IBM Rational Build Forge Evaluation Guide: Part 1. Integrating Build Forge with ClearCase and ClearQuest for unified change and release management

A hosted trial with hands-on experience

Skill Level: Intermediate

Paul Boustany (pboustan@us.ibm.com)
Marketing Engineer
IBM

Tim Feeney (tfeeney@us.ibm.com)
Change and Release Management Practice Lead
IBM Corp.

29 Mar 2007

This evaluation guide takes you through step-by-step instructions designed to give you an overview of some of the features that differentiate IBM® change and release management software. As you progress through the lab-style exercises in the scenario, the hands-on experience you get shows you how IBM® Rational® ClearQuest®, Rational® ClearCase®, and Rational® Build Forge® are tightly integrated to provide development teams with a comprehensive, auditable solution.

Section 1. Before you start

Learn what to expect from this tutorial and how to get the most out of it.

About this series

This series guides you through the core features of IBM® Rational® Build Forge® in a preconfigured environment. By the end of Part 2, you should feel comfortable with the Build Forge server and the integration with the Rational change and release management solution.

About this tutorial

In this evaluation guide, you will learn about the build and release capabilities of Rational Build Forge while logging and correcting a defect found in a typical Web-based application. Each section takes you, step by step, through the stages of this scenario:

- As the Tester, you will confirm that the error exists and generate a defect by using Rational ClearQuest.
- Then you'll transition to the Project Manager role, review the defect in ClearQuest Web, and assign it to a developer.
- As the Software Developer, you will make the necessary code change which is under version control in Rational ClearCase. You will then verify that the change works in your Eclipse development environment.
- Next, as the Build Engineer, you will build the corrected application by using either the Rational Build Forge Eclipse plug-in or the Rational Build Forge Web-based Management Console.
- Lastly, you will act in the role of the Project Manager again and then as the Deployment Engineer who uses Rational Build Forge to deploy the application.

Objectives

Through this scenario, you will see how IBM® Rational® ClearQuest®, Rational® ClearCase®, and Rational Build Forge are tightly integrated to give your development team a comprehensive, auditable change and release management solution. You will also see how Rational Build Forge easily gives you controlled access to production build environments for rapid error detection and troubleshooting from within your IDE.

Prerequisites

This hosted trial provides an environment preconfigured with Rational ClearCase

7.0, Rational ClearQuest 7.0, and Rational Build Forge 7.0. Some knowledge of Rational change and release management products may be helpful, but that is not required.

System requirements

This online trial program uses the Citrix® Access Gateway™ platform to provide a connection from your workstation to a remote server running the IBM software. Therefore, you need to download Citrix client software before starting this evaluation. You can download the Citrix MetaFrame® Presentation Server™ client at no charge from the Citrix site; many versions, including versions for Microsoft® Windows® and Linux® platforms, are available. After you install the client, you will be asked to restart your browser. If you do not have the Citrix client installed when you attempt to access the online trial, you will be prompted to install the client.

Section 2. Which application handles what function

The purpose of this evaluation guide is to give you an overview of IBM change and release management lifecycle products. During the exercises that you complete here, you will work with the combination of IBM Rational ClearCase, ClearQuest, and Build Forge to correct a defect in a sample Java application, and then deploy your changes to a Web server.

ClearCase software configuration management products enable you to effectively manage and control software assets to help you deliver better applications faster and at lower cost.

ClearQuest change management products help you manage the software lifecycle more effectively with automated workflow control and to enforce development processes, from requirement definitions through production.

Build Forge products help you automate complex processes and integrate diverse toolsets to compress development cycles, improve product quality and increase staff productivity.

> **Eclipse option**
> For this evaluation, we are presenting all three applications in a single Eclipse-based interface. However, in your own environment, you can use and integrate these applications either in or out of Eclipse.

Together, these tools accelerate software and systems delivery, make global teams more efficient, and govern end-to-end software processes. They improve visibility and control of software projects and enable fast delivery of high-quality software. By using these tools, you also increase team efficiency through centralized build and release management, accelerate build and release cycles for faster software delivery, and improve software quality through consistent build and release processes.

