text not updated after saving a review...		 Closed
CRUC-626	Debug Logging Documentation		 Closed
CRUC-623	SvnChangeLogBrowser.listChanges() in LiteSVN fails when processing revisions that are above the repo's configured base path		 Closed
CRUC-622	Improve performance of querying for review details		 Closed
CRUC-615	Make sure everything has a help link		 Closed
CRUC-610	Removing revisions from review that have draft comments fails without an explanation		 Closed
CRUC-602	Improve the efficiency of checking whether a review has been completed by all reviewers		 Closed
CRUC-600	REST API should return error when non-existent filter used		 Closed
CRUC-598	Emails always include a "Summary" section		 Closed
CRUC-597	Extend the Remote API to allows reviews to be deleted		 Closed
CRUC-594	Review description preview on dashboard swallows new lines from the review description		 Closed
CRUC-591	renaming a confluence repository adds a new one instead of editing		 Closed
CRUC-589	Change revisionData getAdded/Removed lines an Integer		 Closed
CRUC-588	Search Subtab in Review->Manage Files throws PropertyNotFoundException for all searches		 Closed
CRUC-586	Remove Delete option from project drop down		 Closed
CRUC-579	Changing repository when certain review tabs are selected causes NPE/bad redirect		 Closed
CRUC-566	lightscm package should not be "fisheye"		 Closed
CRUC-565	need get bulk version of getRevisionData() in SCMRepository		 Closed
CRUC-563	Toggle side-by-side and show per frx diff options on review page		 Closed
CRUC-562	REST API: createReview does not set Moderator and author correctly		 Closed
CRUC-561	anchors to some comment types don't work		 Closed
CRUC-560	REST-API: provide comit type information - esp. deleted file status		 Closed
CRUC-556	Next/previous comment arrows do not jump between file and general comments		 Closed

CRUC-554	Upgrade to non-beta atlassian-plugins		 Closed
CRUC-551	Confluence Light SCM plugin dependencies belong in plugin		 Closed
CRUC-549	Adjust IFRAME size when the admin page is resized		 Closed
CRUC-547	Make Repository source check if the engine is available.		 Closed
CRUC-545	NPE Closing Review		 Closed
CRUC-542	Use content manager to retrieve detailed file revisions		 Closed
CRUC-526	Space character in installation path to Crucible disables remote API		 Closed
CRUC-519	Added files display "no change"		 Closed
CRUC-518	Crucible does not seem to be incrementally updating the FishEye cache		 Closed
CRUC-516	REST API: Return detailed information on review in one call - items, comments, reviewers, available actions		 Closed
CRUC-511	Added files when stored show "No Change"		 Closed
CRUC-507	Confluence: implement change set paging		 Closed
CRUC-505	Add "Search for Review" functionality to Crucible remote API		 Closed
CRUC-504	Confluence: get revision comments		 Closed
CRUC-499	Create separate API jar		 Closed
CRUC-496	Document store revisions with review		 Closed
CRUC-494	Don't make source type visible in URLs or elsewhere		 Closed
CRUC-488	clicking summarise without clicking close makes the review 'disappear' from the workflow		 Closed
CRUC-484	you can link a review to itself		 Closed
CRUC-478	Ability to specify that Branches are not supported		 Closed
CRUC-472	Changing tab while editing review deletes details		 Closed
CRUC-461	Line comments are sometimes rendered twice		 Closed
CRUC-456	crucible doesn't create default permission scheme when creating a blank db		 Closed
CRUC-455	creating a review from a changeset with a non-existent repo keeps redirecting to the login screen		 Closed
CRUC-453	Allow adding a screenshot as an attachment to a review		 Closed

CRUC-439	Invite to review		 Closed
CRUC-438	Not sending email notifications after Crucible and Fisheye upgrade		 Closed
CRUC-436	create 'remove all revision' link for all tabs under 'manage files'		 Closed
CRUC-433	REST API: Get review list based on predefined and custom filters		 Closed
CRUC-422	Create Schema Upgrade to remove foreign key constraint as reqd. by delete project work		 Closed
CRUC-407	Minimal OSGi Infrastructure		 Closed
CRUC-404	Turn FE Off When only Crucible licence is present		 Closed
CRUC-402	Provide Plugin Authors with somewhere to store properties		 Closed
CRUC-400	Add Configuration UI to SVN Plugin, and Polish		 Closed
CRUC-397	Stored FRX Create UI		 Closed
CRUC-396	Stored FRX Data Impl		 Closed
CRUC-394	Add Light SCM Revisions to Reviews		 Closed
CRUC-392	Filesystem Light SCM Plugin		 Closed
CRUC-391	Default Repository for a Project can be a Light SCM repo		 Closed
CRUC-389	Light SCM Plugin instances appear in source/repository dropdowns		 Closed
CRUC-388	Subversion LSCM plugin		 Closed
CRUC-387	Confluence LSCM Plugin		 Closed
CRUC-386	Modify File Browser to Use Light SCM Plugins		 Closed
CRUC-385	Modify Changeset Browser to use Light SCM Plugins		 Closed
CRUC-384	Create Light SCM Module Type		 Closed
CRUC-381	Admin Pages for Plugins		 Closed
CRUC-373	Add General Comment via REST API		 Closed
CRUC-362	Return replies to comments via API		 Closed
CRUC-361	Project RSS feed		 Closed
CRUC-349	REST method /rest-service/reviews-v1/CR-1/comments returns HTTP 500 error		 Closed

CRUC-303	Add REST method to retrieve all reviews which involve a given file			Closed
CRUC-302	Add REST methods to allow review creation			Closed
CRUC-297	Upload files for review (not patches)			Closed
CRUC-292	javax.xml.bind.JAXBException generated when invoking the getGeneralComments web service method			Closed
CRUC-244	Add revisions not diffs to a review			Closed
CRUC-186	Allow an abandoned review to be deleted			Closed
CRUC-150	Allow the email address in the from section of the notification emails to be the user's actual email address			Closed
CRUC-18	Load All Users from Crowd/LDAP/etc			Closed
CRUC-662	Updating general comments in the REST api has a permission flaw			Closed
CRUC-574	Allowed Review Participants left blank means what?			Closed
CRUC-548	REST API: Handle allowReviewersToJoin flag on review			Closed
CRUC-540	make metrics-config.xsd available online			Closed
CRUC-487	Page title says "dead" for a review that has been abandoned			Closed
CRUC-479	A comment which was a defect, but is no longer, still shows the defect attributes			Closed
CRUC-457	show authornamex next to revision in revision dropdowns			Closed
CRUC-401	Real Admin Users			Closed
CRUC-399	Optionally Store Files for all FileRevisions			Closed
CRUC-592	Filter names used in Cru do not match the menus			Closed
CRUC-469	Sort list of repositories alphabetically on "Add project" page			Closed
CRUC-419	Typo in email "reviewers are now complete"			Closed

Crucible 1.6 Changelog

On this page:

- From 1.6.5.a to 1.6.6
- From 1.6.4 to 1.6.5.a
- From 1.6.3 to 1.6.4
- From 1.6.2.1 to 1.6.3
- From 1.6.2 to 1.6.2.1
- From 1.6.1 to 1.6.2
- From 1.6.0 to 1.6.1

From 1.6.5.a to 1.6.6

10 February 2009





This release updates the supporting libraries for Crucible plugins. This change enables the use of the new [Git Crucible plugin](#) for performing code reviews against a Git repository.

