# 1. Crucible Documentation Home

## 1.1 About Crucible

### 1.1.1 Background Reading

## 1.2 Crucible User's Guide

### 1.2.1 Using the Crucible Screens

#### 1.2.1.1 Browsing Your Reviews

#### 1.2.1.2 Browsing Source Files

#### 1.2.1.3 Browsing Projects

#### 1.2.1.4 Viewing People's Statistics

#### 1.2.1.5 Searching in Crucible

#### 1.2.1.6 Using RSS Feeds in Crucible

#### 1.2.1.7 Using Keyboard Shortcuts in Crucible

### 1.2.2 Getting Started with Crucible

### 1.2.3 Using the Dashboard

#### 1.2.3.1 Using the Project Dashboard

### 1.2.4 Conducting a Review

#### 1.2.4.1 Creating a Review

##### 1.2.4.1.1 Creating a Review from FishEye

##### 1.2.4.1.2 Creating a Review from JIRA

##### 1.2.4.1.3 Creating a Review within Crucible

##### 1.2.4.1.4 Creating a Review from a URL

##### 1.2.4.1.5 Creating a Patch Review

##### 1.2.4.1.6 Selecting the Files for the Review

#### 1.2.4.2 Adding Reviewers

##### 1.2.4.2.1 Removing Reviewers From An Active Review

### 1.2.4.3 Issuing a Review

### 1.2.4.4 Performing the Review

#### 1.2.4.4.1 Adding Comments

#### 1.2.4.4.2 Flagging Defects

#### 1.2.4.4.3 Completing your Review

#### 1.2.4.4.4 Sending all of a Review's Comments via Email

### 1.2.4.5 Summarising and Closing the Review

### 1.2.4.6 Deleting an Abandoned Review

### 1.2.4.7 Moving a Review to Another Project

### 1.2.5 Changing your User Profile

### 1.2.6 Defining your Workflow

### 1.2.7 Roles and Status Classifications

### 1.2.8 User Favourites

## 1.3 Crucible Administrator's Guide

### 1.3.1 Backing Up and Restoring Crucible Data

### 1.3.2 Creating a Permission Scheme

#### 1.3.2.1 Associating a Permission Scheme with a Project

### 1.3.3 Creating a Project

#### 1.3.3.1 Setting Crucible to Store all Revisions

### 1.3.4 Managing and FishEye

### 1.3.5 Customising Email Notifications

#### 1.3.5.1 Freemarker Data Model for Email Templates

### 1.3.6 Customising the Defect Classifications

### 1.3.7 Customising the Welcome Message

### 1.3.8 Deleting a Project

### 1.3.9 Editing a Project

#### 1.3.9.1 Setting the Default Review Duration for a Project

### 1.3.10 JIRA Integration in Crucible

### 1.3.11 Migrating to an External Database

### 1.3.12 Trusted Applications

## 1.4 Crucible Development Hub

### 1.4.1 Documentation for Crucible Development

#### 1.4.1.1 Crucible's URL Structure

#### 1.4.1.2 Crucible REST API

##### 1.4.1.2.1 Crucible REST API Usage Example

##### 1.4.1.2.2 Conditional Get

##### 1.4.1.2.3 Data Types

##### 1.4.1.2.4 Project Service

##### 1.4.1.2.5 Repository Service

##### 1.4.1.2.6 Review Service

### 1.4.2 Crucible Plugin Types

#### 1.4.2.1 Crucible Web Items

### 1.4.3 Live Code Examples for Crucible Development

#### 1.4.3.1 Bundled Plugins from Crucible

##### 1.4.3.1.1 Confluence SCM Plugin

##### 1.4.3.1.2 File System SCM Plugin

##### 1.4.3.1.3 Perforce SCM Plugin

##### 1.4.3.1.4 Subversion SCM Plugin

### 1.4.3.2 SCM Plugin Examples

#### 1.4.3.2.1 Crucible ClearCase plugin
# 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

1. ![Performing the Review](#) (Crucible 2.0) by Rosie Jameson [Atlassian Technical Writer] (an hour ago)
2. ![file_outdated.png](#) (Crucible 2.0) by Rosie Jameson [Atlassian Technical Writer] (an hour ago)
3. ![file_outdated_menu.png](#) (Crucible 2.0) by Rosie Jameson [Atlassian Technical Writer] (2 hours ago)
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.
- 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.

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

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.

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

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.

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

### 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.

<table>
<thead>
<tr>
<th>Sub-Tab Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revisions</td>
<td>When viewing a file, shows the latest revisions of the file.</td>
</tr>
<tr>
<td>Files</td>
<td>When viewing a folder, shows the contents of the directory.</td>
</tr>
<tr>
<td>Activity</td>
<td>Shows recent activity on the item. There are a number of sub-options here:</td>
</tr>
<tr>
<td></td>
<td>- All Activity — The default view, showing commits, reviews and JIRA issues.</td>
</tr>
<tr>
<td></td>
<td>- Commits — Shows commits in the activity stream.</td>
</tr>
<tr>
<td></td>
<td>- Reviews — Shows review activity in the activity stream.</td>
</tr>
<tr>
<td></td>
<td>- Scroll to Changeset — Opens the changeset ID specified in the text field (press Enter to carry out the action).</td>
</tr>
<tr>
<td></td>
<td>- Filter — Applies constraints to the current activity stream.</td>
</tr>
<tr>
<td></td>
<td>- Show Revisions — If this is selected, then changeset items are automatically expanded to show modified files.</td>
</tr>
<tr>
<td></td>
<td>- Earlier Activity (Left Arrow icon) — Loads a page of earlier activity.</td>
</tr>
<tr>
<td></td>
<td>- Later Activity (Right Arrow icon) — Loads a page of later activity.</td>
</tr>
<tr>
<td>Users</td>
<td>Shows the commit history of the different users that have committed changes on the item.</td>
</tr>
<tr>
<td>Reports</td>
<td>Shows activity charts for the item. Various chart options can be selected in the left navigation bar.</td>
</tr>
<tr>
<td>Source</td>
<td>Shows the contents of the file.</td>
</tr>
<tr>
<td>Query</td>
<td>Allows you to run an advanced search.</td>
</tr>
</tbody>
</table>

*Screenshot: The Repositories View*
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.

<table>
<thead>
<tr>
<th>Sub-Tab Name</th>
<th>Description</th>
</tr>
</thead>
</table>
### Activity

- **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**

An image of the Crucible Projects View showing various activity details.

**Screenshot: The Crucible Projects Index**

An image of the Crucible Projects Index showing a list of projects.

---

### 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)**

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**

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**

<table>
<thead>
<tr>
<th>Heading</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Username</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Reviews</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Line History</strong></td>
<td>The Line History section shows a graph with the number of lines committed to the repository, charted over time.</td>
</tr>
<tr>
<td><strong>Commit Activity</strong></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Stats</strong></td>
<td>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.</td>
</tr>
</tbody>
</table>
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

'Everyone's 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':

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title</td>
<td>Find reviews by searching for words within the title.</td>
</tr>
<tr>
<td>Project</td>
<td>Find reviews under a particular project.</td>
</tr>
<tr>
<td>Author</td>
<td>Find reviews moderated by a particular authors.</td>
</tr>
<tr>
<td>Moderator</td>
<td>Find reviews moderated by a particular moderators.</td>
</tr>
<tr>
<td>Creator</td>
<td>Find reviews created by a particular creator.</td>
</tr>
<tr>
<td>Reviewer</td>
<td>Find reviews that are reviewed by a particular reviewer. This will default to the user logged in.</td>
</tr>
<tr>
<td>Status</td>
<td>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.</td>
</tr>
<tr>
<td>Match Roles</td>
<td>To use all the above filters, choose 'all'. To use any of the filters, choose 'any'.</td>
</tr>
</tbody>
</table>

Review State Filter

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

<table>
<thead>
<tr>
<th>Filter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft</td>
<td>Reviews that are still in 'Draft' state.</td>
</tr>
<tr>
<td>Pending Approval</td>
<td>Reviews that have been moved out of 'Draft' state and are now waiting for the moderator to approve.</td>
</tr>
<tr>
<td>Under Review</td>
<td>The review is now 'Under Review'.</td>
</tr>
<tr>
<td>Summarize</td>
<td>The review is now in 'Summarize' state.</td>
</tr>
<tr>
<td>Closed</td>
<td>Reviews that are now 'Closed'.</td>
</tr>
<tr>
<td>Abandoned</td>
<td>Any reviews that are no longer relevant and can be deleted.</td>
</tr>
<tr>
<td>Rejected</td>
<td>Any reviews that a moderator has rejected.</td>
</tr>
<tr>
<td>Review needs fixing</td>
<td>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.</td>
</tr>
</tbody>
</table>
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.
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*
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

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 Complete 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

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 Summarize button when finished. The Crucible Summarize Review screen opens.

Screenshot: Summarizing a review in Crucible
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

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.

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

**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.

<table>
<thead>
<tr>
<th>Sub-Tab Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity</td>
<td><img src="image" alt="All Activity — The default view." /> <img src="image" alt="Commits — Shows commits in the project (visible when using FishEye)." /> <img src="image" alt="Reviews — Shows reviews in the project." /> <img src="image" alt="Issues — Shows JIRA issues related to this project." /> <img src="image" alt="Show Revisions — Shows or hides revisions in the project (visible when using FishEye)." /> <img src="image" alt="Earlier Activity (Left Arrow icon) — Loads a page of earlier project activity." /> <img src="image" alt="Later Activity (Right Arrow icon) — Loads a page of later project activity." /></td>
</tr>
<tr>
<td>Reviews</td>
<td>Shows recent reviews in the project.</td>
</tr>
</tbody>
</table>

---

**Screenshot: The Crucible Projects View**

---

**Screenshot: The Crucible Projects Index**
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
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
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

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
Next Steps

- Use the ‘Manage Files’ tab to select the files to be reviewed. Read more details about selecting files for review.
- Use the ‘Changesets’, ‘Files’ or ‘Search’ tab to find the files to be added.
- Once you click Save, you’re ready to Add Reviewers.

Creating a Review from a URL

You can set up a URL which you can then click to create a Crucible review.

The format of your URL is as follows:

```
```

The parameters are as follows:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Required?</th>
</tr>
</thead>
<tbody>
<tr>
<td>csid</td>
<td>The changeset ID. You can specify one or more, of the form //repo/csidi (where “%2F” is the URL-encoded form of is ‘/’)</td>
<td>Yes</td>
</tr>
<tr>
<td>repo</td>
<td>The name of your repository.</td>
<td>Yes (unless supplied in the csid)</td>
</tr>
<tr>
<td>title</td>
<td>The title of your new Crucible review.</td>
<td>No</td>
</tr>
<tr>
<td>description</td>
<td>The description of your new Crucible review.</td>
<td>No</td>
</tr>
</tbody>
</table>

When you click the URL, 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:

- 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 (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 `cvs diff -Nu` command from your workspace. For example:

```
cvs 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:
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
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.

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

<table>
<thead>
<tr>
<th>Left Nav Title</th>
<th>Explanation</th>
<th>Special Conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review Content</td>
<td>Shows the list of files selected for review and allows you to manage them.</td>
<td></td>
</tr>
<tr>
<td>Change Sets</td>
<td>Allows you to choose change sets from a Source Code Management (SCM) repository and add them to a review.</td>
<td>Only appears when FishEye is installed.</td>
</tr>
<tr>
<td>Files</td>
<td>Allows you to choose files from a Source Code Management (SCM) repository and add them to a review.</td>
<td></td>
</tr>
<tr>
<td>Search</td>
<td>Allows you to search a Source Code Management (SCM) repository for files or change sets and add them to a review.</td>
<td>Only appears when FishEye is installed.</td>
</tr>
<tr>
<td>Suggestions</td>
<td>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).</td>
<td></td>
</tr>
<tr>
<td>Patches</td>
<td>Allows you to upload patch files to a review.</td>
<td></td>
</tr>
<tr>
<td>Uploads</td>
<td>Allows you to upload any file to a review, including binary files and files outside of a Source Code Management (SCM) repository.</td>
<td></td>
</tr>
</tbody>
</table>

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.
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**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repository</td>
<td>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.</td>
</tr>
<tr>
<td>Author</td>
<td>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.</td>
</tr>
<tr>
<td>Branch</td>
<td>This will only show files and recent changes on that branch from the repository set above.</td>
</tr>
<tr>
<td>Tag</td>
<td>This will only show files and recent changes tagged.</td>
</tr>
<tr>
<td>Add to Review As</td>
<td>Choose the form of the review; see full details.</td>
</tr>
<tr>
<td>Go to Changeset</td>
<td>Allows you to jump to a particular change set by entering its title and pressing Enter.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Icon and Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock icon or the file URL</td>
<td>A view detailing the history of this particular file.</td>
</tr>
<tr>
<td>Changeset ID next to the file URL</td>
<td>A view of the complete file. Amended lines are highlighted on the left in yellow.</td>
</tr>
<tr>
<td>Down arrow icon</td>
<td>Option to download the file.</td>
</tr>
<tr>
<td>Change Indicator (+n -n)</td>
<td>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.</td>
</tr>
</tbody>
</table>
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 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 remove 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.

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

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**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole File</td>
<td>Adds the entire file with all content, rather than just a diff with context.</td>
</tr>
<tr>
<td>Diff to Previous Version</td>
<td>This is the default behaviour, and how previous versions of Crucible worked by default.</td>
</tr>
<tr>
<td>An Iteration</td>
<td>This allows you to add multiple revisions of a file to one review and compare them in-review, in context with the change history.</td>
</tr>
<tr>
<td>Diff to Last Reviewed Version</td>
<td>This creates a review with a diff to the last reviewed changeset.</td>
</tr>
<tr>
<td>Diff to... (a particular revision)</td>
<td>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.</td>
</tr>
<tr>
<td>Whole file</td>
<td>The review content is not limited by changesets, showing the entire file contents.</td>
</tr>
</tbody>
</table>

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**

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,

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**

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:
   - Edit Review
   - Start Review
   - Tools
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

Test-27 test
Draft: Author & Moderator: Rosie Jameson
Draft for: 64 seconds
Comments: 1
Reviewers: Geoff Crain

Review Objectives
documentation testing

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**
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
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
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?

Reply Edit Delete Post Draft

Flagging Defects

Any comment that is not a reply to an existing comment can be flagged as a defect by its author.

Screenshot: Defects

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

Warning

You have draft comments
Draft comments that aren't posted will be deleted

View drafts
Delete drafts
Post drafts

Screenshot: Review complete
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 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.
4. When you have finished the distribution list, click the 'Next' button.
5. The 'Recipients' page appears (see Screenshot 3 below). On that page:
   6. In the 'To:' field you can enter multiple email addresses, separated by commas.
   7. When you have finished your message, click the 'Send' button.
6. 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

Screenshot 3: The 'Message' Screen in Crucible
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.

Review Closed

**Review Summary**
Testing successful. Thanks for your time, everyone.

Reviews requiring your attention:

<table>
<thead>
<tr>
<th>Task</th>
<th>Review</th>
<th>Owner</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review</td>
<td>CR-FE-2170</td>
<td></td>
<td>03 Jul</td>
</tr>
</tbody>
</table>
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.

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.
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
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.

<table>
<thead>
<tr>
<th>State change</th>
<th>Default is ‘Yes’. A Crucible review moves through different states e.g: 'Draft', 'Under Review'. An email is sent when the state changes.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Comment added</td>
<td>Default is ‘Yes’. An email is sent when a comment is added to a review.</td>
</tr>
<tr>
<td>Participant finished</td>
<td>Default is ‘Yes’. An email is sent (to the Moderator only) when any reviewer has completed their review.</td>
</tr>
<tr>
<td>General message</td>
<td>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.</td>
</tr>
<tr>
<td>My actions</td>
<td>Default is ‘No’. If set to ‘Yes’, an email is sent every time you perform an action on a review.</td>
</tr>
</tbody>
</table>

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*
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*
7. Author creates the review

Moderator invites reviewers to the review

Reviewers comment on the code

Author responds to comments

Discussion points are settled by Moderator

Moderator summarizes and closes the review
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.

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**

### Review Objectives

38809: CRUC-1616: Proper polling period and resetting of inactivity period.

**Review Comments**

 Nicolas Venegas says:

- sensible polling periods and backoff
- reset inactivity timer when you're actively using the review
- stop polling when window loses focus

**Add Star**

---

**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**

### Projects

<table>
<thead>
<tr>
<th>Project name</th>
<th>Latest Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Project</td>
<td>on 24 Jun 2009</td>
</tr>
<tr>
<td>Confluence Hosted</td>
<td>on 25 Feb 2009</td>
</tr>
<tr>
<td>Jira Studio</td>
<td>on 22 Apr 2008</td>
</tr>
<tr>
<td>FishEye</td>
<td>on 24 Jun 2009</td>
</tr>
<tr>
<td>Clover</td>
<td>on 24 Jun 2009</td>
</tr>
<tr>
<td>Clover IDEA plugin</td>
<td>on 24 Jun 2009</td>
</tr>
<tr>
<td>Testing Project</td>
<td>on 23 Jun 2009</td>
</tr>
<tr>
<td>Clover Eclipse Plugin</td>
<td>on 24 Jun 2009</td>
</tr>
</tbody>
</table>

**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.

**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.

**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.

**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.

**Viewing Your Favourite Items**
To view your favourite items, click the **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 to remove the item from your 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
Back 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

<table>
<thead>
<tr>
<th>Component</th>
<th>Purpose</th>
<th>Defaults</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL Database</td>
<td>Refers to the SQL content database (used by both FishEye and Crucible and containing all user profile data, reviews and their comments).</td>
<td>Backed up by default.</td>
</tr>
<tr>
<td>Cache</td>
<td>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.</td>
<td>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.</td>
</tr>
<tr>
<td>Plugins</td>
<td>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).</td>
<td>Configuration data for all plugins are backed up by default, as well as all plugins installed in FISHEYE_INST/var/plugins/user.</td>
</tr>
<tr>
<td>Templates</td>
<td>In this context, these are custom freemarker templates that you or your users have created. They live in FISHEYE_INST/template.</td>
<td>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.</td>
</tr>
<tr>
<td>Uploads</td>
<td>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.</td>
<td>Uploads are backed up by default. You can choose not to back them up for example when the FISHEYE_INST/var/data/uploads directory is already covered by some other backup mechanism.</td>
</tr>
</tbody>
</table>

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 ||

<table>
<thead>
<tr>
<th>Quiet mode</th>
<th>Suppresses output</th>
<th>-q OR --quiet</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output filename</td>
<td>Specify a different path and filename to the FISHEYE_INST/backup/backup/YYYY-DD-MM_HH.mm.zip file. When filename is omitted, the backup filename contains the date and time.</td>
<td>-f OR --file</td>
<td>FISHEYE_INST/backup/ is the default directory.</td>
</tr>
<tr>
<td>Compression level</td>
<td>Sets the Zip compression level, from 1-9. Runs at level 6 if no argument is passed.</td>
<td>--compression OR -c</td>
<td>Yes (6)</td>
</tr>
<tr>
<td>Anonymise</td>
<td>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).</td>
<td>-a OR --anonymise</td>
<td>No</td>
</tr>
<tr>
<td>Cache Backup</td>
<td>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.</td>
<td>--cache</td>
<td>No. By default, the cache data is excluded from backups.</td>
</tr>
</tbody>
</table>

## 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.

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

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](https://bugs.sun.com/bugdatabase/view_bug.do?bug_id=5210270). 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.

<table>
<thead>
<tr>
<th>Option name</th>
<th>Description</th>
<th>Allowed Values</th>
</tr>
</thead>
</table>
### Scheduling Backups

啶

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

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/ Crunchible 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.

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

### 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.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Projects using this scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>default</td>
<td>Default Project</td>
</tr>
</tbody>
</table>

**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](image)
### List of Crucible Permissions

The following permissions are available:

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

**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.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Anonymous</th>
<th>All Logged In</th>
<th>Individuals</th>
<th>Groups</th>
<th>Review Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abandon</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Moderator</td>
</tr>
<tr>
<td>Approve</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Moderator</td>
</tr>
<tr>
<td>Close</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Moderator</td>
</tr>
<tr>
<td>Comment</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
<tr>
<td>Complete</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer</td>
</tr>
<tr>
<td>Create</td>
<td>false</td>
<td>true</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Delete</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Moderator</td>
</tr>
<tr>
<td>Edit Review Details</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Moderator</td>
</tr>
<tr>
<td>Recover</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Moderator</td>
</tr>
<tr>
<td>Reject</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Author, Moderator</td>
</tr>
<tr>
<td>Re-Open</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Moderator</td>
</tr>
<tr>
<td>Submit</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Author, Moderator</td>
</tr>
<tr>
<td>Summarize</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Moderator</td>
</tr>
<tr>
<td>Uncomplete</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer</td>
</tr>
<tr>
<td>View</td>
<td>false</td>
<td>true</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
</tbody>
</table>

⚠️ 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.

<table>
<thead>
<tr>
<th>Permission</th>
<th>Anonymous</th>
<th>All Logged In</th>
<th>Individuals</th>
<th>Groups</th>
<th>Review Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abandon</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Author, Moderator</td>
</tr>
<tr>
<td>Approve</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Author, Moderator</td>
</tr>
<tr>
<td>Close</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
<tr>
<td>Comment</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
<tr>
<td>Complete</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer</td>
</tr>
<tr>
<td>Create</td>
<td>false</td>
<td>true</td>
<td>None</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Delete</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Author, Moderator</td>
</tr>
<tr>
<td>Edit Review Details</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
<tr>
<td>Recover</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
<tr>
<td>Reject</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Author, Moderator</td>
</tr>
<tr>
<td>Re-Open</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
<tr>
<td>Submit</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Creator, Author, Moderator</td>
</tr>
<tr>
<td>Summarize</td>
<td>false</td>
<td>false</td>
<td>None</td>
<td>None</td>
<td>Reviewer, Creator, Author, Moderator</td>
</tr>
</tbody>
</table>
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
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.
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.
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.

<table>
<thead>
<tr>
<th>Template filename</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>crucible-notification-subject.ftl</td>
<td>Subject template</td>
</tr>
<tr>
<td>crucible-notification-header.ftl</td>
<td>Header template</td>
</tr>
<tr>
<td>crucible-notification-footer.ftl</td>
<td>Footer template</td>
</tr>
<tr>
<td>state-closed-notification.ftl</td>
<td>State Closed template</td>
</tr>
<tr>
<td>all-completed-notification.ftl</td>
<td>All Completed template</td>
</tr>
<tr>
<td>state-changed-notification.ftl</td>
<td>State Changed template</td>
</tr>
</tbody>
</table>
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.

```text
${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**
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.

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

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

```html
<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`:

```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

<table>
<thead>
<tr>
<th>Key</th>
<th>Name</th>
<th>Default Repository</th>
<th>Default Moderator</th>
<th>Default Reviewer Users</th>
<th>Default Reviewer Groups</th>
<th>Default Review Time</th>
<th>Edit Crucible Settings</th>
<th>Edit FishEye Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>CR</td>
<td>Default Project</td>
<td>CLOV</td>
<td></td>
<td></td>
<td></td>
<td>No time restriction</td>
<td>edit</td>
<td>view/edit</td>
</tr>
<tr>
<td>CR-CH</td>
<td>Confluence Hosted</td>
<td>CH</td>
<td></td>
<td></td>
<td></td>
<td>No time restriction</td>
<td>edit</td>
<td>delete</td>
</tr>
<tr>
<td>CR-JST</td>
<td>Jira Studio</td>
<td>JST</td>
<td>Don Brown</td>
<td></td>
<td></td>
<td>No time restriction</td>
<td>edit</td>
<td>view/edit</td>
</tr>
</tbody>
</table>

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.
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*.
8. *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.
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 weeks) 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*

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Text Field</td>
<td>A descriptive name for the JIRA server.</td>
<td>Yes</td>
</tr>
<tr>
<td>URL</td>
<td>Text Field</td>
<td>The Internet address of the JIRA server.</td>
<td>Yes</td>
</tr>
<tr>
<td>Subtask Type ID</td>
<td>Number</td>
<td>This is required to enable creating issues from a Crucible comment.</td>
<td>No</td>
</tr>
<tr>
<td>Subtask Resolution Action ID</td>
<td>Number</td>
<td>This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.</td>
<td>No</td>
</tr>
<tr>
<td>Subtask Resolution ID</td>
<td>Number</td>
<td>This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.</td>
<td>No</td>
</tr>
<tr>
<td>Allow Unassigned</td>
<td>True/False Button</td>
<td>Allow unassigned sub-tasks.</td>
<td>No</td>
</tr>
<tr>
<td>Username</td>
<td>Text Field</td>
<td>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).</td>
<td>Yes</td>
</tr>
<tr>
<td>Password</td>
<td>Text Field</td>
<td>The password for the account on the JIRA instance.</td>
<td></td>
</tr>
<tr>
<td>Include in Activity Streams</td>
<td>Check Box</td>
<td>Allows JIRA information to appear on the Dashboard. (Requires a JIRA instance with the streams plugins installed)</td>
<td>No</td>
</tr>
<tr>
<td>Authenticate as Trusted Application</td>
<td>Check Box</td>
<td>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.</td>
<td>No</td>
</tr>
</tbody>
</table>

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 'Edit' 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
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.