Section 3. Modifying the application

A. Test the RationalBank Web application

In this subsection, you will be fixing a problem with the RatlBankWeb application. We will illustrate this problem first by executing the deployed application.

1. Navigate to the Java™ 2 Platform, Enterprise Edition (J2EE™) perspective by clicking the J2EE icon in the upper-right corner of the window (Figure 1).

Figure 1. Navigating to the J2EE perspective

2. Click the **Web icon** to open the Web browser (Figure 2).

Figure 2. Opening the Web browser

3. Enter RatlBank URL: `http://localhost:9090/RatlBankWeb/`.
4. Log in to the application (Figure 3) by entering these parameters:
   - **IBM ID:** `101`
   - **Password:** `101`

Figure 3. RationalBank login screen

5. Click **Submit**.
6. Select the account number **101-1001**, and then click **Submit**.
7. Select the **List Logged transactions** action, and then click **Submit** again. A screen similar to the one in Figure 4 should display.

Figure 4. Account transactions screen
Notice that the transaction date has obviously not been formatted correctly. Therefore, you will use Rational ClearQuest create a defect record to be sure that your team addresses this problem.

B. Create a defect report in Rational ClearQuest

Now, log into ClearQuest as **Dana, the Tester**, to submit a defect for the transaction date display issue that you found in the application.

1. Navigate to the **Rational ClearQuest** perspective (Figure 5).

Figure 5. Navigating to Rational ClearQuest

2. Select **ClearQuest > Database > Connect > dana,7.0.0@RBU** to log in as Dana. (See Figure 6.)

Figure 6. Logging in as the tester
If you get a password prompt:
This evaluation was preconfigured to remember Dana's password. However, if you are prompted for a password, enter: `dana`. Then select the option to **Remember Password**. (See Figure 7.)

Figure 7. Entering Dana's password in the Connect screen

3. Select **ClearQuest > New > Defect** to create a new defect report (Figure 8).

Figure 8. Creating a new defect report
Tip:
You can also use the highlighted icon in the toolbar (Figure 9) to submit a new defect record.

Figure 9. Icon in the toolbar to create a defect report

4. The input entry form for a new defect record will display. Enter the following parameters:
   - **Headline:** `Transaction date is displayed incorrectly`
   - **Severity:** `2 major`

Tip: Mandatory fields
The red labels for these fields indicate that they are mandatory fields, which must be filled in before ClearQuest will save the record. This is just one of the many ways that ClearQuest allows you to build in automated support for your change

management process.

5. Click **OK** to submit the defect report.

6. Log out of the Rational ClearQuest client by clicking **ClearQuest > Database > Disconnect > dana**. (See Figure 10.)

Figure 10. Logging out as Dana

C. Assign the defect to a developer to handle

As the **Project Manager, Pat**, you need to get someone to work on this new defect. You decide to assign it to **Alex, a Software Developer**. To do that, you need to follow these steps:

1. Navigate back to the web browser in the J2EE perspective and enter the URL for the Rational ClearQuest web server http://localhost/cqweb/login

2. Log in as the Project Manager Pat by entering the following parameters:
   - **User Name:** `pat`.
   - **Password:** `pat`
   - **Schema Repository:** `7.0.0`
   - **Database:** `RBU`

4. Click the **Login** button. Pat's ClearQuest Web account is configured to run the **Submitted High-Severity Defects** query upon login.

5. Review the query results, and then, while still in the **Results** set display, click the check box for the number of the defect that you created in the previous step (Figure 11).