The Git plugin is not currently bundled with Crucible but may be downloaded from the Atlassian Maven repository here: <https://maven.atlassian.com/browse/com.atlassian.crucible.plugins/crucible-git-scm-plugin/1.0>

The Git plugin should be considered an early access release. It allows reviews to be performed against a local Git repository clone. Note that the plugin does not update the cloned repository automatically. For more information on the Git plugin, please see the [documentation](#).

We are very interested in any feedback users have on the Git Crucible plugin. Please post feedback in the [Crucible forums](#).

Full list of issues fixed in this release:













JIRA Issues (2 issues)			
Key	Summary	Priority	Status
CRUC-1406	test		 Closed
CRUC-900	GIT SCM Module		 Closed

From 1.6.4 to 1.6.5.a

22 December 2008

This release contains a number of improvements and bug fixes.

Full list of issues fixed in this release:











JIRA Issues (6 issues)			
Key	Summary	Priority	Status
CRUC-947	Patch context displays Index Out of bounds exception		 Closed
CRUC-938	doco that we require jdk 1.5.x and 1.6.0u4+		 Closed
CRUC-928	Crucible 1.6.4 REST API fails under JavaSE 6		 Closed
CRUC-867	file names with "--" in their name will not be rendered properly in crucible reviews		 Closed
CRUC-886	Context lines can be duplicated with Patch diff options		 Closed
CRUC-683	Add a "expand all unchecked" option to top of review screen		 Closed













































From 1.6.3 to 1.6.4

20 November 2008

This release contains bug fixes and minor improvements, and includes the new plugin points developed for [AtlasCamp 2008](#).

Full list of issues fixed in this release:

JIRA Issues (27 issues)			
Key	Summary	Priority	Status
CRUC-753	Deadlock in HSQL?		 Closed
CRUC-926	GET /reviews-v1/<review id>/reviewitems/X fail to return reviewitems of certain types		 Closed
CRUC-862	Crucible 1.6.4 documentation updates		 Closed
CRUC-846	Publish new CAC pages on 1.6.4 release		 Closed
CRUC-844	Cant update versions of new files with comments		 Closed

CRUC-837	P4 Lite SCM plugin does not repeat paths when listing changesets		 Closed
CRUC-835	Reviews created in 1.6.2 with revisions of unknown content type (text) show as binary in 1.6.3 and never updated		 Closed
CRUC-834	Different background color for user comments		 Closed
CRUC-832	Can't add new revision of a file already under review if commented on - even if comment is deleted		 Closed
CRUC-827	upgrade to hsql 1.8.0.10		 Closed
CRUC-826	error when sending review to moderator		 Closed
CRUC-821	change all references of 'summary' to 'all comments'		 Closed
CRUC-809	File upload of a single text file gives java.lang.Exception: Not enough revisions to diff		 Closed
CRUC-793	File in Review shows up as "Updating" and does not show source due to a Runtime Exception thrown by the SVNlite plugin		 Closed
CRUC-788	add REVISION_LINK and CHANGESSET attributes to light scm plugins		 Closed
CRUC-787	No authentication required to retrieve user list in API		 Closed
CRUC-778	rework for CR-FE-696 FE-703 - making quicksearch respect quoted queries		 Closed
CRUC-765	When in Crucible, a AJAX timeout destroys the Crucible page		 Closed
CRUC-748	Add inline diff option bar to patch reviews		 Closed
CRUC-727	Configuring a plugin causes IE to hit 100% CPU and crashes the browser		 Closed
CRUC-582	Provide detailed information for cvs and light SCM repos via REST		 Closed
CRUC-398	Stored FRX REST API		 Closed
CRUC-820	Create JSR-170 (JackRabbit) SCM Plugin as Code Example		 Closed
CRUC-802	File Upload from the Manage Files area drops the path info on submit		 Closed
CRUC-328	Reviewer Auto Suggest		 Closed
CRUC-255	Users lists unsorted when Allowed Participants restricted by Group		 Closed
CRUC-568	Auto Suggest Reviewer		 Closed

From 1.6.2.1 to 1.6.3

5 November 2008

This release rolls together several improvements and bug fixes.

- Auto-save draft comments.
- Performance improvements when using Light SCM repositories.
- Bundle a Performer Light SCM implementation.
- Various REST API improvements, including Conditional-Get support, improved error handling and revised review searching, which now allows any criteria to be omitted.
- JSON serialization has been added to the REST API, allowing the use of JSON in REST API calls. This feature is in an experimental state at present. Please report any issues discovered.






























Please be aware of the upgrade notes regarding Light SCM repositories (this does not impact FishEye repositories):

- The configuration storage for the bundled File-system, Confluence and Subversion Light SCM plugins changed. Once you have upgraded to 1.6.3 you will need to re-add those repositories. Please read the [Crucible 1.6.3 Upgrade Guide](#).
- The Light SCM plugin API was changed in this release. Light SCM plugins compiled against the old API will not work in this release of Crucible.

Full list of issues fixed in this release:

JIRA Issues (78 issues)			
Key	Summary	Priority	Status
CRUC-729	Can't review binary file becoming textual		Closed
CRUC-725	Light SCM allows creation of repos with the same name as Fisheye repos		Closed
CRUC-701	Support Performer repositories in RepositoryService		Closed
CRUC-486	Invalid rendering of Search Comments report (table part)		Closed
CRUC-785	any screenshots of scroll to changeset for doco		Closed
CRUC-784	filesystem lightscm plugin should not use "current" as the revision name		Closed
CRUC-777	rework for CR-FE-697 CRUC-728		Closed
CRUC-776	rework for CR-FE-702 CRUC-624		Closed
CRUC-775	Create unit tests that verify JSON serialization		Closed
CRUC-773	Add JSON support to Release Notes		Closed
CRUC-772	Add documentation on how to use JSON to Confluence		Closed
CRUC-771	Improve the I&F of manage files		Closed
CRUC-769	Add JSON support to REST		Closed
CRUC-768	Upgrade to Jersey 1.0, and include jersey-spring for possible future springification		Closed
CRUC-767	Autosave race condition		Closed
CRUC-764	Add this change to the release notes		Closed
CRUC-759	Improve Confluence Light SCM Performance		Closed
CRUC-758	make scroll to changeset look better		Closed
CRUC-757	Custom filter object in Crucible REST should not use primitive values		Closed
CRUC-752	[crucible] in the closed review email subject		Closed
CRUC-750	Plugin config change will affect user configs		Closed
CRUC-749	Get Selenium Tests working		Closed
CRUC-744	summary email includes deleted comments		Closed
CRUC-739	Getting error popup on review due to confusion about a directory.		Closed
CRUC-736	Automatically save draft comments (autosave)		Closed
CRUC-733	Crucible without FishEye still says FishEye in titles		Closed
CRUC-724	Implement Light SCM plugin for Performer		Closed