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**

![](image.png)

**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](https://dev.mysql.com/downloads/jdbc/) 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:

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

3. You will also need to set the Server Character set 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]`:
4. Use the `status` command to verify database character encoding information:

```
mysqld> status
```

```
.Expression: Using the MySQL Status Command

mysql> status
------------------
/Applications/xampp/xamppfiles/bin/mysql Ver 14.12 Distrib 5.0.67, for apple-darwin8.11.1 (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: Connect via UNIX socket
Server charset: utf8
Db charset: utf8
Conn charset: latin1
UNIX socket: /Applications/xampp/xamppfiles/var/mysql/mysql.sock
Uptime: 29 days 1 hour 34 min 56 sec

Threads: 6 Questions: 49230134 Slow queries: 0 Opens: 460445 Flush tables: 1
Open tables: 4 Queries per second avg: 19.607

------------------
```

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 Database

Migrating database...

From: jdbc:hsqldb:file:/Users/ervizjst/centrqua/demotion/var/data/crurd/crurdle
To: jdbc:mysql://localhost/crurdle

Started
Creating table definitions...
- Hsq1: 1 rows written, 6 tables completed.
- Usrs: 9 rows written, 1 tables completed.
- Avrs: 7227 rows written, 2 tables completed.
- Grps: 11143 rows written, 3 tables completed.
- Ads: 12874 rows written, 3 tables completed.
- Vists: 12700 rows written, 4 tables completed.
- Chs: 22735 rows written, 6 tables completed.
- Cols: 30492 rows written, 5 tables completed.
- Cls: 30493 rows written, 5 tables completed.
- Shop: 30197 rows written, 6 tables completed.

Ok
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):

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

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

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

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

```sql
> 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.
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 it 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Id</th>
<th>Url Patterns</th>
<th>IP Address Patterns</th>
<th>Edit</th>
<th>Delete</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="https://extranet.atlassian.com">https://extranet.atlassian.com</a></td>
<td>confluence: 3427555</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
- Crucible SCM Plugins

Event Listeners
- Event Listener Plugins

Servlets
- 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
- Subversion SCM Plugin
- Perforce SCM Plugin
- Confluence SCM Plugin
- File System SCM Plugin

Other SCM Plugins
- Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)
- Crucible ClearCase plugin

Servlet Examples
- 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.

<table>
<thead>
<tr>
<th>How to Build a Crucible Plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to Build a Crucible Plugin - start here to learn how to set up your development environment, create a plugin template and start coding.</td>
</tr>
<tr>
<td>Development Platform for Crucible</td>
</tr>
<tr>
<td>Crucible API Javadocs</td>
</tr>
<tr>
<td>Crucible REST API</td>
</tr>
</tbody>
</table>

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
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
<?xml version="1.0" encoding="UTF-8" standalone="yes"?><loginResult><token>jim:2:4455f9a4387298a83aaee6902e8843f89</token></loginResult>
```

The value of the cookie should be set to jim:2:4455f9a4387298a83aaee6902e8843f89.

**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:

```xml
<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>CR-FE-1</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:
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:

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 be able to create a new review, so use Basic HTTP authentication for this call):

That particular review item is a new file, so the fromPath and fromRevision elements are empty.
**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:**
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"
 }},
 "versionedComments":null
}}
```

Response:
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 (\(\text{Last-Modified} \leq \text{If-Modified-Since}\)), 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](https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html).

**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:**

```
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 (\(\text{Last-Modified} \leq \text{If-Modified-Since}\)), 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](https://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html).

**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:**

```
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"
}}
```
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:

```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>CR-3</permaId>
  <projectKey>CR</projectKey>
  <state>Review</state>
</reviewData>
```

Sample JSON:

```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"
}}
```
Sample JSON
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.: 


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:
<actionData>
  <name>action:recoverReview</name>
</actionData>

<actionData>
  <name>action:createReview</name>
</actionData>

<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>
Sample JSON:

```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": "evzijs",  
          "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": "evzijs",
          "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).
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:

```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:

```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)
...
"
}
```
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:

```xml
<svnRepositoryData>
  <enabled>true</enabled>
  <name>local</name>
  <type>svn</type>
  <url>file:///Users/tomd/dev/svn/</url>
</svnRepositoryData>
```

Example JSON:

```json
{"svnRepositoryData": {
  "enabled":true,
  "name":"Local",
  "type":"svn",
  "path":null,
  "url":"file:///Users/tomd/dev/svn/"
}}
```

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
  • 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

Comments

Add a Reply to a Comment
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:

```xml
<comments>
  <versionedLineCommentData>
    ...
  </versionedLineCommentData>
</comments>
```

Example JSON Return Data:

```json
{"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:

```xml
<comments>
  <versionedLineCommentData>
    ...
  </versionedLineCommentData>
  <versionedLineCommentData>
    ...
  </versionedLineCommentData>
</comments>
```

Example JSON Return Data:

```json
{"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:**

```
{

}
```
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:

```xml
<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/>
<permaIdAsString>CMT:1</permaIdAsString>
<replies/>
<user>
<displayName>Matt Quail</displayName>
<userName>matt</userName>
</user>
<permId>
<id>CMT:1</id>
</permId>
</generalCommentData>
</comments>
```
<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>
Example JSON Return Data:

```json
{"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"}},
                       {"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": "2008-10-30T16:33:02.726+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

**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
<?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>
  </metricsData>
</metrics>
```
<label>Ranking</label>

<name>rank</name>
<type>INTEGER</type>

<values>
  <name>Major</name>
</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>

<configVersion>1</configVersion>

<defaultValue>
  <name>Improvement desirable</name>
</defaultValue>

<name>Classification</name>
<type>INTEGER</type>

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


</values>
Example JSON Return Data:

```json
{
  "metrics": {
    "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:
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.

```json
{
"reviewers": {
  "reviewer": [
    {
      "displayName": "Peter Moore",
      "userName": "pete",
      "completed": false
    },
    {
      "displayName": "Brendan Humphreys",
      "userName": "brendan",
      "completed": false
    }
  ]
}
```
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 Edition
  - 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

```html
<!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="/addFile" 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 PermId 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 reponse 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:
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):

```
<createReview>
  <reviewData>
  ...
  </reviewData>
  <patch>
    <![CDATA[
      ... text of patch goes here ...
    ]]>  
  </patch>
</createReview>
```

```
{createReview: {
  "reviewData": {
    "allowReviewersToJoin":false,
    "author":{
      "userName":"joe"},
    "creator":{"username":"joe"},
    "moderator":{"userName":"matt"},
    "description":"Review objectives",
    "metricsVersion":1,
    "name":"Stuff to review",
    "projectKey":"CR"
  },
  "patch":"... text of patch goes here ..."}
}
```

3. A review containing revisions from a set of changesets. XML/JSON:
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:**

```xml
<createReview>
  <reviewData>...
  </reviewData>
  <changesets>
    <changesetData>
      <id>...the id...</id>
    </changesetData>
    ... more changesets ...
  </changesets>
</createReview>
```

**Example JSON Return Data:**

```json
{
  "createReview": {
    "reviewData": false,
    "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"
    }
  }
}
```
Example JSON Return Data:

```json
{"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:**

```xml
<detailedReviews>
  <detailedReviewData>
    ...
  </detailedReviewData>
  ...
</detailedReviews>
```

**Example JSON Return Data:**

```json
{"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:

```xml
<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:
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:
```xml
<reviewData>
...
</reviewData>
```

Example JSON Return Data:
```json
{"reviewData": {
 "allowReviewersToJoin":false,
 ... 
}}
```

Get Review Details

URL:
```plaintext
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:
```xml
<detailedReviews>
<detailedReviewData>
...
</detailedReviewData>
...
</detailedReviews>
```

Example JSON Return Data:
```json
{"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:

```xml
<detailedReviews>
  <detailedReviewData>
    ...
  </detailedReviewData>
  ...
</detailedReviews>
```

Example JSON Return Data:

```json
{"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:
Example JSON Return Data:

```json
{
"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:

```xml
<reviews>
 <reviewData> 
 ... 
 </reviewData> 
 ... 
</reviews>
```

Example JSON Return Data:
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:

```xml
<reviews>
  <reviewData>...
  ...
</reviewData>
  ...
</reviews>
```

Example JSON Return Data:

```json
{"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:

```xml
<reviews>
  <reviewData>
    ...
  </reviewData>
  ...
</reviews>
```

Example JSON Return Data:

```json
{"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:

```xml
<detailedReviewData>
  ...
</detailedReviewData>
```

Example JSON Return Data:

```json
{"detailedReviewData": { "allowReviewersToJoin":false, ...
  }}
```

Get Version Info
GET /reviews-v1/versionInfo

**Description:**
Returns the release number and build date of Crucible.

**Status Code:**
200 (OK) on success.

**Example XML Return Data:**

```xml
<versionInfo>
  <buildDate>2008-10-28</buildDate>
  <releaseNumber>1.6.3</releaseNumber>
</versionInfo>
```