Figure 11. Results set display

5. Now click the **Change State** icon (Figure 12).

Figure 12. Changing the state

6. Select **Assign**, and enter these details:
   - **Priority:** `1-Resolve Immediately`
   - **Owner:** `alex`

8. Click **Save** to complete assignment of the defect resolution.

9. Click **View** to verify the changes made to the defect record.

10. Log out of ClearQuest by selecting **Logout** in the upper-right corner.

Tip:
You can configure ClearQuest to use e-mail notifications to keep team members informed of changes or updates. This can also improve response time. In this example, assigning the defect to Alex automatically e-mails him to make him aware of this high-priority assignment.

D. Work on the defect

You will now log into ClearQuest as **Alex**, the developer, to review your assigned defects and to begin work on the defect to fix the transaction date display.

1. Switch to the **ClearQuest** perspective (Figure 13).

Figure 13. Switching to the ClearQuest perspective

2. Select **ClearQuest > Database > Connect > alex,7.0.0@RBU** to open the ClearQuest database and log in as Alex (Figure 14).

Figure 14. Logging in as Alex

3. Enter Alex's password (Figure 15). As in the previous task, this evaluation has been preconfigured to remember Alex's password. However, if you are prompted for a password, just enter: `alex`. Also, select the **Remember Password** option.

Figure 15. Entering Alex's password

4. A query should run automatically to display Alex's To-Do list (Figure 16).

Figure 16: To-Do list for Alex

5. Right-click the defect record that you created earlier, and select **Utilities > WorkOn**. This will change the defect to the **Opened** state, and the display will indicate to the rest of the team that Alex is now actively working on this defect (Figure 17).

Figure 17: Beginning work on the defect item from Alex's To-Do list

6. You will be prompted to select which ClearCase view to do the work in. Select **alex_RatlBankWeb_intg** (Figure 18), and then click on **OK**.

Figure 18: Selecting the ClearCase view to work in

E. Connect to the ClearCase Web server

> **ClearCase Remote Client for Eclipse**
> Before you can execute any Rational ClearCase operations in Eclipse using the ClearCase Remote Client, you must connect to the Rational ClearCase Web server. There is a specific Rational ClearCase perspective to enable you to do this and to carry out many other Remote Client operations, such as loading views and creating baselines.

1. Switch to the **ClearCase** perspective. The **ClearCase Navigator** and **ClearCase Details** views will display. This indicates that a Rational ClearCase view has already been created (Figure 19), so you can re-use that one,

Figure 19: ClearCase view to use

2. To connect to the Rational ClearCase Web server, click the **alex_RatlBankWeb_Intg** view in the **ClearCase Navigator**, and then select **ClearCase > Connect** from the menu to open the login window (Figure 19a).

Figure 19a. Logging in to the ClearCase Web server

3. Enter `alex` as the password and click **OK**.

4. While you have been disconnected from the server, changes might have been made in the integration stream, so it's time to update your view:

   A. Right-click the **alex_RatlBankWeb_Intg** view in the **ClearCase Navigator** and select **Tools > Update Resource**.

   B. When the **Update Resources** window displays, click **Apply** to update the view. A list of updated elements (if applicable) will be displayed in **ClearCase View Configuration**.

6. Switch to the J2EE perspective (Figure 20). You might have to click on the **Open Perspective** icon and select **Other** to find the J2EE perspective.

Figure 20. Switching to the J2EE perspective

> **J2EE perspective in Eclipse**
> The Eclipse Web Tools Platform (WTP) project includes a special perspective for editing and running Web applications called the J2EE perspective. You will use this perspective for making the change. In this case, you know that the listTransactions JSP file contains the code for displaying the transaction date. The error is within that JSP.

6. In the **Project Explorer**, open the WebContent folder (**RatlBankWeb > WebContent**). You should see the source code for the RatlBankWeb application, including the listTransactions.jsp file that you will need to edit to fix the transaction defect (Figure 21).

Figure 21. Contents of the WebContent folder

7. Check out the **listTransactions.jsp** file by right-clicking on it and selecting **Team > Check Out**, as you see in Figure 22. (The **Checkout Resources** window will display. You should see that the **Activity** is defaulted to the defect record that you created earlier. Your most recent **WorkOn** action indicated that this activity would be worked on in the view for this workspace.)