CRUC-723	Converting Crucible reviews from FishEye repo to LightSCM SVN fails to load revisions		 Closed
CRUC-722	Remove the use of deprecated CrucibleRevision.getSource()		 Closed
CRUC-712	Create unit tests for REST		 Closed
CRUC-711	document Alt+Click for selecting review text		 Closed
CRUC-708	Change REST filter retrieval from POST to GET		 Closed
CRUC-706	Retrieving non-existing metrics version in REST gives 500 "Internal Server Error", should be 404		 Closed
CRUC-702	summary email documentation		 Closed
CRUC-700	add maybe-details and maybe-filehistory to FileSummary		 Closed
CRUC-699	Revision Details should be a map		 Closed
CRUC-698	maybe-details provided by SCMs are not used		 Closed
CRUC-697	Change ManageFiles tab so that it does not require so much information from the SCM		 Closed
CRUC-696	Address performance problems in LightSCM plugin API		 Closed
CRUC-695	Crucible REST should throw an exception when adding changesets to reviews that already have comments		 Closed
CRUC-694	Adding changesets on open reviews messes up the in-line comments		 Closed
CRUC-693	Still cannot delete projects		 Closed
CRUC-691	Implement Conditional Get in REST API		 Closed
CRUC-688	FR_EXTRA.FRX_ORDER needs a unique constraint		 Closed
CRUC-680	Changeset dates are wrong		 Closed
CRUC-679	Add xwork action that returns the text summary (for copying and pasting).		 Closed
CRUC-678	Add send summary button and form		 Closed
CRUC-677	Update summary template to include comments		 Closed
CRUC-671	Create review from changeset gives HTTP 500		 Closed
CRUC-670	Improve REST error reporting (HTTP return codes)		 Closed
CRUC-658	Scroll To: box for changelogs		 Closed
CRUC-653	[admin] adding a "default reviewer" incorrectly adds an "allowed reviewer" in some cases		 Closed
CRUC-627	make stored reviews viewable when the source isn't available		 Closed
CRUC-624	XML Parsing Error from Crucible Review Service using allReviews filter		 Closed
CRUC-509	Add info about IDE integration in Web UI		 Closed
CRUC-470	Summary Email Improvements		 Closed
CRUC-395	Stored FRX		 Closed
CRUC-383	Maven Archetype for plugin developers		 Closed
CRUC-380	Light SCM		 Closed
CRUC-327	Copy and paste is not working		 Closed
CRUC-763	DELETE operations in Crucible REST should return status 204 "No Content" to be more RESTful		 Closed
CRUC-747	Auto save draft comment needs to delete the draft on cancel		 Closed
CRUC-742	Images in patches don't work		 Closed

CRUC-735	Error message in Crucible file management is misplaced		 Closed
CRUC-728	add CSID and SOURCE_URL to lightSCM details		 Closed
CRUC-713	Change REST return code from 200 to 201 "Created" for several POST actions		 Closed
CRUC-710	You do not have permission to see all the search results, seen in (my) to summarise and out for review		 Closed
CRUC-709	Refactor exception handling issues		 Closed
CRUC-705	Create RestXxxServices via Spring		 Closed
CRUC-689	Revision details missing from choose diff dropdown		 Closed
CRUC-687	<i>no comment</i> commit message tooltip		 Closed
CRUC-667	re address update of revision details		 Closed
CRUC-665	Correct documentation for REST API for getting file information		 Closed
CRUC-656	Previously deleted files show as having an old version in a review when they are added again		 Closed
CRUC-655	Downloaded files do not have the correct file name		 Closed
CRUC-529	Have a way to select text in source windows		 Closed
CRUC-475	Create documentation on creating reviews using remote API		 Closed
CRUC-536	Improper code colorization for C++		 Closed

From 1.6.2 to 1.6.2.1

24 October 2008

This release fixes a problem in 1.6.2 when running Crucible on Windows. Due to a file-lock issue, the upgrade script in 1.6.2 could not start.

- [CRUC-781](#) Upgrade from 1.6 to 1.6.2 fails for windows machines.

From 1.6.1 to 1.6.2

21 October 2008

This release fixes a bug in the way Crucible stores review data. This bug was introduced in Crucible 1.6.0. **We strongly recommend all 1.6.0, 1.6.0-beta and 1.6.1 users immediately upgrade to this release.**

If this bug occurs in your Crucible instance, you will find that review data created after that point will be corrupt. If you find that is the case please contact [Crucible support](#) for assistance.

- [CRUC-743](#) Switch from CACHED tables mode back to memory table.

From 1.6.0 to 1.6.1

24 September 2008

This is a bug fix release.

- Crowd 1.3 users will need to upgrade to Crowd 1.4.4 or later due to an incompatibility with this version of Crucible.
- [CRUC-673](#) NPE when viewing a review.
- [CRUC-674](#) NPE when closing a review.

Crucible 1.6.3 Upgrade Guide

Upgrade Notes

- The configuration storage for the bundled File-system, Confluence and Subversion Light SCM plugins changed. Please follow the procedure below.
- The Light SCM plugin API was changed in this release. Light SCM plugins compiled against the old API will not work in this release of

Crucible.

Upgrade Procedure

Due to a configuration change, you will need to delete and re-add your LightSCM repositories.

1. Before you shut down Crucible, take a note of your Light SCM configuration. You can view this configuration in the **Repository List** Admin page, in the **LightSVN**, **Confluence** and **File System** sections.
2. Follow the general instructions on [upgrading Crucible](#).
3. While Crucible is shut down, delete the `confluence`, `svn` and `filesystem` config files in `FISHEYE_INST/var/plugins/user`.
4. Once Crucible has been restarted, re-add the Light SCM repositories from step 1.