**Example JSON Return Data:**

```json
{"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.

<table>
<thead>
<tr>
<th>Source Code Management (SCM) Plugins</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Crucible SCM Plugins</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Servlets</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Servlet Modules</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Event Listeners</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Event Listener Plugins</td>
</tr>
</tbody>
</table>

**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
Web Items Listing and Reference

<table>
<thead>
<tr>
<th>Key</th>
<th>Description</th>
<th>Helpers available</th>
</tr>
</thead>
<tbody>
<tr>
<td>system.admin</td>
<td>Links on the admin menu. Sections: repositories, global, system.</td>
<td>application, user</td>
</tr>
<tr>
<td>system.crucible.dashboard</td>
<td>After 'My Dashboard' in the dropdown project/dashboard selector menu</td>
<td>application, user</td>
</tr>
<tr>
<td>system.crucible.review</td>
<td>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.</td>
<td>application, user, project, review</td>
</tr>
<tr>
<td>system.crucible.review.comment</td>
<td>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.</td>
<td>application, user, project, review, comment</td>
</tr>
<tr>
<td>system.crucible.review.fileitem</td>
<td>Actions which can be performed on a revision in a review. Displayed next to the 'Remove', 'Change Diff' buttons.</td>
<td>application, user, project, review, reviewItem, repository, comment</td>
</tr>
<tr>
<td>system.header.application</td>
<td>Links like 'Crucible' and 'Fisheye' in the header. Sections: fisheye, crucible (sections are shown when a particular application is selected)</td>
<td>application, user</td>
</tr>
<tr>
<td>system.header.item</td>
<td>Links in the header, separated by a pipe.</td>
<td>application, user</td>
</tr>
<tr>
<td>system.main</td>
<td>Links on the Crucible or Fisheye main page. These appear at the bottom of the Crucible or FishEye main page. Sections: fisheye, crucible</td>
<td>application, user</td>
</tr>
<tr>
<td>system.userprofile.tab</td>
<td>The user profile tabs.</td>
<td>application, user</td>
</tr>
</tbody>
</table>

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.
The following conditions take parameters:

- HasProjectPermission
- HasReviewPermission
- IsReviewInState

The usage and conditions that these parameters apply to are tabled below.

<table>
<thead>
<tr>
<th>Parameter Value</th>
<th>Parameter Name</th>
<th>Description</th>
<th>Applies to</th>
</tr>
</thead>
<tbody>
<tr>
<td>action:abandonReview</td>
<td>actionName</td>
<td>Causes the current review to be abandoned.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
<tr>
<td>action:approveReview</td>
<td>actionName</td>
<td>Causes the current review to be approved.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
<tr>
<td>action:closeReview</td>
<td>actionName</td>
<td>Causes the current review to be closed.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
<tr>
<td>action:recoverReview</td>
<td>actionName</td>
<td>Causes the current review to be recovered.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
<tr>
<td>action:reopenReview</td>
<td>actionName</td>
<td>Causes the current review to be re-opened.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
<tr>
<td>action:rejectReview</td>
<td>actionName</td>
<td>Causes the current review to be rejected.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
<tr>
<td>action:submitReview</td>
<td>actionName</td>
<td>Causes the current review to be submitted.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
<tr>
<td>action:summarizeReview</td>
<td>actionName</td>
<td>Causes the current review to be summarised.</td>
<td>HasProjectPermission, HasReviewPermission</td>
</tr>
</tbody>
</table>

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
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 Dashboard](image)

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

![Open Reviews](image)

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

![Comment](image)

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

![File Item](image)
system.header.application

This item relates to product name links in the Crucible header.

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
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
- Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)
- Crucible ClearCase plugin

### Servlet Examples
- 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Crucible Confluence SCM plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.0-SNAPSHOT</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.6.2 and later</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Atlassian</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CRUCIBLE/">http://confluence.atlassian.com/display/CRUCIBLE/</a></td>
</tr>
<tr>
<td>Price</td>
<td>Bundled with Crucible</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>IssueTracking</td>
<td>Jira</td>
</tr>
<tr>
<td></td>
<td>FishEye: <a href="http://svn.atlassian.com/fisheye/browse/public/contrib/crucible/crucible-confluence-scm-plugin">http://svn.atlassian.com/fisheye/browse/public/contrib/crucible/crucible-confluence-scm-plugin</a></td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Crucible File System SCM plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.2</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.6.0+</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Atlassian</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CRUCIBLE/">http://confluence.atlassian.com/display/CRUCIBLE/</a></td>
</tr>
<tr>
<td>Price</td>
<td>Bundled with Crucible</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>Not Applicable</td>
</tr>
<tr>
<td>IssueTracking</td>
<td><a href="http://developer.atlassian.com/jira/browse/CRU">http://developer.atlassian.com/jira/browse/CRU</a></td>
</tr>
</tbody>
</table>

**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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Crucible Perforce SCM plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.2</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.6.4+</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Atlassian</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CRUCIBLE/">http://confluence.atlassian.com/display/CRUCIBLE/</a></td>
</tr>
<tr>
<td>Price</td>
<td>Bundled with Crucible</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
</tbody>
</table>
### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Crucible Subversion SCM plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.2</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.6.0+</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Atlassian</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CRUCIBLE/">http://confluence.atlassian.com/display/CRUCIBLE/</a></td>
</tr>
<tr>
<td>Price</td>
<td>Bundled with Crucible</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>Not Applicable</td>
</tr>
</tbody>
</table>

### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Crucible ClearCase plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0.1.0</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.6.1 to 2.0</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Ross Rowe</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CRUCIBLE/Crucible+ClearCase+plugin">http://confluence.atlassian.com/display/CRUCIBLE/Crucible+ClearCase+plugin</a></td>
</tr>
<tr>
<td>Price</td>
<td>Free</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>TBA</td>
</tr>
<tr>
<td>IssueTracking</td>
<td>Jira</td>
</tr>
<tr>
<td>Subversion URL</td>
<td><a href="https://svn.atlassian.com/svn/public/contrib/crucible/crucible-clearcase-plugin">https://svn.atlassian.com/svn/public/contrib/crucible/crucible-clearcase-plugin</a> or browse via fisheye</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name for the repository eg. Project</td>
</tr>
<tr>
<td>Main Component</td>
<td>The component containing the source files eg. project_main@pvob</td>
</tr>
<tr>
<td>Integration Stream</td>
<td>The integration stream for your project eg. stream:Project_Integration@pvob</td>
</tr>
<tr>
<td>View Location</td>
<td>The location of the main component for the view (snapshot or dynamic) for the project eg. c:/projects/project/project_main</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1.0</td>
<td>7 Jul 2009</td>
<td>CCCP-3 Updated plugin to support Crucible 2.0</td>
</tr>
<tr>
<td>0.0.5</td>
<td>4 Mar 2009</td>
<td>Updated fix for CCCP-1</td>
</tr>
<tr>
<td>0.0.4</td>
<td>10 Feb 2009</td>
<td>Recompiled plugin to support Crucible 1.6.6</td>
</tr>
<tr>
<td>0.0.3</td>
<td>4 Feb 2009</td>
<td>Included fix for CCCP-1</td>
</tr>
<tr>
<td>0.0.2</td>
<td>11 Nov 2008</td>
<td>Updated plugin to support Crucible 1.6.3</td>
</tr>
<tr>
<td>0.0.1</td>
<td>1 Oct 2008</td>
<td>Initial version of plugin</td>
</tr>
</tbody>
</table>

Screenshots

There are no images attached to this page.

Crucible Git Plugin

<table>
<thead>
<tr>
<th>Name</th>
<th>Crucible Git plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.0</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.6.6</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Atlassian</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CRUCIBLE/Crucible+Git+plugin">http://confluence.atlassian.com/display/CRUCIBLE/Crucible+Git+plugin</a></td>
</tr>
<tr>
<td>Price</td>
<td>Free</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>TBA</td>
</tr>
<tr>
<td>IssueTracking</td>
<td>JIRA</td>
</tr>
<tr>
<td>Subversion URL</td>
<td>Subversion FishEye</td>
</tr>
<tr>
<td>Download JAR</td>
<td>Atlassian Maven Repository (supports Crucible 1.6.6)</td>
</tr>
<tr>
<td>Download Source</td>
<td>TBA</td>
</tr>
</tbody>
</table>

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:
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name for the repository eg. Project</td>
</tr>
<tr>
<td>Repository Path</td>
<td>The location of the local Git repository clone</td>
</tr>
</tbody>
</table>

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**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>10 Feb 2009</td>
<td>Early Access as part of Crucible 1.6.6</td>
</tr>
</tbody>
</table>

**Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)**

<table>
<thead>
<tr>
<th>Name</th>
<th>Example Crucible SCM Plugin for JSR-170 (Apache JackRabbit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.0-SNAPSHOT</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.6.3</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Erik van Zijst, Atlassian</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CRUCIBLE/">http://confluence.atlassian.com/display/CRUCIBLE/</a></td>
</tr>
<tr>
<td>Price</td>
<td>Free</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>Crucible SCM API JavaDoc</td>
</tr>
<tr>
<td>Browse Source</td>
<td>FishEye</td>
</tr>
<tr>
<td>Download Source</td>
<td>Subversion</td>
</tr>
</tbody>
</table>

**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.jdk:jdk: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:
Failed to resolve artifact.

---

1) com.sun.jdmk:jmxtools:jar:1.2.1

Try downloading the file manually from:


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

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0-SNAPSHOT</td>
<td>10-November-2008</td>
<td>Initial release</td>
</tr>
</tbody>
</table>

Screenshots
Servlet Examples

This page lists the Servlet code examples for Crucible plugin developers.

- Basic Servlet Example
- Crucible Reporting plugin

## Basic Servlet Example

<table>
<thead>
<tr>
<th>Name</th>
<th>Example FishEye Servlet Plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>0.1</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.7</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Anna Buttfield, Atlassian</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/FISHEYE/">http://confluence.atlassian.com/display/FISHEYE/</a></td>
</tr>
<tr>
<td>Price</td>
<td>Free</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>Data Package Summary</td>
</tr>
<tr>
<td></td>
<td>Services Package Summary</td>
</tr>
<tr>
<td>IssueTracking</td>
<td>N/A</td>
</tr>
<tr>
<td>Subversion URL</td>
<td>FishEye svn</td>
</tr>
<tr>
<td>Download JAR</td>
<td>Attached to this page: compiled jar sources jar</td>
</tr>
<tr>
<td>Download Source</td>
<td>svn</td>
</tr>
</tbody>
</table>

**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

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>7-November-2008</td>
<td>Initial release</td>
</tr>
</tbody>
</table>

Screenshots

Crucible Reporting plugin

<table>
<thead>
<tr>
<th>Name</th>
<th>Crucible Reporting plugin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>1.0.0</td>
</tr>
<tr>
<td>Product Versions</td>
<td>1.5.x, 1.6.x</td>
</tr>
<tr>
<td>Author(s)</td>
<td>Ross Rowe</td>
</tr>
<tr>
<td>Homepage</td>
<td><a href="http://confluence.atlassian.com/display/CODEGEIST/Crucible+Reporting+plugin">http://confluence.atlassian.com/display/CODEGEIST/Crucible+Reporting+plugin</a></td>
</tr>
<tr>
<td>Price</td>
<td>Free</td>
</tr>
<tr>
<td>License</td>
<td>BSD</td>
</tr>
<tr>
<td>JavaDocs</td>
<td>crucible-export-plugin-javadoc.zip</td>
</tr>
<tr>
<td>IssueTracking</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td>or browse via fisheye</td>
</tr>
<tr>
<td>Download JAR</td>
<td>crucible-export-1.0.0.jar (for Crucible 1.5.x) / crucible-export-1.6.2.jar (for Crucible 1.6.x)</td>
</tr>
<tr>
<td>Download Source</td>
<td>crucible-export.zip</td>
</tr>
</tbody>
</table>
**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](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.

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**

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6.2</td>
<td>22 Mar 2009</td>
<td>CRPT-2 Sort versioned comments for a specific file (thanks Soren!)</td>
</tr>
<tr>
<td>1.6.1</td>
<td>10 Dec 2008</td>
<td>Added ability to include defect and revision information in report (thanks Soren!)</td>
</tr>
<tr>
<td>1.6.0</td>
<td>22 Sep 2008</td>
<td>Updated plugin to support Crucible 1.6.0 beta</td>
</tr>
</tbody>
</table>

**Crucible 1.5 support**
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 \n   -DarchetypeGroupId=com.atlassian.maven.archetypes \n   -DarchetypeArtifactId=crucible-plugin-archetype \n   -DarchetypeVersion=1-SNAPSHOT \n   -DremoteRepositories=https://maven.atlassian.com/repository/ \n   -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:

```bash
export FISHEYE_OPTS="-Xdebug -Xrunjdwp: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:

```xml
.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:

```java
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`:
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:

```bash
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`.

Note that this is a provided dependency – the plugin does not need to include the `atlassian-events` classes.
<?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>
    <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:
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 SCMModule 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 SCMModule to read and write the configuration:

public class ExampleSCMModule implements SCMModule, Configurable<List<ExampleConfiguration>> {
    private ModuleDescriptor moduleDescriptor;
    private ModuleConfigurationStore store;

    public ExampleSCMModule(ModuleConfigurationStore store) {
        this.store = store;
    }

    [...]

    public List<ExampleConfiguration> getConfiguration() {
        byte[] configData = store.getConfiguration(moduleDescriptor);
        if (configData != null) {
            try {
                return (List<ExampleConfiguration>)getXStream().fromXML(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.