Figure 22. Starting to check out the Transactions JSP file

8. Click **Apply** to complete checking out the file.

F. Make the change

Tip:
The integration between Rational ClearQuest and Rational ClearCase keeps track of all changes made as a result of working on the defect.

1. Double-click on the **listTransactions.jsp** file to open it for editing.

2. The cause of the defect is located on line 251. Scroll right, and then edit the line by changing:
   `transaction.getTimeStamp()`
   to
   `transaction.getTimeStamp().getTime()` , as Figure 23 shows.

> **Quick navigation tip**
> A quick way of navigating to a specific line in Eclipse is to hold down Ctrl and L together (Cntl + L). When the **Go To Line** dialog displays,

enter 251, and then click on **OK** to go to that line.

.

Figure 23. Changing the code in the Transactions JSP file to correct the defect

3. Select **File > Save** from the drop-down menu to save your changes.

G. Test the change

You can run the application inside of Eclipse to test it:

1. Right-click on the **RatlBankWeb project** in the **Project Navigator**, and select **Run As > Run on Server** (Figure 24).

Figure 24. Selections to run the file on the server

2. The **Run on Server** window will display and prompt you to click **Finish**. After a short time, a Web browser showing the application should open (Figure 25).

Figure 25. Application display in the Web browser

3. Double-click **RationalBank: Transactions** in the tab header to maximize the display.

4. Log in, and then test the Transaction page again by using the procedure described in **Testing the RationalBank Web application**, starting with Step A-1 in this section ("Modify the transaction"). This time, however, you should see that the transaction date has been formatted correctly, as Figure 26 shows.

Figure 26. Test results showing that the transaction date now displays correctly

5. Close the **RationalBank: Transactions** tab by clicking the **X** on the tab header.

6. Check in the listTransactions.jsp file. You have fixed and tested it, so it is ready to be incorporated into your integration build.

   A. Right-click the **listTransactions.jsp** file, and select **Team > Check In**.

   B. When the **Checkin Resources** window displays, click **Apply** to

check in the file.

Section 4. Building the application

> **Alternate build option**
> At this point, you can either continue following this article to build the application from the Eclipse plug-in or jump to the Alternate build option section to learn how to build it from the Rational Build Forge Self-Service plug-in. You do not need to go through both methods.

In this section, as **Jan, the Build Engineer**, you will execute a formal build of the RatlBankWeb application. After the build completes, you will be able to view the results of the build and observe the build record created in Rational ClearQuest. With a successful build ready, you will create a deployment record in Rational ClearQuest to request that the application be deployed to the system test environment.

Through this scenario, you will see how Rational Build Forge is integrated with Rational ClearQuest and Rational ClearCase to manage the build of applications and capture important details about the build that support your compliance and governance processes. You'll also see how you can automate your build workflow by using these tools.

A. Build the application by using Rational Build Forge

1. Log in to Rational Build Forge. The Build Engineer works in Rational Build Forge's easy-to-use, browser-based GUI to perform a build.

2. Navigate to the **J2EE perspective**, and click in the **Web navigation bar**. Then enter the Rational Build Forge server URL (see Figure 27): `http://localhost:82/fullcontrol`.

Figure 27. Entering the Rational Build Forge server URL

Tip:
The Rational Build Forge **Management Console** is a Web interface that is ideal for distributed teams, because it enables secure and controlled access to the system anytime, anywhere. It gives team members a real-time view of all activities, including fine-grained, role-based control to pause, cancel, and resume processes on multiple production machines.

3. Log in as Jan, the Build Engineer, using these parameters, and then click **Login**.
   - **Username:** `jan`.
   - **Password:** `jan`.

> **Rational Build Forge permissions**
> As the Build Engineer, Jan has privileges above and beyond those that the developer has, such as adding, modifying and deleting objects associated with or in support of a build project (users, projects, and environment variables, for example). A developer typically has sufficient privileges to execute all of or a subset of a build project.
>
> Note that some of these steps are accessible by developers and some are accessible only to the Build Engineer (usually Rational ClearCase baselining and staging operations). Also note that some of these steps have been specified as threaded so that they will execute in parallel.