Crucible 1.5 Release Notes

14 April 2008

Atlassian presents Crucible 1.5

Crucible release 1.5 brings new enhancements that make your code review activities quicker and easier. The all-new per-project page consolidates the display of work done on a particular goal or product, while filtered search for defects and comments provides rapid access to Crucible content that you need to see, now.

Highlights of this release:

- Project Dashboard
- Filtered comments & defects search, with statistical summary
- Customisable email templates
- Improvements to Crucible Plugin API beta
- Plus numerous improvements and bug-fixes



Upgrading to Crucible 1.5

You can now download Crucible from [here](#). If upgrading from a previous version, please follow the [Upgrade Guide](#).

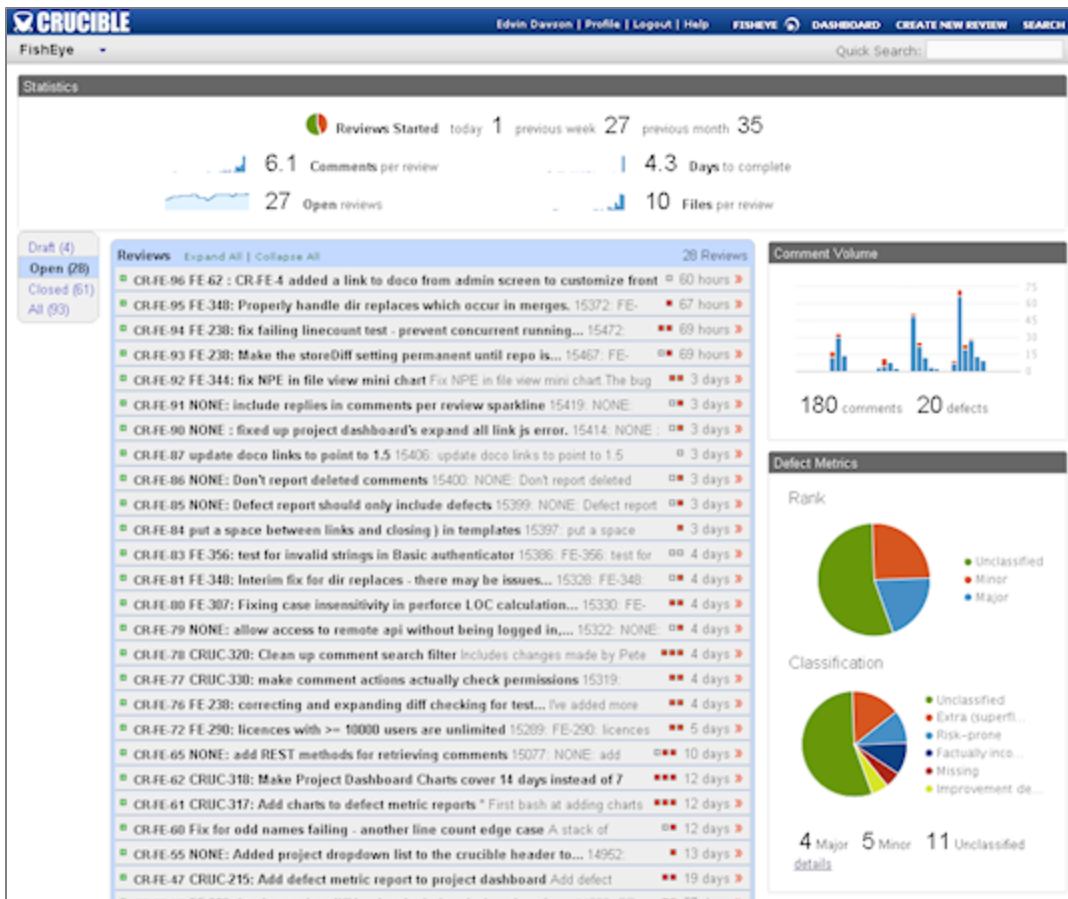
Highlights of Crucible 1.5



Project Dashboard

Crucible 1.5 introduces the Project Dashboard, which allows you to see open reviews that belong to a given project, presented with additional project-related data and graphs.

Screenshot: Crucible Project Dashboard

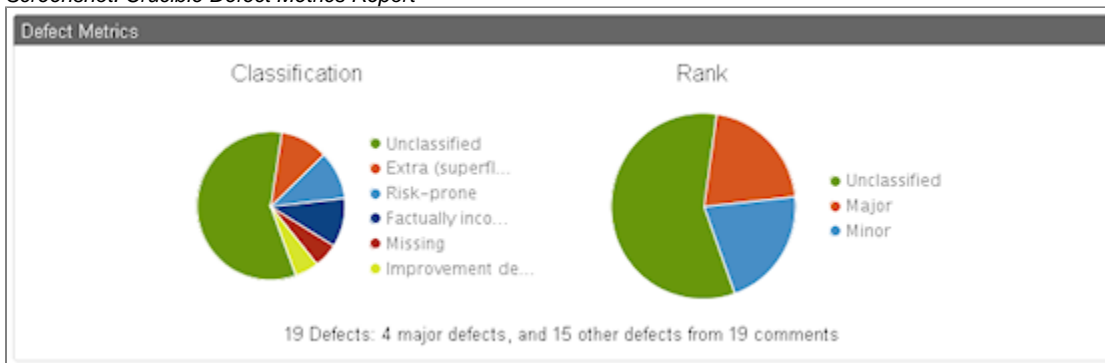


2

Filtered comments & defects search, with statistical summary

Defects and comments are now searchable, easing the difficulty of finding a particular piece of work or revision (and its relevant comments). These search results now also show a very useful statistical summary. Also, a new defect metrics report is available.

Screenshot: *Crucible Defect Metrics Report*



3

Customisable email templates

You can now customise the content and appearance of email notifications that get sent to Crucible users. For example you can append a legal




















disclaimer, alter the subject line or provide custom header text for all messages.



Improvements to Crucible Plugin API beta

Now with REST support and the ability to upload patches, the Crucible Plugin API beta is for Crucible integrators who want to extend Crucible to interoperate with their enterprise infrastructure or processes.