```java
package com.atlassian.scm;
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;
import class ExampleSCMRepository

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 config.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 {
```
 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 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()`:

```java
public synchronized Collection<SCMRepository> getRepositories() {
    if (repos == null) {
        repos = new ArrayList<SCMRepository>();
        for (ExampleConfiguration config : getConfiguration()) {
            repos.add(new ExampleSCMRepository(config));
        }
    }
    return repos;
}
```

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`:
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:

```java
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, one to edit a repository's configuration 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
<html>
<head>
    <link rel="stylesheet" href="$request.contextPath/$STATICDIR/main.css" type="text/css" />
</head>
<body>
    <div class="box formPane">
        <table class="adminTable">
            #if ($configs.empty)
            <tr><td>No File System repositories are configured.</td></tr>
            #else
            <tr><td>Name</td><td>Base Path</td><td>Edit</td><td>Delete</td></tr>
            #foreach ($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>
```
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:
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**

```
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Version</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared Application Access Layer API</td>
<td>Base POM for Atlassian projects</td>
<td>1.0.0.b-01</td>
<td>Enabled</td>
</tr>
<tr>
<td>Example File System SCM</td>
<td>Example SCM implementation for local file system</td>
<td>Enabled</td>
<td>Disable</td>
</tr>
<tr>
<td>Example File System SCM</td>
<td>Allows Configuration of File System example SCM Plugin</td>
<td>Enabled</td>
<td>Disable</td>
</tr>
<tr>
<td>crucible-svn-scm-plugin</td>
<td>An SCM provider for Subversion</td>
<td>1.0-SNAPSHOT</td>
<td>Disabled</td>
</tr>
</tbody>
</table>
```

Next, click “Configure” to create a file system based repository:

**Screenshot: Creating a File-System Based Repository**
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*

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 the 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 SCMModule 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 its 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.

<table>
<thead>
<tr>
<th>Java Runtime</th>
<th>A JDK or JRE version 1.5 or greater.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>You can download a Java Runtime for Windows/Linux/Solaris <a href="#">here</a>.</td>
</tr>
<tr>
<td></td>
<td>On Mac OS X, the JDK is bundled with the operating system.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
</tbody>
</table>

| 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 | Crucible has been tested with Firefox 3, Internet Explorer 7, and Safari 4. IE6 is NOT supported. Crucible should work with most modern browsers. |

Font size

(Optional 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.

| 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/](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

<table>
<thead>
<tr>
<th>Field</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name:</td>
<td>Example Server</td>
</tr>
<tr>
<td>Repository Server:</td>
<td>example.com:666</td>
</tr>
<tr>
<td>Repository Path:</td>
<td>//depot/code/example/main/</td>
</tr>
<tr>
<td>Perforce Username:</td>
<td>admin</td>
</tr>
<tr>
<td>Perforce Password:</td>
<td>***************</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Choose a unique name for the repository.</td>
</tr>
<tr>
<td>Repository Root</td>
<td>Enter the repository root URL for the repository. If you are not sure what the repository root is, please see the instructions below under &quot;Finding your Repository Root&quot;.</td>
</tr>
<tr>
<td>Repository Path</td>
<td>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 '<a href="http://svn.example.com/svn5/">http://svn.example.com/svn5/</a>', you would enter 'svn5' into this field.</td>
</tr>
<tr>
<td>SVN Username</td>
<td>Enter the username of the Subversion account that Crucible will use. Note that this account should only have read-only access to the repository.</td>
</tr>
<tr>
<td>SVN Password</td>
<td>Enter the password of the Subversion account that Crucible will use.</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Choose a unique name for the repository.</td>
</tr>
<tr>
<td>URL</td>
<td>Enter the URL of your Confluence instance.</td>
</tr>
<tr>
<td>Space Key</td>
<td>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.</td>
</tr>
</tbody>
</table>

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/](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**

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Choose a unique name for the repository.</td>
</tr>
<tr>
<td>Base Path</td>
<td>Choose the lowest level of directory that Crucible will access.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name for the repository eg. Project</td>
</tr>
<tr>
<td>Repository Path</td>
<td>The location of the local Git repository clone</td>
</tr>
</tbody>
</table>

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

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Choose a unique name for the repository.</td>
</tr>
<tr>
<td>Repository Server</td>
<td>Enter the base URL and port for the repository, for example: example.com:666.</td>
</tr>
<tr>
<td>Repository Path</td>
<td>Add the path to your Perforce repository. For example: //depot/code/example/main.</td>
</tr>
<tr>
<td>Perforce Username</td>
<td>Enter the username of the Perforce account that Crucible will use. (optional)</td>
</tr>
<tr>
<td>Perforce Password</td>
<td>Enter the password of the Perforce account that Crucible will use. (optional)</td>
</tr>
</tbody>
</table>

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

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>subversion5</td>
</tr>
<tr>
<td>Repository Root</td>
<td><a href="http://svn.example.com">http://svn.example.com</a></td>
</tr>
<tr>
<td>Repository Path</td>
<td>svn5</td>
</tr>
<tr>
<td>SVN Username</td>
<td>subversion-admin</td>
</tr>
<tr>
<td>SVN Password</td>
<td>**********</td>
</tr>
</tbody>
</table>

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:

```bash
>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

To set up Confluence as a Code Repository in Crucible,

1. 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.
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.

<table>
<thead>
<tr>
<th>Field</th>
<th>What to enter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Choose a unique name for the repository.</td>
</tr>
<tr>
<td>URL</td>
<td>Enter the URL of your Confluence instance.</td>
</tr>
<tr>
<td>Space Key</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

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
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.

1. 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.)
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. Shutdown 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 you 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 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

#### 1

**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.
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.
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
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

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
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
External Databases and Backup

Crucible now supports MySQL and Postgres in addition to the embedded HSQL database. Backup and restore capabilities have also been greatly enhanced.

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:
<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-1616</td>
<td>Tune polling period.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1615</td>
<td>Reset the delay when actively using the review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1612</td>
<td>Move the warning to a Gmail-style overlay</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-59</td>
<td>External database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1656</td>
<td>RC3 - NPE when trying to create first project on empty CRU install</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-824</td>
<td>If we haven't already we should automatically create a backup of our CRUDB (and other configuration files) before upgrading</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1661</td>
<td>File marked as unreviewed for defect author</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1651</td>
<td>clicking [add] branch in SVN admin symbolic screen causes page-pop and 404</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1649</td>
<td>toggling Display Preferences &gt; Show Source in a review does nothing</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1644</td>
<td>Some display preferences don't update when clicked</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1641</td>
<td>error mapping JIRA servers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1623</td>
<td>Admin: Default reviewers are not re-saved</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1611</td>
<td>Commenting doesn't work in Safari 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1609</td>
<td>CAC documentation for iterative review info from CRUC-1536</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1481</td>
<td>Warning dialogs and polling when new stuff added since last refresh</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1404</td>
<td>Review throws 500 error NPE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1403</td>
<td>Rename &quot;crucibledb&quot; to &quot;sql&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1399</td>
<td>Default Reviewer for project is not exposed through the SPI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1398</td>
<td>overdue but review has not been started. Setting a due date in the past causes this.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1393</td>
<td>Refactor Email Comments Page into a Dialog</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1392</td>
<td>fix remove file behaviour of current content panel</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1391</td>
<td>Use newDirTreeBox in the files pane</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1390</td>
<td>Fix Uploads</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-1388</td>
<td>Missing css files on Send Email page / Layout needs love</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>--------------------------------------------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>CRUC-1386</td>
<td>fix suggest reviewer ui</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1383</td>
<td>suggest reviewers - turn blame calculation back on</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1382</td>
<td>defect metrics charts are broken</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1381</td>
<td>.png file not detected as binary</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1380</td>
<td>Email Review button from the dashboard doesn't work / Email review in tools menu gives a 404 - permaid is undefined</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1379</td>
<td>clicking HELP opens help twice / Help in Tools menu will open a new window and change your window</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1378</td>
<td>page load error &quot;The requested URL /atlaseye/static/kidi0j/2static/script/fe/fisheye-profile.js was not found on this server.&quot;</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1377</td>
<td>wrong review actions (comments.txt and Email Review) are visible</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1376</td>
<td>we need to handle multiple errors in a smarter way</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1375</td>
<td>There is no jira quick link in the edit review dialog</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1374</td>
<td>Review info popup on the dashboard shouldn't show when clicking on a link such as the review permaid</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1373</td>
<td>Page load error &quot;The requested URL /atlaseye/static/kidi0j/2static/images/sprite_arrows.png was not found on this server&quot;</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1371</td>
<td>External dialog 'View …' links need to redirect to review and trigger filter</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1370</td>
<td>Show current content panel in manage files</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1369</td>
<td>make the side filters like the fisheye tabs</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1368</td>
<td>bring back the custom filter</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1367</td>
<td>make the pages consistent</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1366</td>
<td>Add warnings for incomplete FRXs and unresolved JIRAs to dialogs</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1365</td>
<td>Move review summary form into a dialog.</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1364</td>
<td>Move creation-time review suggestions into a dialog</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1363</td>
<td>Convert project selection screen to a dialog</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1362</td>
<td>Backup broke Crucible Blitz</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1361</td>
<td>General Issues</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1360</td>
<td>Manage Files</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1359</td>
<td>Edit review form</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-----------------</td>
<td>---------</td>
<td></td>
</tr>
<tr>
<td>CRUC-1358</td>
<td>Interstitials/dialogs</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1357</td>
<td>Dashboard Fixes</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1356</td>
<td>Remove driver url parameter from Restore and use new DBInfo properties</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1354</td>
<td>Refactor programatic frx filtering to not invoke click events</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1351</td>
<td>Add path and base filename to backup UI</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1345</td>
<td>Extend QuartzManager to support persistent configuration data per job</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1344</td>
<td>Rest Api returns the wrong URL for download Patch</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1330</td>
<td>Put custom quartz job scheduling in a separate tab</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1329</td>
<td>Dynamically update general comment counts</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1328</td>
<td>Add general comments to the tree</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1327</td>
<td>Add the BackupItem selection to the scheduled backup</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1325</td>
<td>Add file type css place holders</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1319</td>
<td>Update doco for admin backup page</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1317</td>
<td>Dynamically update comment counts</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1316</td>
<td>Add comment counts to the frx tree</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1305</td>
<td>Springify Backup Manager</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1303</td>
<td>Write unit test</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1294</td>
<td>Check start/stop no longer needed in restore</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1293</td>
<td>Render maintenance page for non-admin pages when a backup is being made</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1291</td>
<td>Add “Change Diff” dropdown to manipulate both fromRev and toRev independently</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1290</td>
<td>Collapse common directories</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1289</td>
<td>Apply tree html structure and css styling to list</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1288</td>
<td>Treeify FRX list pane</td>
<td>Resolved</td>
<td></td>
</tr>
</tbody>
</table>