B. Examine the project environment and build steps

1. Click **Environments** in the left frame of the window, and then click **RatlBankWeb Int**. This shows all of the environment variables that have been defined for building the RatlBankWeb project. These can be a combination of external (operating system) environment variables and internal (Rational Build Forge) environment variables.
2. Click **Projects** in the left frame, and then click **RatlBankWeb Int** to show the steps that have been defined for building the RatlBankWeb project (Figure 28). Click some of the steps to further explore the project.

Figure 28. Examining the build steps

C. Execute the RatlBankWeb_Int build

Execute the build directly from inside of the Rational Build Forge Management Console.

1. Click **Start Project**.

Tip:
The specific values of the environment variables for this execution are displayed

under the **Run Details** tab, and the steps to be run will be displayed under the **Steps** tab. You can override any of these at this point.

2. Click **Execute**, and the build will start (Figure 29).

Figure 29. State shows that the build is running

3. When the state of the execution transitions from Waiting to **Running**, click the build, in this instance **48_INT**, to display the steps as they run (sooner is fine, too, but there is nothing to display until the project is running).

Note: The build number may be different at the time you are reading this tutorial. Please substitute 48_INT with your build number whenever the build number is referenced in this guide.

Tip:
You can click any step name (Figure 30) to drill down into any of the steps to display more detail about what Build Forge is doing.

Figure 30. Step names display

CAUTION
Do not select a refresh rate of less than 10 seconds, because the GUI will not have time to respond to one refresh request before it needs to process the next refresh action.

4. Click the **Refresh** list box, and select a refresh rate of **10 seconds** (Figure 31).

5. Observe the progression through the build project.

Figure 31. Setting the refresh rate

6. The steps for the project run span multiple display pages. You can click the single arrow (**>**) to go forward one page or on the double arrows (**>>**) to go to the end, or the last page in the project run. Conversely, use **<** or **<<**, respectively, to go back one page or back to the beginning. (See Figure 32.)

Figure 32. Arrows for viewing different pages of the project run

7. While viewing **Page 2 of 2** of the project run, wait until the project run status shows **Completed** before moving on to the next step (Figure 33).

Figure 33. Checking project run status

8. After the build has completed in the previous step, click the **Create Deployment Baseline** step of the project run.

9. On the **Step Log** tab, select the **Display Flattened Log in New Window** link (Figure 34).

Figure 34. Remember to click both places

Save this information!
The build project created a deployment baseline to use when deploying the revised application. You will need this baseline label information in the deployment section of this evaluation.

.

10. A new browser window will display showing the complete Step Log for the Create Deployment Baseline step. Scroll down to the end of the **Step Log** to see the baseline created for the `RATLBANKWEB_REL` component (Figure 35).

Figure 35. Save this information for later

D. Examine the Bill of Materials

Rational Build Forge generates a Bill of Materials (BOM) after each project run. The BOM contains information about the steps in the run and the changes to files that resulted from the build. You can provide the BOM to people who need to know about the project run, such as your quality assurance department, to help them understand the contents of a new build. The BOM can also serve as an audit resource for your build and release process.

1. Click **Bill of Materials** to display the Bill of Materials (Figure 36). It shows the steps that were executed, as well as the Rational ClearCase source code changes that were included in the build and the Rational ClearQuest defects that were resolved.

Figure 36. Viewing the Bill of Materials

2. Expand **Build Steps** to display the steps that were executed and their status (Figure 37).

Figure 37. Checking the status of executed steps

3. Expand **Source Changes** to display the versions of Rational ClearCase source-controlled elements that were included in the build (Figure 38).

Also note that the BOM includes the differences between the version used in the build and its previous version.