Plus numerous improvements and bug-fixes

JIRA Issues (38 issues)			
Key	Summary	Priority	Status
CRUC-344	Defect pie chart on comment search page includes non-defect comments.	↑	 Closed
CRUC-332	create a better looking indication icon	↑	 Closed
CRUC-324	Add Project Dropdown to Search Pages	↑	 Closed
CRUC-323	Create 'Reviews Started' Pie Chart for Project Dashboard	↑	 Closed
CRUC-318	Make Project Dashboard Charts cover 14 days instead of 7	↑	 Closed
CRUC-314	sidebyside diff line numbers must be aligned to the top	↑	 Closed
CRUC-307	Optimize chart queries	↑	 Closed
CRUC-299	link to project page	↑	 Closed
CRUC-289	permissions are not checked in rss feeds (but are in list html display)	↑	 Closed
CRUC-285	"project project" in Permission Screen in admin	↑	 Closed
CRUC-283	Escape HTML in Javascript properly	↑	 Closed
CRUC-280	remove debug log from Process Notification	↑	 Closed
CRUC-247	Flag files which have been modified since a review was created	↑	 Closed
CRUC-246	Parentheses should be valid URL characters	↑	 Closed
CRUC-242	Add RPC call to create review from patch	↑	 Closed
CRUC-240	Checking box in "manage files" -> "changesets" spins forever	↑	 Closed
CRUC-232	Address all issues outline here	↑	 Closed
CRUC-220	permalink for a comment does not work when summarize mode is active	↑	 Closed
CRUC-212	Create Project Dashboard page	↑	 Closed

CRUC-206	URL recognition in Crucible comment does not always match whole string			Closed
CRUC-201	Allow project to be specified on create review URL			Closed
CRUC-194	Documentation: restoring Crucible backup data			Closed
CRUC-184	When Creating Review From FishEye, Default to Right Project			Closed
CRUC-172	Create 'Getting started with Crucible' document as part of User Guide revisions			Closed
CRUC-147	Add Permissions Checks to All Operations on Reviews			Closed
CRUC-128	email notification of review closure prints literal "null" for absent field value			Closed
CRUC-85	Sometimes we need to see differences in UNICODE files.			Closed
CRUC-21	change icon for "View History In Fisheye"			Closed
CRUC-290	Warnings during backup: File does not exist /data/crucible/inst/var/data/crddb/crucible.lck & File does not exist /data/crucible/inst/var/data/crddb/crucible.log			Closed
CRUC-282	Crucible fails to upload patch file with file information that contain blank spaces between file path and timestamp			Closed
CRUC-266	let the subject of the email be modifiable via the notification template			Closed
CRUC-260	How to customize the email notification content			Closed
CRUC-248	Configurable email subject line			Closed
CRUC-241	Incorrect dates in RSS feeds			Closed
CRUC-230	Crucible allows circular review linking and can't delete a review link			Closed
CRUC-226	Title wrong for post-approval in manage files tab			Closed
CRUC-162	From field on emails is incorrect or a least deceiving			Closed
CRUC-238	Perforce unit tests do not work on Windows platform			Closed

Crucible 1.5 Changelog

On this page:

- [From 1.5.3 to 1.5.4](#)
- [From 1.5.2 to 1.5.3](#)
- [From 1.5.1 to 1.5.2](#)
- [From 1.5.0 to 1.5.1](#)

From 1.5.3 to 1.5.4

1 August 2008

This release contains minor improvements and bug fixes.

JIRA Issues (10 issues)			
Key	Summary	Priority	Status
CRUC-638	How to (easily) add specific changeset to a review		Closed
CRUC-513	If we display no change on a revision, comments made are hidden by default.		Closed
CRUC-503	Upgrade to crowd-integration-client 1.4.4		Closed
CRUC-498	Draft comment replies are deleted when completing review		Closed
CRUC-454	two notification emails sent		Closed
CRUC-429	Prevent two projects with same key different case		Closed
CRUC-284	AnnotatorTag error: Could not get blame for file		Closed
CRUC-633	How to remove a reviewer from a review		Closed
CRUC-538	Comments weirdly duplicated on occasion		Closed
CRUC-417	large var/data/crudb/crucible.log		Closed

From 1.5.2 to 1.5.3

23 June 2008

This release contains bug fixes.











JIRA Issues (1 issues)			
Key	Summary	Priority	Status
CRUC-462	"File closed: var\data\data0.bin" after backups		Closed

From 1.5.1 to 1.5.2

27 May 2008

This release contains bug fixes.



















JIRA Issues (16 issues)			
Key	Summary	Priority	Status
CRUC-262	Urgent : SVNRepository methods are not reenterable		Closed
CRUC-405	[Performance] AnnotatorTag can hang in sort loop for (perhaps) large diffs		Closed
CRUC-370	create action to remove all revisions from a review		Closed
CRUC-363	Remove all Revisions in manager files-> Changesets tab doesn't work		Closed
CRUC-355	Review Stats Wrong		Closed
CRUC-354	Personal dashboard stats don't match		Closed
CRUC-336	can't search for review keys in quicksearch		Closed
CRUC-185	Abandoned Reviews Still Show in FishEye		Closed
CRUC-434	'Expand Commented' appears when there are general comments but no revision comments		Closed
CRUC-413	Crucible always creates a new default project after it is deleted		Closed
CRUC-406	Remove 'All Projects' from projects dropdown		Closed

CRUC-376	Prevent deletion of default project		 Closed
CRUC-371	Comment volume chart reports confusion		 Closed
CRUC-358	Height not respected if less than 144 in defect chart		 Closed
CRUC-335	[CSS] project list a little wonky in IE		 Closed
CRUC-233	Changing or expanding the term - 'Approve'		 Closed

From 1.5.0 to 1.5.1

24 April 2008

This release contains bug fixes.

JIRA Issues (9 issues)			
Key	Summary	Priority	Status
CRUC-364	Starting Crucible 1.5 Spits Lots of Errors Out		 Closed
CRUC-367	LOC exceptions filling Debug log to the tune of 3-6GB per day		 Closed
CRUC-353	Crucible seems not to be able to handle uploaded git patches		 Closed
CRUC-350	javax.servlet.ServletException		 Closed
CRUC-304	Add REST method to retrieve review items for a given review		 Closed
CRUC-291	NullPointerException thrown when invoking the getAllRevisionComments web service method		 Closed
CRUC-161	Create document stating when a customer purchases Crucible that they do not need to have the latest version of FishEye		 Closed
CRUC-357	Sparklines have wrong content type		 Closed
CRUC-333	In IE, "view diff to latest" partially obscured		 Closed

Crucible 1.2 Release Notes

December 5, 2007.

The Atlassian Crucible team is delighted to present Crucible 1.2.

Crucible release 1.2 brings you a host of popular new features. You can now group your reviews into projects (similar to [JIRA](#) projects) and authorise your users via project permission schemes.

New user management screens make the administrator's job a lot easier. The new built-in integration with Atlassian [Crowd](#) extends your authentication and authorisation capabilities. You can now include users and groups from one or more Crowd directories, and provide single sign-on (SSO) across Atlassian products plus any other applications that support SSO.

Crucible's integration with [JIRA](#) and [FishEye](#) is now closer than ever before. Read the details below.