---

CRUC-1359: Edit review form
CRUC-1358: Interstitials/dialogs
CRUC-1357: Dashboard Fixes
CRUC-1356: Remove driver url parameter from Restore and use new DBInfo properties
CRUC-1354: Refactor programatic frx filtering to not invoke click events
CRUC-1351: Add path and base filename to backup UI
CRUC-1345: Extend QuartzManager to support persistent configuration data per job
CRUC-1344: Rest Api returns the wrong URL for download Patch
CRUC-1330: Put custom quartz job scheduling in a separate tab
CRUC-1329: Dynamically update general comment counts
CRUC-1328: Add general comments to the tree
CRUC-1327: Add the BackupItem selection to the scheduled backup
CRUC-1325: Add file type css place holders
CRUC-1319: Update doco for admin backup page
CRUC-1317: Dynamically update comment counts
CRUC-1316: Add comment counts to the frx tree
CRUC-1305: Springify Backup Manager
CRUC-1303: Write unit test
CRUC-1294: Check start/stop no longer needed in restore
CRUC-1293: Render maintenance page for non-admin pages when a backup is being made
CRUC-1291: Add “Change Diff” dropdown to manipulate both fromRev and toRev independently
CRUC-1290: Collapse common directories
CRUC-1289: Apply tree html structure and css styling to list
CRUC-1288: Treeify FRX list pane
<p>| 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 |</p>
<table>
<thead>
<tr>
<th>CRUC-1171</th>
<th>XML-RPC bollocks</th>
<th></th>
<th>Closed</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-1170</td>
<td>Delete a defect's linked JIRA subtask when deleting the defect</td>
<td>✔</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1169</td>
<td>ui &amp; action</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1168</td>
<td>XML-RPC bollocks</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1167</td>
<td>Resolve a defect's linked JIRA subtask from Crucible</td>
<td>✔</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1166</td>
<td>XML-RPC bollocks</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1165</td>
<td>UI display of jira link and status + post error creating</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1164</td>
<td>schema</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1163</td>
<td>Automatically add defects as subtasks to JIRA &amp; link to them</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1162</td>
<td>guessing + jira validation</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1161</td>
<td>edit details ui + actions</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1160</td>
<td>schema</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1159</td>
<td>Link a review to a single JIRA and enable auto defect creation</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1158</td>
<td>just do it</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1157</td>
<td>Configure a JIRA instance for use by Crucible</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1156</td>
<td>admin UI + create review</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1155</td>
<td>schema</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1154</td>
<td>Configure a default review duration for a project in week days so the due date can be pre-populated</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1153</td>
<td>highlighting overdue</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1152</td>
<td>sorting in the lists</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1151</td>
<td>Order any list of reviews by due date &amp; highlight any that are overdue</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1150</td>
<td>due date field in UI</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1149</td>
<td>schema</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1148</td>
<td>Set or change a due date on a review</td>
<td></td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1143</td>
<td>just do it</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>----</td>
<td>--------</td>
</tr>
<tr>
<td>CRUC-1142</td>
<td>Combine diff and annotation views</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1141</td>
<td>just do it</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1140</td>
<td>show the reviews you can't update without buttons</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1139</td>
<td>Author colours</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1138</td>
<td>Invisible markers (between lines)</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1137</td>
<td>Mouseover</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1136</td>
<td>Improve comment marker behaviour (tetris bar)</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1134</td>
<td>replace yahoo shite if we can</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1133</td>
<td>event binding</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1132</td>
<td>error handling</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1131</td>
<td>namespaces &amp; multiple files</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1130</td>
<td>remove prototype</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1129</td>
<td>client side js model</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1128</td>
<td>refactor commentator.js and friends</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1127</td>
<td>just do it</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1126</td>
<td>handle modifier keypress</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1125</td>
<td>just do it</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1124</td>
<td>Popup a keyboard shortcut help overlay when i press ´?´</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1123</td>
<td>add user pref</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1122</td>
<td>add complete status resetter to the defect notification</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1121</td>
<td>Reset my complete status when new defects are added to a review</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1119</td>
<td>Tweak comment scroll tracker behaviour</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1118</td>
<td>Improve revision slider behaviour</td>
<td>✅</td>
<td>Resolved</td>
</tr>
<tr>
<td>CRUC-1117</td>
<td>add unread comment counts per revision</td>
<td>✅</td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-1114</td>
<td>fix rendering/script issue in IE</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>-----------</td>
<td>----------------------------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>CRUC-1113</td>
<td>make the slider handles move in unison</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1108</td>
<td>Mark comment read when it hits top of the viewport</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-1097</td>
<td>Remove the use of half closed TCP connections</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1083</td>
<td>render author + participant results</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-1048</td>
<td>Change the admin backup page to use the new backup/restore implementation</td>
<td>Resolved</td>
<td></td>
</tr>
<tr>
<td>CRUC-927</td>
<td>Add a warning in the product that re-sync in large LDAP servers can cause performance issue</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-922</td>
<td>Admin screens with config.xml write and create/migrate workflow</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-921</td>
<td>Set up maintenance Mode</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-920</td>
<td>set up tests to compare upgrades of each database type</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-918</td>
<td>Write tests to determine/compare expected results from database tables</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-915</td>
<td>Usage Documentation for web-based backup/restore feature</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-912</td>
<td>Integration Tests</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-910</td>
<td>Admin Backup Page</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-909</td>
<td>Backup Status Page</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-907</td>
<td>Online Backup/Restore</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-882</td>
<td>REST API return other people's drafts</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-861</td>
<td>Allow soft wrap on diff and annotated display</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-845</td>
<td>Describe this in the CAC docs.</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-823</td>
<td>Add an Expand Unchecked Files option</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-770</td>
<td>Support file upload via REST</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-603</td>
<td>Support Anonymous Users Properly</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-581</td>
<td>Include Light SCM Repos in information returned by REST API</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-466</td>
<td>Improve built-in group management</td>
<td>Closed</td>
<td></td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>JIRA Issues (35 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>CRUC-1501</td>
</tr>
<tr>
<td>CRUC-1496</td>
</tr>
</tbody>
</table>

From 2.0 Beta2 to 2.0 Beta3

5th June 2009

Full list of issues in this release:
From 1.6.6 to 2.0 Beta2

Full list of issues in this release:
**JIRA Issues (13 issues)**

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-1488</td>
<td>Improve user interface in the frx tree</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1478</td>
<td>Single frx visible mode</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1476</td>
<td>File revision comments aren't autosaved</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1465</td>
<td>Synchronise comment buttons when states are changed</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1464</td>
<td>Fix rev slider labels so that slider can snap</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1463</td>
<td>User has no display name</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1462</td>
<td>Selecting changeset in Manage Files fails</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1458</td>
<td>Account signup pages still use old styles</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1454</td>
<td>Timezone problem in review popup</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1439</td>
<td>Crucible Standalone - Manage Files links to FishEye</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1396</td>
<td>Empty reviews cannot be abandoned/closed/viewed</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1116</td>
<td>No notifications sent for comments created through REST</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
<tr>
<td>CRUC-1427</td>
<td>Old error page for disabled FishEye Urls</td>
<td>🟡</td>
<td>🍃 Closed</td>
</tr>
</tbody>
</table>

---

**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.
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

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

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

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**

<table>
<thead>
<tr>
<th>Due</th>
<th>Reviewers</th>
</tr>
</thead>
<tbody>
<tr>
<td>15 days ago</td>
<td><img src="image" alt="Avatar" /></td>
</tr>
<tr>
<td>11 days ago</td>
<td><img src="image" alt="Avatar" /></td>
</tr>
<tr>
<td>10 days ago</td>
<td><img src="image" alt="Avatar" /></td>
</tr>
<tr>
<td>3 days ago</td>
<td><img src="image" alt="Avatar" /></td>
</tr>
<tr>
<td>4 hours ago</td>
<td><img src="image" alt="Avatar" /></td>
</tr>
<tr>
<td>3 hours ago</td>
<td><img src="image" alt="Avatar" /></td>
</tr>
<tr>
<td>79 minutes ago</td>
<td><img src="image" alt="Avatar" /></td>
</tr>
</tbody>
</table>

**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**
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
Plus numerous improvements and bug fixes

Visit our issue tracker 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.

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 you 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.
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.
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**

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

<table>
<thead>
<tr>
<th>Option</th>
<th>Type</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Text Field</td>
<td>A descriptive name for the JIRA server.</td>
<td>Yes</td>
</tr>
<tr>
<td>URL</td>
<td>Text Field</td>
<td>The Internet address of the JIRA server.</td>
<td>Yes</td>
</tr>
<tr>
<td>Subtask Type ID</td>
<td>Number</td>
<td>This is required to enable creating issues from a Crucible comment.</td>
<td>No</td>
</tr>
<tr>
<td>Subtask Resolution Action ID</td>
<td>Number</td>
<td>This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.</td>
<td>No</td>
</tr>
<tr>
<td>Subtask Resolution ID</td>
<td>Number</td>
<td>This is required to enable creating issues from a Crucible comment. See below for instructions on obtaining this number.</td>
<td>No</td>
</tr>
<tr>
<td>Allow Unassigned</td>
<td>True/False Button</td>
<td>Allow unassigned sub-tasks.</td>
<td>No</td>
</tr>
<tr>
<td>Username</td>
<td>Text Field</td>
<td>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).</td>
<td>Yes</td>
</tr>
<tr>
<td>Password</td>
<td>Text Field</td>
<td>The password for the account on the JIRA instance.</td>
<td>Yes</td>
</tr>
<tr>
<td>Include in Activity Streams</td>
<td>Check Box</td>
<td>Allows JIRA information to appear on the Dashboard.</td>
<td>No</td>
</tr>
<tr>
<td>Authenticate as Trusted Application</td>
<td>Check Box</td>
<td>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.</td>
<td>No</td>
</tr>
</tbody>
</table>

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

---

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
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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>URL</th>
<th>Subtask ID</th>
<th>Subtask Resolution ID</th>
<th>Allow Unassigned</th>
<th>Activity Stream Provider</th>
<th>Trusted Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>jac</td>
<td><a href="https://jira.atlassian.com">https://jira.atlassian.com</a></td>
<td>6</td>
<td>2</td>
<td>true</td>
<td>true</td>
<td>false</td>
</tr>
</tbody>
</table>

**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

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.

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.
Enhanced Pre-commit Reviews & Image Support

In addition to Crucible's patch support, 1.6 enables any file to be uploaded for review. The new upload functionality enables two files to be uploaded and compared in the review window, with diff highlighting. Read more.

Crucible now supports before and after inline image previews.

Screenshot: Crucible Review of Uploaded Files
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.

Expanded API

The Crucible API now allows programmable review creation, along with a host of other additions. Read more.

Plus numerous improvements and bug fixes

<table>
<thead>
<tr>
<th>JIRA Issues (109 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
</tr>
<tr>
<td>CRUC-661</td>
</tr>
<tr>
<td>CRUC-646</td>
</tr>
<tr>
<td>CRUC-608</td>
</tr>
<tr>
<td>CRUC-539</td>
</tr>
<tr>
<td>CRUC-227</td>
</tr>
<tr>
<td>CRUC-668</td>
</tr>
<tr>
<td>CRUC-664</td>
</tr>
<tr>
<td>CRUC-663</td>
</tr>
<tr>
<td>CRUC-659</td>
</tr>
<tr>
<td>CRUC-654</td>
</tr>
<tr>
<td>CRUC-647</td>
</tr>
<tr>
<td>CRUC-645</td>
</tr>
<tr>
<td>CRUC-640</td>
</tr>
<tr>
<td>CRUC-639</td>
</tr>
</tbody>
</table>
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 commit 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

CRUC-551  Confluence Light SCM plugin dependencies belong in plugin

CRUC-549  Adjust IFRAME size when the admin page is resized

CRUC-547  Make Repository source check if the engine is available.

CRUC-545  NPE Closing Review

CRUC-542  Use content manager to retrieve detailed file revisions

CRUC-526  Space character in installation path to Crucible disables remote API

CRUC-519  Added files display "no change"

CRUC-518  Crucible does not seem to be incrementally updating the FishEye cache

CRUC-516  REST API: Return detailed information on review in one call - items, comments, reviewers, available actions

CRUC-511  Added files when stored show "No Change"

CRUC-507  Confluence: implement change set paging

CRUC-505  Add "Search for Review" functionality to Crucible remote API

CRUC-504  Confluence: get revision comments

CRUC-499  Create separate API jar

CRUC-496  Document store revisions with review

CRUC-494  Don't make source type visible in URLs or elsewhere

CRUC-488  clicking summarise without clicking close makes the review 'disappear' from the workflow

CRUC-484  you can link a review to itself

CRUC-478  Ability to specify that Branches are not supported

CRUC-472  Changing tab while editing review deletes details

CRUC-461  Line comments are sometimes rendered twice

CRUC-456  crucible doesn't create default permission scheme when creating a blank db

CRUC-455  creating a review from a changeset with a non-existent repo keeps redirecting to the login screen

CRUC-453  Allow adding a screenshot as an attachment to a review
CRUC-439 Invite to review
CRUC-438 Not sending email notifications after Crucible and Fisheye upgrade
CRUC-436 create ‘remove all revision’ link for all tabs under ‘manage files’
CRUC-433 REST API: Get review list based on predefined and custom filters
CRUC-422 Create Schema Upgrade to remove foreign key constraint as reqd. by delete project work
CRUC-407 Minimal OSGi Infrastructure
CRUC-404 Turn FE Off When only Crucible licence is present
CRUC-402 Provide Plugin Authors with somewhere to store properties
CRUC-400 Add Configuration UI to SVN Plugin, and Polish
CRUC-397 Stored FRX Create UI
CRUC-396 Stored FRX Data Impl
CRUC-394 Add Light SCM Revisions to Reviews
CRUC-392 Filesystem Light SCM Plugin
CRUC-391 Default Repository for a Project can be a Light SCM repo
CRUC-390 Light SCM Plugin instances appear in source/repository dropdowns
CRUC-388 Subversion LSCM plugin
CRUC-387 Confluence LSCM Plugin
CRUC-386 Modify File Browser to Use Light SCM Plugins
CRUC-385 Modify Changeset Browser to use Light SCM Plugins
CRUC-384 Create Light SCM Module Type
CRUC-381 Admin Pages for Plugins
CRUC-373 Add General Comment via REST API
CRUC-362 Return replies to comments via API
CRUC-361 Project RSS feed
CRUC-349 REST method /rest-service/reviews-v1/CR-1/comments returns HTTP 500 error
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 authorname 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:

<table>
<thead>
<tr>
<th>JIRA Issues (2 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>CRUC-1406</td>
</tr>
<tr>
<td>CRUC-900</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>JIRA Issues (6 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>CRUC-947</td>
</tr>
<tr>
<td>CRUC-938</td>
</tr>
<tr>
<td>CRUC-928</td>
</tr>
<tr>
<td>CRUC-867</td>
</tr>
<tr>
<td>CRUC-886</td>
</tr>
<tr>
<td>CRUC-683</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>JIRA Issues (27 issues)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
</tr>
<tr>
<td>CRUC-753</td>
</tr>
<tr>
<td>CRUC-926</td>
</tr>
<tr>
<td>CRUC-862</td>
</tr>
<tr>
<td>CRUC-846</td>
</tr>
<tr>
<td>CRUC-844</td>
</tr>
<tr>
<td>CRUC-837</td>
</tr>
<tr>
<td>---</td>
</tr>
<tr>
<td>CRUC-835</td>
</tr>
<tr>
<td>CRUC-834</td>
</tr>
<tr>
<td>CRUC-832</td>
</tr>
<tr>
<td>CRUC-827</td>
</tr>
<tr>
<td>CRUC-826</td>
</tr>
<tr>
<td>CRUC-821</td>
</tr>
<tr>
<td>CRUC-809</td>
</tr>
<tr>
<td>CRUC-793</td>
</tr>
<tr>
<td>CRUC-788</td>
</tr>
<tr>
<td>CRUC-787</td>
</tr>
<tr>
<td>CRUC-778</td>
</tr>
<tr>
<td>CRUC-765</td>
</tr>
<tr>
<td>CRUC-748</td>
</tr>
<tr>
<td>CRUC-727</td>
</tr>
<tr>
<td>CRUC-582</td>
</tr>
<tr>
<td>CRUC-398</td>
</tr>
<tr>
<td>CRUC-820</td>
</tr>
<tr>
<td>CRUC-802</td>
</tr>
<tr>
<td>CRUC-328</td>
</tr>
<tr>
<td>CRUC-255</td>
</tr>
<tr>
<td>CRUC-568</td>
</tr>
</tbody>
</table>

**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 Perforce 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)

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-729</td>
<td>Can't review binary file becoming textual</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-725</td>
<td>Light SCM allows creation of repos with the same name as Fisheye repos</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-701</td>
<td>Support Perforce repositories in RepositoryService</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-486</td>
<td>Invalid rendering of Search Comments report (table part)</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-785</td>
<td>any screenshots of scroll to changeset for doco</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-784</td>
<td>filesystem lightscm plugin should not use “current” as the revision name</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-777</td>
<td>rework for CR-FE-697 CRUC-728</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-776</td>
<td>rework for CR-FE-702 CRUC-624</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-775</td>
<td>Create unit tests that verify JSON serialization</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-773</td>
<td>Add JSON support to Release Notes</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-772</td>
<td>Add documentation on how to use JSON to Confluence</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-771</td>
<td>Improve the I&amp;F of manage files</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-769</td>
<td>Add JSON support to REST</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-768</td>
<td>Upgrade to Jersey 1.0, and include jersey-spring for possible future springification</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-767</td>
<td>Autosave race condition</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-764</td>
<td>Add this change to the release notes</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-759</td>
<td>Improve Confluence Light SCM Performance</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-758</td>
<td>make scroll to changeset look better</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-757</td>
<td>Custom filter object in Crucible REST should not use primitive values</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-752</td>
<td>[crucible] in the closed review email subject</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-750</td>
<td>Plugin config change will affect user configs</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-749</td>
<td>Get Selenium Tests working</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-744</td>
<td>summary email includes deleted comments</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-739</td>
<td>Getting error popup on review due to confusion about a directory.</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-736</td>
<td>Automatically save draft comments (autosave)</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-733</td>
<td>Crucible without FishEye still says FishEye in titles</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-724</td>
<td>Implement Light SCM plugin for Perforce</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-723</td>
<td>Converting Crucible reviews from FishEye repo to LightSCM SVN fails to load revisions</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-722</td>
<td>Remove the use of deprecated CrucibleRevision.getSource()</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-712</td>
<td>Create unit tests for REST</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-711</td>
<td>document Alt+Click for selecting review text</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-708</td>
<td>Change REST filter retrieval from POST to GET</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-706</td>
<td>Retrieving non-existing metrics version in REST gives 500 &quot;Internal Server Error&quot;, should be 404</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-702</td>
<td>summary email documentation</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-700</td>
<td>add maybe-details and maybe-filehistory to FileSummary</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-699</td>
<td>Revision Details should be a map</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-698</td>
<td>maybe-details provided by SCMs are not used</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-697</td>
<td>Change ManageFiles tab so that it does not require so much information from the SCM</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-696</td>
<td>Address performance problems in LightSCM plugin API</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-695</td>
<td>Crucible REST should throw an exception when adding changesets to reviews that already have comments</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-694</td>
<td>Adding changesets on open reviews messes up the in-line comments</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-693</td>
<td>Still cannot delete projects</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-691</td>
<td>Implement Conditional Get in REST API</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-688</td>
<td>FR_EXTRA.FRX_ORDER needs a unique constraint</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-680</td>
<td>Changeset dates are wrong</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-679</td>
<td>Add xwork action that returns the text summary (for copying and pasting).</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-678</td>
<td>Add send summary button and form</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-677</td>
<td>Update summary template to include comments</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-671</td>
<td>Create review from changeset gives HTTP 500</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-670</td>
<td>Improve REST error reporting (HTTP return codes)</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-658</td>
<td>Scroll To: box for changelogs</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-653</td>
<td>[admin] adding a &quot;default reviewer&quot; incorrectly adds an &quot;allowed reviewer&quot; in some cases</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-627</td>
<td>make stored reviews viewable when the source isn't available</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-624</td>
<td>XML Parsing Error from Crucible Review Service using allReviews filter</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-509</td>
<td>Add info about IDE integration in Web UI</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-470</td>
<td>Summary Email Improvements</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-395</td>
<td>Stored FRX</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-383</td>
<td>Maven Archetype for plugin developers</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-380</td>
<td>Light SCM</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-327</td>
<td>Copy and paste is not working</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-763</td>
<td>DELETE operations in Crucible REST should return status 204 &quot;No Content&quot; to be more RESTful</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-747</td>
<td>Auto save draft comment needs to delete the draft on cancel</td>
<td>Closed</td>
<td></td>
</tr>
<tr>
<td>CRUC-742</td>
<td>Images in patches don't work</td>
<td>Closed</td>
<td></td>
</tr>
</tbody>
</table>
CRUC-735  Error message in Crucible file management is misplaced
CRUC-728  add CSID and SOURCE_URL to lightSCM details
CRUC-713  Change REST return code from 200 to 201 "Created" for several POST actions
CRUC-710  You do not have permission to see all the search results, seen in (my) to summarise and out for review
CRUC-709  Refactor exception handling issues
CRUC-705  Create RestXxxServices via Spring
CRUC-689  Revision details missing from choose diff dropdown
CRUC-687  <i>no comment</i> commit message tooltip
CRUC-667  re address update of revision details
CRUC-665  Correct documentation for REST API for getting file information
CRUC-656  Previously deleted files show as having an old version in a review when they are added again
CRUC-655  Downloaded files do not have the correct file name
CRUC-529  Have a way to select text in source windows
CRUC-475  Create documentation on creating reviews using remote API
CRUC-536  Improper code colorization for C++

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
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

1

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
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

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.

4

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

<table>
<thead>
<tr>
<th>JIRA Issues (38 issues)</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-344 Defect pie chart on comment search page includes non-defect comments.</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-332 create a better looking indication icon</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-324 Add Project Dropdown to Search Pages</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-323 Create ‘Reviews Started’ Pie Chart for Project Dashboard</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-318 Make Project Dashboard Charts cover 14 days instead of 7</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-314 sidebyside diff line numbers must be aligned to the top</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-307 Optimize chart queries</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-299 link to project page</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-289 permissions are not checked in rss feeds (but are in list html display)</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-285 &quot;project project&quot; in Permission Screen in admin</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-283 Escape HTML in Javascript properly</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-280 remove debug log from Process Notification</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-247 Flag files which have been modified since a review was created</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-246 Parentheses should be valid URL characters</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-242 Add RPC call to create review from patch</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-240 Checking box in &quot;manage files&quot; -&gt; &quot;changesets&quot; spins forever</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-232 Address all issues outline here</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-220 permalink for a comment does not work when summarize mode is active</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-212 Create Project Dashboard page</td>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>
CRUC-206 URL recognition in Crucible comment does not always match whole string

CRUC-201 Allow project to be specified on create review URL

CRUC-194 Documentation: restoring Crucible backup data

CRUC-184 When Creating Review From FishEye, Default to Right Project

CRUC-172 Create ‘Getting started with Crucible’ document as part of User Guide revisions

CRUC-147 Add Permissions Checks to All Operations on Reviews

CRUC-128 email notification of review closure prints literal “null” for absent field value

CRUC-85 Sometimes we need to see differences in UNICODE files.

CRUC-21 change icon for "View History In Fisheye"

CRUC-290 Warnings during backup: File does not exist /data/crucible/inst/var/data/crudb/crucible.lck & File does not exist /data/crucible/inst/var/data/crudb/crucible.log

CRUC-282 Crucible fails to upload patch file with file information that contain blank spaces between file path and timestamp

CRUC-266 let the subject of the email be modifiable via the notification template

CRUC-260 How to customize the email notification content

CRUC-248 Configurable email subject line

CRUC-241 Incorrect dates in RSS feeds

CRUC-230 Crucible allows circular review linking and can’t delete a review link

CRUC-226 Title wrong for post-approval in manage files tab

CRUC-162 From field on emails is incorrect or a least deceiving

CRUC-238 Perforce unit tests do not work on Windows platform

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)

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-638</td>
<td>How to (easily) add specific changeset to a review</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-513</td>
<td>If we display no change on a revision, comments made are hidden by default.</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-503</td>
<td>Upgrade to crowd-integration-client 1.4.4</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-498</td>
<td>Draft comment replies are deleted when completing review</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-454</td>
<td>two notification emails sent</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-429</td>
<td>Prevent two projects with same key different case</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-284</td>
<td>AnnotatorTag error: Could not get blame for file</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-633</td>
<td>How to remove a reviewer from a review</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-538</td>
<td>Comments weirdly duplicated on occasion</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-417</td>
<td>large var/data/crudb/crucible.log</td>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>

### From 1.5.2 to 1.5.3

**23 June 2008**

This release contains bug fixes.

#### JIRA Issues (1 issues)

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-462</td>
<td>&quot;File closed: var\data\data0.bin&quot; after backups</td>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>

### From 1.5.1 to 1.5.2

**27 May 2008**

This release contains bug fixes.