Figure 38. Checking the elements included in the build

4. This BOM also contains information about the Rational ClearQuest defects tested by this build, which you can see by expanding **Rational ClearQuest Defect Information** (Figure 39).

5. Click **Logout** in the upper-right corner of the **Rational Build Forge Management Console**.

Figure 39. Display of any defects tested in the build

E. Examine the defect

With the integration of Rational Build Forge and Rational ClearQuest, defect records can be populated with build results and automatically transitioned to an appropriate state in your change management workflow.

1. Navigate back to the **Rational ClearQuest** perspective (Figure 40).

Figure 40. Returning to the ClearQuest perspective

2. Select **File > Database > Connect > jan,7.0.0@RBU** to login as the Build Engineer, Jan.

Entering Jan's password
This evaluation was preconfigured to remember Jan's password. However, if you are prompted for a password, enter `jan`, and then select the option to **Remember Password** (Figure 41).

Figure 41. Enter Jan's password if prompted

3. Find the defect that you created earlier by double-clicking **Public Queries > All Defects**.

4. Scroll to the bottom of the list to find the defect that you submitted in the previous step, and then double-click that **defect record**.

5. In the **View Defect** window (Figure 42), notice that this record has been resolved automatically for you by Rational Build Forge (current state shows Resolved).

6. Click the **Notes** tab to display the build where it was resolved.

Figure 42. The View Defect display

7. Click the **Unified Change Management** tab (Figure 43) to see information about where the work on this defect took place in Rational ClearCase.

Figure 43. The Unified Change Management tab

Unified Change Management benefits
Applying a Unified Change Management (UCM)) process enabled by Rational ClearCase and Rational ClearQuest allows all changes to all files and directories to be tracked when you are working on a particular change request, such as a defect record or fix. This **change set** is tracked by the integration and is available for review from the **UCM** tab of a Rational ClearQuest record.

.

8. Click **View Change Set**.

9. In the **Properties** window for the Rational ClearCase activity (the defect that you are reviewing), note that the listTransactions.jsp file is listed as being changed as a result of working on this defect.

10. For more information, right-click the **listTransactions.jsp** file and select **Version Tree** (Figure 44). Also notice the options to perform other Rational ClearCase operations on the file, such as show **History** and **Compare with Previous Version**.

Figure 44. Viewing the Version Tree display

11. Examine the **version tree** for the listTransactions.jsp file (Figure 45). Note that, for this version of the file, both the activity and the new baseline (or baselines) that were automatically created by Rational Build Forge have been recorded.

Figure 45. Examining the activity record

F. Display the build record

You can also display the details of the Build Record that was automatically created in Rational ClearQuest by Rational Build Forge.

1. To navigate to the **build record**, double-click **Public Queries**, and then click the **All Builds** query (Figure 46).

Figure 46. All Builds query results

2. Double-click the build record associated with your build (**48_INT**), as Figure 47 shows, to display the Build Record (Figure 48).

Figure 47. Selecting the Build Record to view

3. Click the **Build Details** tab for the build record associated with this build (**48_INT**). Here, you will see the **Build Log**, which shows the steps executed and their Pass or Fail results, plus a **Build Web URL** hyperlink to the project run in Rational Build Forge.

Figure 48. Display of the build record

4. Double-click the **Build Web URL** field to see the project run information in the Rational Build Forge Web GUI.

5. Log in as Jan, the Build Engineer, by entering the following parameters and then clicking **Login**.

   - **Username:** `jan`
   - **Password:** `jan`

7. Then close the browser window. After you have logged in, you will see the project run data associated with this Build Record (Figure 49).

Figure 49. Project run data display

7. Click **Logout** in the upper-right corner of the **Rational Build Forge Management Console** (Figure 50).

Figure 50. Logging out of the console

8. Click **Cancel** to close the **Build Record** window.

G. Create a deployment record

Rational ClearQuest supports a deployment approval process through creation of records of type **DTDeployment**. You associate each deployment record that you create in Rational ClearQuest with a release record. You then use Rational ClearQuest to transition the deployment record through the ordered testing environments in your release's deployment process. Of course, you must get approval from the appropriate people in relevant roles before the deployment record can move from its current environment into the next environment in the series.