Highlights of this release:

- Reviews grouped into projects
- Customisable permission schemes
- Plugin API

- Enhancements to user management
- JIRA integration
- Crucible 1.2 includes FishEye 1.4
- Plus over 20 improvements and bug-fixes

Responding to your feedback:

★ 8 new feature requests/improvements implemented

★ 9 votes satisfied

Your [votes and issues][<http://jira.atlassian.com/browse/CRUC>] help us keep improving our products, and are much appreciated.

Upgrading to Crucible 1.2

You can now download Crucible from [here](#). If upgrading from a previous version, please follow the [Upgrade Guide](#).

Highlights of Crucible 1.2

1

Reviews grouped into projects

- Crucible now supports [projects](#) - every review will belong to a project.
- Each project has a unique key (prefix), modelled on JIRA keys.
- You can add your own projects via the new administration screens.
- You can specify review defaults per project, such as the default users for each role and a default repository.
- And you can restrict the users/groups who can perform a particular role, e.g. only team leaders can be moderators.
- Each project has its own permission scheme (see below).

Repository List	
Identification	
Name:	<input type="text"/>
Key:	<input type="text"/>
Repository	
Default Repository:	svn <input type="button" value="v"/>
Moderator	
Default Moderator:	<input type="text"/> <i>Start typing a user name then press enter to select.</i>
Default Reviewers	
<input type="checkbox"/> Let allowed review participants join a review	
Users:	<input type="text"/> <i>Start typing a user name then press enter to select.</i>
Groups:	<input type="text"/> <i>Start typing a group name then press enter to select.</i>
Allowed Review Participants	
Users:	<input type="text"/> <i>Start typing a user name then press enter to select.</i>
Groups:	<input type="text"/> <i>Start typing a group name then press enter to select.</i>
Project Permissions	
Permission Scheme:	default <input type="button" value="v"/>
<input type="button" value="Save"/>	

2

Customisable permission schemes

- A [permission scheme](#) is a set of actions which a user can perform (e.g. create a review, approve a review, etc).
- Each project can have its own custom permission scheme — or you can use the same scheme for multiple projects.
- The permission scheme for a review is determined by the review's project.

Edit Permission Scheme - new scheme		
<div> <input type="text" value="Top Secret"/> <input type="button" value="Rename"/> </div>		
Permissions	Users / Groups / Review Roles	
Modify Files Ability to change the set of revisions being reviewed.	<ul style="list-style-type: none"> • Anonymous users: false • All logged in users: false • Individual users: • Groups: • Roles: Moderator Creator 	edit
Close Ability to close a review once it has been summarized.	<ul style="list-style-type: none"> • Anonymous users: false • All logged in users: false • Individual users: • Groups: • Roles: Moderator 	edit
View Review Ability to view a review.	<ul style="list-style-type: none"> • Anonymous users: true • All logged in users: true • Individual users: • Groups: • Roles: 	edit
Uncomplete Ability to indicate they have not completed a review, after indicating they have completed a review.	<ul style="list-style-type: none"> • Anonymous users: false • All logged in users: false • Individual users: • Groups: • Roles: Reviewer 	edit
Re-Open Ability to re-open a closed review.	<ul style="list-style-type: none"> • Anonymous users: false • All logged in users: false • Individual users: • Groups: • Roles: Moderator 	edit

3

Plugin API

- A new plugin Crucible programming interface (API), in **beta** for this release, supports the following functionality:
 - Create or modify reviews and comments.
 - Add files, patches, etc to reviews.
 - Invoke state transitions.
 - Add custom servlet handlers.
- More [information](#).

4

Enhancements to user management

In Crucible 1.1.2, we introduced support for public signup (self-registration). Now in Crucible 1.2:

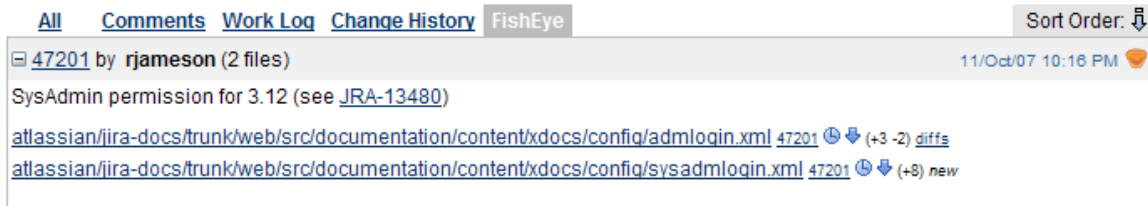
- Administrator can make the email address for [self-signups](#) optional.
- Improved user interface makes user administration easier.
- [Groups](#) are supported.
- Read the [FishEye documentation](#).

5

JIRA integration

The new version 1.2 of the [FishEye-for-JIRA](#) plugin includes some useful improvements:

- new 'FishEye' tab for JIRA issues and projects
- improved ability to create a [Crucible review](#) from the 'FishEye' tab within a JIRA issue — just click the Crucible icon: 



6













Crucible 1.2 includes FishEye 1.4

... and provides closer integration than ever before.

- FishEye screens include links to existing Crucible reviews. So you can see which files/changesets have been reviewed.
- EyeQL allows you to search for Crucible data. For example, you can search for files that have not yet been reviewed.
- Crucible now has built-in [Crowd/SSO](#) support.
- See the [FishEye 1.4 Release Notes](#).

7

Plus over 20 improvements and bug-fixes

JIRA Issues (32 issues)			
Key	Summary	Priority	Status
CRUC-181	Old screenshot in 1.2 release notes	↑	 Closed
CRUC-175	LHS abandon button on Crucible review screen broken	↑	 Closed
CRUC-166	Some files not work	↑	 Closed
CRUC-152	sysinfo admin screen should show both CRU and FE license string	↑	 Closed
CRUC-146	Add review information to ALL search result pages	↑	 Closed
CRUC-143	add review-constraints in the EyeQL where clause	↑	 Closed
CRUC-140	ensure "linked" reviews are exported thru data in remote api	↑	 Closed
CRUC-122	Move review to another project	↑	 Closed
CRUC-121	EyeQL return clause: reviews	↑	 Closed
CRUC-113	Allow creation of reviews for multiple changesets through /cru/create URL	↑	 Closed
CRUC-109	Presented with Post/delete drafts buttons when no comments drafted made	↑	 Closed
CRUC-107	Ability to change username	↑	 Closed