#### JIRA Issues (16 issues)

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-262</td>
<td>Urgent : SVNRepository methods are not reenterable</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-405</td>
<td>[Performance] AnnotatorTag can hang in sort loop for (perhaps) large diffs</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-370</td>
<td>create action to remove all revisions from a review</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-363</td>
<td>Remove all Revisions in manager files-&gt; Changesets tab doesn't work</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-355</td>
<td>Review Stats Wrong</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-354</td>
<td>Personal dashboard stats don't match</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-336</td>
<td>can't search for review keys in quicksearch</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-185</td>
<td>Abandoned Reviews Still Show in FishEye</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-434</td>
<td>'Expand Commented' appears when there are general comments but no revision comments</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-413</td>
<td>Crucible always creates a new default project after it is deleted</td>
<td></td>
<td>Closed</td>
</tr>
<tr>
<td>CRUC-406</td>
<td>Remove 'All Projects' from projects dropdown</td>
<td></td>
<td>Closed</td>
</tr>
</tbody>
</table>
### JIRA Issues

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-364</td>
<td>Starting Crucible 1.5 Spits Lots of Errors Out</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-367</td>
<td>LOC exceptions filling Debug log to the tune of 3-6GB per day</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-353</td>
<td>Crucible seems not to be able to handle uploaded git patches</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-350</td>
<td>javax.servlet.ServletException</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-304</td>
<td>Add REST method to retrieve review items for a given review</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-291</td>
<td>NullPointerException thrown when invoking the getAllRevisionComments web service method</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-161</td>
<td>Create document stating when a customer purchases Crucible that they do not need to have the latest version of FishEye</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-357</td>
<td>Sparklines have wrong content type</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-333</td>
<td>In IE, &quot;view diff to latest&quot; partially obscured</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**From 1.5.0 to 1.5.1**

**24 April 2008**

This release contains bug fixes.

### 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](http://jira.atlassian.com/browse/CRUC). 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: | 
| Key: |
| Repository
| Default Repository: | svn |
| Moderator
| Default Moderator: | Start typing a user name then press enter to select. |
| Default Reviewers
| 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
| Permission Scheme: | default |
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.

<table>
<thead>
<tr>
<th>Permissions</th>
<th>Users / Groups / Review Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modify Files</td>
<td>Anonymous users: false</td>
</tr>
<tr>
<td>Ability to change the set of revisions being reviewed.</td>
<td>edit</td>
</tr>
<tr>
<td>Close</td>
<td>Anonymous users: false</td>
</tr>
<tr>
<td>Ability to close a review once it has been summarized.</td>
<td>edit</td>
</tr>
<tr>
<td>View Review</td>
<td>Anonymous users: true</td>
</tr>
<tr>
<td>Ability to view a review.</td>
<td>edit</td>
</tr>
<tr>
<td>Uncomplete</td>
<td>Anonymous users: false</td>
</tr>
<tr>
<td>Ability to indicate they have not completed a review, after indicating they have completed a review.</td>
<td>edit</td>
</tr>
<tr>
<td>Re-Open</td>
<td>Anonymous users: false</td>
</tr>
<tr>
<td>Ability to re-open a closed review.</td>
<td>edit</td>
</tr>
</tbody>
</table>

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.

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](https://fisheye.redhat.com/)
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:

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.

Plus over 20 improvements and bug-fixes

<table>
<thead>
<tr>
<th>JIRA Issues (32 issues)</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
<td><strong>Summary</strong></td>
<td><strong>Priority</strong></td>
</tr>
<tr>
<td>CRUC-181</td>
<td>Old screenshot in 1.2 release notes</td>
<td></td>
</tr>
<tr>
<td>CRUC-175</td>
<td>LHS abandon button on Crucible review screen broken</td>
<td></td>
</tr>
<tr>
<td>CRUC-166</td>
<td>Some files not work</td>
<td></td>
</tr>
<tr>
<td>CRUC-152</td>
<td>sysinfo admin screen should show both CRU and FE license string</td>
<td></td>
</tr>
<tr>
<td>CRUC-146</td>
<td>Add review information to ALL search result pages</td>
<td></td>
</tr>
<tr>
<td>CRUC-143</td>
<td>add review-constraints in the EyeQL where clause</td>
<td></td>
</tr>
<tr>
<td>CRUC-140</td>
<td>ensure &quot;linked&quot; reviews are exported thru data in remote api</td>
<td></td>
</tr>
<tr>
<td>CRUC-122</td>
<td>Move review to another project</td>
<td></td>
</tr>
<tr>
<td>CRUC-121</td>
<td>EyeQL return clause: reviews</td>
<td></td>
</tr>
<tr>
<td>CRUC-113</td>
<td>Allow creation of reviews for multiple changesets through /cru/create URL</td>
<td></td>
</tr>
<tr>
<td>CRUC-109</td>
<td>Presented with Post/delete drafts buttons when no comments drafted made</td>
<td></td>
</tr>
<tr>
<td>CRUC-107</td>
<td>Ability to change username</td>
<td></td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th>JIRA Issues (26 issues)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Key</strong></td>
<td><strong>Summary</strong></td>
</tr>
<tr>
<td>CRUC-127</td>
<td>Improperly stopping crucible causes data loss!</td>
</tr>
<tr>
<td>CRUC-274</td>
<td>Create IDE integration with IntelliJ IDEA</td>
</tr>
<tr>
<td>CRUC-273</td>
<td>Create IDE integration with Eclipse</td>
</tr>
<tr>
<td>CRUC-256</td>
<td>Check boxes for checking off files reviewed in a review are lost if navigate to raw text by &quot;download raw text&quot; icon</td>
</tr>
<tr>
<td>CRUC-249</td>
<td>Accept review comments in Japanese (unicode characters)</td>
</tr>
<tr>
<td>CRUC-243</td>
<td>race condition between crucible backup and repository scan</td>
</tr>
<tr>
<td>CRUC-235</td>
<td>File tabs should be kept open</td>
</tr>
<tr>
<td>CRUC-210</td>
<td>Create JIRA issues from code reviews</td>
</tr>
<tr>
<td>CRUC-208</td>
<td>Nullpointer exception when creating new review</td>
</tr>
<tr>
<td>CRUC-202</td>
<td>update ‘Repositories’ screenshot</td>
</tr>
<tr>
<td>CRUC-196</td>
<td>Error on attempting to create a Review.</td>
</tr>
<tr>
<td>CRUC-180</td>
<td>’Show Full Source’ should continue to highlight the diff</td>
</tr>
<tr>
<td>CRUC-177</td>
<td>Comments on reviews are being sent but don't appear in the review when viewed online</td>
</tr>
<tr>
<td>CRUC-165</td>
<td>Folder Issue</td>
</tr>
<tr>
<td>CRUC-139</td>
<td>Can't add new versions of the same file to a review</td>
</tr>
<tr>
<td>CRUC-112</td>
<td>Point online help links to new Crucible doc space</td>
</tr>
<tr>
<td>CRUC-106</td>
<td>Add content to new documentation page ‘Crucible and FishEye’</td>
</tr>
<tr>
<td>CRUC-105</td>
<td>Upload PDF, XML and HTML versions of Crucible docs</td>
</tr>
<tr>
<td>CRUC-103</td>
<td>Create admin page for Projects</td>
</tr>
<tr>
<td>CRUC-100</td>
<td>Move Crucible docs to Confluence</td>
</tr>
<tr>
<td>CRUC-99</td>
<td>Feature request: I would like to see statistics about number of major and minor issues found per developer.</td>
</tr>
<tr>
<td>CRUC-92</td>
<td>Allow Anonymous Access Per Repository</td>
</tr>
<tr>
<td>CRUC-84</td>
<td>Ability to create a review directly from JIRA</td>
</tr>
<tr>
<td>CRUC-50</td>
<td>Allow review of entire file, not a particular diff</td>
</tr>
<tr>
<td>CRUC-28</td>
<td>After Crucible forums are moved, update all links in docs</td>
</tr>
<tr>
<td>CRUC-258</td>
<td>Allow all users to be able to a “Moderator” or “Author” in a review project</td>
</tr>
</tbody>
</table>

**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)

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-207</td>
<td>allow hyphens in Crucible project keys</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-204</td>
<td>Add project to dashboard filter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-195</td>
<td>groups access control to repositories failed after configured External authentication (LDAP)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### From 1.2 to 1.2.1

This is a small bug-fix release.

### JIRA Issues (6 issues)

<table>
<thead>
<tr>
<th>Key</th>
<th>Summary</th>
<th>Priority</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRUC-199</td>
<td>Top of project edit form says Repositories List</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-198</td>
<td>multiple copies of revisions added to review if you add changesets in reverse chronological order</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-182</td>
<td>Projects Don't Persist Through Pages</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-179</td>
<td>Support addition of multiple revisions of the same file that appears in multiple changesets</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-178</td>
<td>Hit NPE when perform backup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRUC-40</td>
<td>Getting wrong changeset number causes diff to be missing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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 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
Issuing a review to the reviewers is known as approving the review.

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.

Without prejudice to 'code inspection', 'peer review' or a myriad of other terms, Crucible uses the phrase code review for simplicity.

A comment is a short textual note that is linked to a review, revision/diff, source line, or to another comment.

The creator is the person who creates the review. In most cases this person will also act as moderator.

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:

<table>
<thead>
<tr>
<th>Permission</th>
<th>Description</th>
<th>Default Assignees</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Edit'</td>
<td>Ability to edit a review's details and change the set of revisions being reviewed.</td>
<td>'Creator' 'Moderator'</td>
</tr>
<tr>
<td>'View'</td>
<td>Ability to view a review. (People without this permission will not know that the review exists.)</td>
<td>Anonymous users</td>
</tr>
<tr>
<td>'Abandon'</td>
<td>Ability to abandon (i.e. cancel) a review.</td>
<td>'Moderator' 'Creator'</td>
</tr>
<tr>
<td>'Re-Open'</td>
<td>Ability to re-open a closed or abandoned review.</td>
<td>'Creator' 'Moderator'</td>
</tr>
<tr>
<td>'Uncomplete'</td>
<td>Ability of a reviewer to change their individual review status from 'Complete' to 'Uncomplete'.</td>
<td>'Reviewer'</td>
</tr>
<tr>
<td>'Reject'</td>
<td>Ability to reject a review submitted for approval (i.e. prevent it from being issued to reviewers).</td>
<td>'Moderator'</td>
</tr>
<tr>
<td>'Complete'</td>
<td>Ability of a reviewer to change their individual review status to 'Complete'.</td>
<td>'Reviewer'</td>
</tr>
<tr>
<td>'Comment'</td>
<td>Ability to add or remove a comment to or from a review.</td>
<td>'Creator' 'Author' 'Reviewer' 'Moderator'</td>
</tr>
<tr>
<td>'Approve'</td>
<td>Ability to approve a review (i.e. issue it to the reviewers).</td>
<td>'Moderator'</td>
</tr>
<tr>
<td>'Submit'</td>
<td>Ability to submit a review for approval (i.e. request that the review be issued to the reviewers).</td>
<td>'Creator' 'Author'</td>
</tr>
<tr>
<td>'Close'</td>
<td>Ability to close a review once it has been summarised.</td>
<td>'Moderator'</td>
</tr>
<tr>
<td>'Delete'</td>
<td>Ability to delete a review.</td>
<td>'Creator' 'Moderator'</td>
</tr>
<tr>
<td>'Summarise'</td>
<td>Ability to summarise a review. (Normally this would be done after all reviewers have completed their review.)</td>
<td>'Moderator'</td>
</tr>
<tr>
<td>'Create'</td>
<td>Ability to create a review.</td>
<td>All logged-in users</td>
</tr>
<tr>
<td>'Recover'</td>
<td>Ability to resurrect an abandoned (i.e. cancelled) review.</td>
<td>'Creator' 'Moderator'</td>
</tr>
</tbody>
</table>
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:
<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Draft</td>
<td>See Creating a Review.</td>
</tr>
<tr>
<td>Require Approval</td>
<td>Relevant only when the moderator is not the creator. See Issuing a Review.</td>
</tr>
<tr>
<td>Under Review</td>
<td>See Issuing a Review and Reviewing the Code.</td>
</tr>
<tr>
<td>Summarize</td>
<td>See Summarising and Closing the Review.</td>
</tr>
<tr>
<td>Closed</td>
<td>See Summarising and Closing the Review.</td>
</tr>
</tbody>
</table>

- 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:

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abandoned</td>
<td>This happens when a review is deleted.</td>
</tr>
<tr>
<td>Rejected</td>
<td>Any reviews that a moderator has rejected.</td>
</tr>
</tbody>
</table>

**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.