When creating a DTDeployment Record, you associate a deployment unit based on the results of your most recent build. The deployment unit lists the artifacts to be

deployed.

> **About deployment units**
> A deployment unit defines the list of artifacts to be deployed to an environment. Deployment units are both created and managed by using the Rational ClearCase command-line utility called the **XML Decorator**, or **du_tool.pl**.

1. Back in the **Rational ClearQuest Perspective**, select **ClearQuest > New > DTDeployment** (Figure 51).

Figure 51. Starting a new DT deployment

2. Enter these parameters:
   - **Headline:** `Deploy RatlBankWeb v2 Build`
   - **Release:** `RatlBank_2.0`

3. Click the **Browse** button next to the **Deployment Unit Reference** field, and then navigate to **C:\Builds\RatlBankWeb_48_INT\RatlBankReleases\web** and select **du_decorated_48_INT.xml** (Figure 52).

Note:
The `RatlBankWeb_48_INT` number will be different, because it is based on your build ID.

Figure 52. Selecting the XML file
Note:
If your build tag was different, you will need to browse to the folder with that build tag in the name and select the corresponding XML file.

4. Click **Open**.

5. For **Owner**, enter `dana`.

6. You will see two deployment options in the **Deploy Type** field (Figure 53). Select **BuildForge**.

Figure 53. Selecting the deployment type

7. Your deployment record should look similar to what Figure 54 shows.

Save this information!
Make note of the record ID (**RBU00000256**), and save it with the baseline label name that you saved previously, because you will need this when you deploy the application.

.

Figure 54. Deployment Record display

8. Click **OK**.

G. Associate the build record with the deployment record

1. Double-click the **Recently Submitted > DTDeployment(1)** query.

Figure 55. Selecting the DTDeployment(1) query

2. In the query results, double-click the **deployment record** that you created in the previous step.

Figure 56. Selecting the deployment record

3. Then click the **Modify** button.

Figure 58. Selecting the Modify icon
Modifying an existing ClearQuest record

To modify the **Release** record, use the highlighted icon in the toolbar.

Figure 59. Toolbar icon for modifying an existing ClearQuest record

4. Click the **Build** tab in the displayed record form.

5. Click the **Add** button.

6. In the **Browse Record Type** window, click **Browse**. This will display the directory tree view of all personal and public queries available.

7. Select **Public Queries > All Builds**, and then click **OK**.

Figure 60. Selecting the All Builds query

8. The first build record in the list should be for your recent build. Select **that record**, and then click **OK**.

9. Click **OK** to save the record modifications.

10. Then click **Disconnect from Rational ClearQuest**.

Section 5. Alternate build option

The IBM Rational Build Forge Self-Service plug-in extends build management capabilities to your desktop by leveraging integrated development environment (IDE) technology. Access to third-party vendor applications within IDEs is typically offered through plug-ins. A **plug-in** is a lightweight set of code that gives you seamless access to one or more external applications exposed as menus, buttons, and so forth, directly within the IDE graphical interface.

The Rational Build Forge Self-Service plug-in allows you to view and build Build Forge projects within a controlled environment from directly within your IDE. Capabilities are limited to viewing build results, initiating new builds, and viewing the status of currently running builds. You cannot create new projects, delete projects, or alter existing project step data. System-level permissions and access to Rational Build Forge projects are honored; therefore, you can view and run only projects for which you have access permissions. For this evaluation, you have been granted unlimited access.

Running the build with the self-service plug-in

1. Switch to the **Rational Build Forge** perspective (Figure 71).

Figure 71. Switching to the Build Forge view

2. Connect to the Rational Build Forge server by right-clicking **BF Mgmt Console** and selecting **Refresh** (Figure 72).