CRUC-104	AnnotatorTag error reviewing patch	↑	 Closed
CRUC-102	Create Project Model object and Hibernate DB upgrade	↑	 Closed
CRUC-101	Add 'Projects'	↑	 Closed
CRUC-93	Crucible should preserve request params/URLs through login redirects	↑	 Closed
CRUC-90	More Administrator Options	↑	 Closed
CRUC-89	Add 'Default Reviewers'	↑	 Closed
CRUC-88	Review Groups	↑	 Closed
CRUC-86	add "allow anyone" as a per-project default	↑	 Closed
CRUC-80	Should be able to create a review when there are no configured repositories	↑	 Closed
CRUC-78	merge cru/fe src and content trees	↑	 Closed
CRUC-73	Generate HEAD review from directory	↑	 Closed
CRUC-68	Dragging to deselect source lines no longer works	↑	 Closed
CRUC-65	beta plugin api	↑	 Closed
CRUC-56	Self-registration	↑	 Closed
CRUC-43	Webservice API for Reviews	↑	 Closed
CRUC-95	Change Diff Buttons Losing State	↓	 Closed
CRUC-94	Show Existing Reviews In Fisheye	↓	 Closed
CRUC-61	Author should be able to "complete"	↓	 Closed
CRUC-36	Should my review status go back to incomplete if I start adding comments?	↓	 Closed
CRUC-115	Accept Patch Review From Clipboard	↕	 Closed

Crucible 1.2 Changelog



On this page:



































- [From 1.2.2 to 1.2.3](#)
- [From 1.2.1 to 1.2.2](#)
- [From 1.2 to 1.2.1](#)

From 1.2.2 to 1.2.3

7 February 2008

This release contains bug fixes (including those in from [FishEye 1.4.3](#)).

JIRA Issues (26 issues)			
Key	Summary	Priority	Status
CRUC-127	Improperly stopping crucible causes data loss!		 Closed
CRUC-274	Create IDE integration with IntelliJ IDEA	↑	 Closed
CRUC-273	Create IDE integration with Eclipse	↑	 Closed
CRUC-256	Check boxes for checking off files reviewed in a review are lost if navigate to raw text by "download raw text" icon	↑	 Closed
CRUC-249	Accept review comments in Japanese (unicode characters)	↑	 Closed




CRUC-243	race condition between crucible backup and repository scan		 Closed
CRUC-235	File tabs should be kept open		 Closed
CRUC-210	Create JIRA issues from code reviews		 Closed
CRUC-208	Nullpointer exception when creating new review		 Closed
CRUC-202	update 'Repositories' screenshot		 Closed
CRUC-196	Error on attempting to create a Review.		 Closed
CRUC-180	'Show Full Source' should continue to highlight the diff		 Closed
CRUC-177	Comments on reviews are being sent but don't appear in the review when viewed online		 Closed
CRUC-165	Folder Issue		 Closed
CRUC-139	Can't add new versions of the same file to a review		 Closed
CRUC-112	Point online help links to new Crucible doc space		 Closed
CRUC-106	Add content to new documentation page 'Crucible and FishEye'		 Closed
CRUC-105	Upload PDF, XML and HTML versions of Crucible docs		 Closed
CRUC-103	Create admin page for Projects		 Closed
CRUC-100	Move Crucible docs to Confluence		 Closed
CRUC-99	Feature request: I would like to see statistics about number of major and minor issues found per developer.		 Closed
CRUC-92	Allow Anonymous Access Per Repository		 Closed
CRUC-84	Ability to create a review directly from JIRA		 Closed
CRUC-50	Allow review of entire file, not a particular diff		 Closed
CRUC-28	After Crucible forums are moved, update all links in docs		 Closed
CRUC-258	Allow all users to be able to a "Moderator" or "Author" in a review project		 Closed

From 1.2.1 to 1.2.2

This release contains some minor improvements and bug fixes.







- Trusted Application Support**
 FishEye/Crucible now allows you to set up trusted communications with other Atlassian applications. At this point, the JIRA FishEye plugin supports Trusted Applications. The JIRA FishEye plugin can request information from FishEye on behalf of the currently logged-in user, and FishEye will not ask the user to log in again or to supply a password. Previously FishEye/Crucible would have used a single 'system' account to determine permissions. Now, FishEye/Crucible can apply the correct permission settings for the logged-in user.

- Hyphens are now allowed in project key names.

JIRA Issues (3 issues)			
Key	Summary	Priority	Status
CRUC-207	allow hyphens in Crucible project keys	↑	 Closed
CRUC-204	Add project to dashboard filter	↑	 Closed
CRUC-195	groups access control to repositories failed after configured External authentication (LDAP)	↑	 Closed

From 1.2 to 1.2.1

This is a small bug-fix release.

JIRA Issues (6 issues)			
Key	Summary	Priority	Status
CRUC-199	Top of project edit form says Repositories List	↑	 Closed
CRUC-198	multiple copies of revisions added to review if you add changesets in reverse chronological order	↑	 Closed
CRUC-182	Projects Don't Persist Through Pages	↑	 Closed
CRUC-179	Support addition of multiple revisions of the same file that appears in multiple changesets	↑	 Closed
CRUC-178	Hit NPE when perform backup	↑	 Closed
CRUC-40	Getting wrong changeset number causes diff to be missing	↑	 Closed

Crucible 1.2 Upgrade Guide

Upgrade Notes

- During the upgrade, a default [project](#) and default [permission scheme](#) will be created. All existing reviews will be assigned to the default project.

Upgrade Procedure

- Please read the [Release Notes](#) for the version you are upgrading to, as well as any versions you are skipping.
- Follow the instructions on [upgrading Crucible](#).

Crucible 1.1 Release Notes



Crucible 2.0 has now been released. Read the [Release Notes](#).

Crucible 1.1 allows pre-commit (patch) reviews, side-by-side diff mode, syntax highlighting in diffs, and many other bug fixes and improvements.

Upgrading Crucible

You can now download Crucible from [here](#). Information on installing Crucible can be found [here](#). If upgrading from a previous version, please follow the [Upgrade Guide](#).

Highlights of Crucible 1.1

- Pre-commit review ([patch review](#)).
- Progress tracking through a review by marking each file as 'done'.
- Side-by-side diff mode within the review display.
- Syntax highlighting when displaying a diff.
- Many small UI fixes and improvements. Refer to the [changelog](#) for more details.

Crucible 1.1 Changelog

On this page:

- [From 1.1.3 to 1.1.4](#)
- [From 1.1.2 to 1.1.3](#)
- [From 1.1.1 to 1.1.2](#)
- [From 1.1 to 1.1.1](#)

From 1.1.3 to 1.1.4

This release updates the included FishEye component and includes a number of performance improvements and bug fixes for Subversion and Perforce repository indexing.

From 1.1.2 to 1.1.3

This release fixes a bug [CRUC-104](#) that prevented Crucible from correctly displaying large patches.

From 1.1.1 to 1.1.2

This release adds some new user-related functions and includes bug fixes.

New Features

- You can now allow users to create their own user accounts (sign-up).
- You can now allow anonymous browsing of reviews.
- Users can now add themselves as a reviewer ('Join a review'). This is an option that is configured per review.

Bug fixes

- Fix problem where Crucible would only display the top part of each diff in a patch.
- Fix various JavaScript and UI errors.
- Fix various IE6 and IE7 problems.
- Fix problem where some users were redirected to /uar/browser.css after login.

From 1.1 to 1.1.1

This is a small bug-fix release. It addresses a stack-overflow problem for some configurations.

Crucible 1.1 Upgrade Guide

Upgrade Notes

- As of version 1.0, Crucible now requires a JVM version 1.5 or later. Previously, 1.4+ was required.
- Crucible 1.1.4 includes FishEye 1.3.8.
- Upgrading from 1.0.4 (or earlier) will force a complete re-index of P4 repositories.

Upgrade Procedure

- Please read the [Release Notes and Upgrade Guides](#) for the version you are upgrading to, as well as any versions you are skipping.
- Follow the instructions on [upgrading Crucible](#).

Glossary

Code review takes many forms and has many names. For conciseness, Crucible has adopted the following terms and meanings:

[approve](#)

[author](#)

[code review](#)

[comment](#)

creator

defect

moderator

participant

permission

permission scheme

project

review duration

reviewer

role

state

statement of objective

user

approve

Issuing a review to the reviewers is known as *approving* the review.

author

The *author* is the person primarily responsible for acting on the outcomes of the review. In the vast majority of cases the author will be the person who made the code change under review.

Note: to map your repository username to your FishEye/Crucible username, see [Changing your User Profile](#).

code review

Without prejudice to 'code inspection', 'peer review' or a myriad of other terms, Crucible uses the phrase *code review* for simplicity.

See [About Crucible](#) and [Background Reading](#).

comment

A *comment* is a short textual note that is linked to a review, revision/diff, source line, or to another comment.

See [Adding Comments](#).

creator

The *creator* is the person who [creates the review](#). In most cases this person will also act as [moderator](#).

defect

A *defect* is a comment flagged as something that requires addressing and includes optional defect classifications.

See [Flagging Defects](#) and [Customising the Defect Classifications](#).

moderator

The *moderator* is the person responsible for [creating](#) the review, [approving](#) the review, determining when reviewing is finished, [summarising](#) the outcomes and [closing](#) the review. By default, the moderator is the [creator](#).

participant

Crucible uses the terms [creator](#), [author](#), [moderator](#), and [reviewer](#) to describe the *roles* of review participants.

permission

A *permission* is the ability to perform a particular action in Crucible, e.g. 'Create Review'. Permissions are assigned to particular users, groups or [review roles](#) by means of [permission schemes](#).

The following permissions are available:

Permission	Description	Default Assignees
'Edit'	Ability to edit a review's details and change the set of revisions being reviewed.	'Creator' 'Moderator'
'View'	Ability to view a review. (People without this permission will not know that the review exists.)	Anonymous users All logged-in users 'Creator' 'Author' 'Reviewer' 'Moderator'
'Abandon'	Ability to abandon (i.e. cancel) a review.	'Moderator' 'Creator'
'Re-Open'	Ability to re-open a closed or abandoned review.	'Creator' 'Moderator'
'Uncomplete'	Ability of a reviewer to change their individual review status from 'Complete' to 'Uncomplete'.	'Reviewer'
'Reject'	Ability to reject a review submitted for approval (i.e. prevent it from being issued to reviewers).	'Moderator'
'Complete'	Ability of a reviewer to change their individual review status to 'Complete'.	'Reviewer'
'Comment'	Ability to add or remove a comment to or from a review.	'Creator' 'Author' 'Reviewer' 'Moderator'
'Approve'	Ability to approve a review (i.e. issue it to the reviewers).	'Moderator'
'Submit'	Ability to submit a review for approval (i.e. request that the review be issued to the reviewers).	'Creator' 'Author'
'Close'	Ability to close a review once it has been summarised.	'Moderator'
'Delete'	Ability to delete a review.	'Creator' 'Moderator'
'Summarise'	Ability to summarise a review. (Normally this would be done after all reviewers have completed their review.)	'Moderator'
'Create'	Ability to create a review.	All logged-in users
'Recover'	Ability to resurrect an abandoned (i.e. cancelled) review.	'Creator' 'Moderator'

permission scheme

A *permission scheme* assigns particular [permissions](#) to any or all of the following:

- Particular [Users](#).
- Particular [Groups](#).
- All logged-in users.
- [Anonymous Users](#)
- People in particular [Review Roles](#), such as:
 - 'Author';
 - 'Reviewer';
 - 'Creator';
 - 'Moderator'.

The scheme's permissions will apply to all reviews belonging to the [project\(s\)](#) with which the scheme is associated.

You can create as many permission schemes as you wish. Each permission scheme can be associated with many projects or just one project, allowing you to tailor appropriate permissions for individual projects as required.

See [Creating a Permission Scheme](#).

project

A Crucible *project* is a collection of [reviews](#), typically reviews that all relate to the same application. In addition to providing a logical way of grouping reviews together, a project allows you to

- define default [moderators](#), [authors](#) and [reviewers](#) for the reviews in that project.
- define which people are eligible to be [reviewers](#) for the reviews in that project.
- use [permission schemes](#) to restrict who can perform particular actions (e.g. 'Create Review') in that project.

Every Crucible review belongs to a project. Each project has a *name* (e.g. **ACME Development**) and a *key* (e.g. **ACME**). The project key becomes the first part of that project's *review keys*, e.g. **ACME-101**, **ACME-102**, etc:

By default, Crucible contains one project. This default project has the key 'CR' and the name '**Default Project**'.

See [Creating a Project](#).

review duration

The *review duration* is the period of time for which a review will run.

See [Setting the Default Review Duration for a Project](#).

reviewer

A *reviewer* is a person assigned to [review the change](#). Reviewers can make [comments](#) and indicate when they have [completed their review](#). The [moderator](#) and [author](#) are implicitly considered reviewers.


role

See [participant](#).

state

A Crucible review moves through the following states in the following sequence:

Draft	See Creating a Review .
Require Approval	Relevant only when the moderator is not the creator . See Issuing a Review .
Under Review	See Issuing a Review and Reviewing the Code .
Summarize	See Summarising and Closing the Review .
Closed	See Summarising and Closing the Review .

 Reviews can be re-opened, i.e. moved from **Summarize** or **Closed** back to **Under Review**.

A review may also be in the following states:

Abandoned	This happens when a review is deleted.
Rejected	Any reviews that a moderator has rejected.

statement of objective

A *statement of objective* is an optional text description of the review and any specific areas the [reviewers](#) should focus on.

user

A *user* is a person using Crucible.