Figure 72. Connecting to the server and refreshing the view

3. Expand the **Rational Build Forge Management Console** to view the list of projects visible to **Alex**. Right-click **RatlBankWeb Int** and select **Execute this build** to run the integration build project (Figure 73).

Figure 73. Expanded console view showing Alex's projects

4. A new build should display in the **Build Info** view (Figure 74).

Figure 74. Build Info view
Note:
Depending on the state of the evaluation, the actual build tag may vary.

5. Click the build entry in the **Build Info** view to see the **Build Log** for the running build.

6. In this version of the Build Forge Eclipse plug in, the Build Log is not refreshed automatically. Reselect the build in the **Build Info** view several times to refresh and update the Build Log. The log displays the steps that are being executed as part of the build project that you are running. The step currently being executed has the black and blue arrows next to it. The screen capture in Figure 75 shows the **Update Build View**.

Figure 75. Build Log view during the build
Note:
This step may take several minutes to complete.

7. After several minutes, all steps should have completed. When a step is finished, a green box shows in the upper-right corner of the icon preceding the step (Figure 76). (Boxes that are grayed out are not executable by the developer, based on the current project security settings.) When all steps have completed successfully, proceed to the next step.

Figure 76. Build Log view when steps are finished
Note: You (or the developer) would then re-test the application to confirm that you did not break the build. For the sake of time, we will skip this step, and assume that it still works.

8. Expand one or more of the build steps to see their results.

Section 6. Deploying the application

In this section:

- You perform the role of **Project Manager**, Pat, and approve the deployment requested previously by the Build Engineer.
- Then, as the **Deployment Engineer**, you will review the deployment request and see that it is for a successful build of the RatlBankWeb application. When you are ready to deploy the application, you will use Rational ClearQuest to initiate a deployment that uses Build Forge to distribute the application.
- After observing that the deployment was successfully completed, you will transition back to the role of **Tester** and confirm that the deployed application does, indeed, fix the defect.

Through this scenario, you will observe how Rational ClearQuest tracks deployments of applications between environments and how those deployments can require approval to support your governance and compliance processes. Through the integration between Rational Build Forge and Rational ClearCase, the correct application files will be extracted from Rational ClearCase and deployed by a Build Forge project to the appropriate server.

A. Log in to ClearQuest as the Project Manager

1. Navigate back to the Web browser in the **J2EE perspective**, and enter the URL for the Rational ClearQuest Web server: `http://localhost/cqweb/login`.
2. Log in as the Project Manager, Pat, using the following parameters:
   - **User Name:** `pat`
   - **Password:** `pat`
   - **Schema Repository:** `7.0.0`
   - **Database:** `RBU`
4. Click **Login**.

B. Approve the deployment record

The RatlBankWeb governance process stipulates that deployments to the System Test environment require a Project Manager's approval.

Tip:
The **Deployment Tracking** package in Rational ClearQuest allows deployment

workflows to be customized to require the approval of a person in a particular role for a given release to a given environment. By using the ClearQuest **eSignature** feature, you can specify that these approvals require an electronic signature, as well.

1. Log into Rational Build Forge.
2. Expand the **Public Queries** to the **DeploymentQueries** folder and select **My Approvals** (Figure 61).

Figure 61. Selecting My Approvals

3. In the **Results** set, in the number icon (#) column, click the **icon for the row** for the Approval Record associated with the Deployment Record that you created in the previous exercise (Figure 62).

Figure 62. Selecting the approval record

4. Click **Change State**, and then select **Approve** (Figure 63).

Figure 63. Changing the state to Approved

5. *Optionally*, enter a comment.
6. Note that the eSignature tab is still red. To conform to your governance process, we are requiring that the approver provide an additional sign-off. Click the **eSignature** tab, and enter `pat` for both the username and the password.
7. Click **Save** to approve the deployment.
8. Log out of Rational ClearQuest by selecting **Logout** in the upper-right corner of the screen.

C. Log into Build Forge

1. Navigate to the **J2EE perspective** and click in the **Web navigation bar**.
2. Enter the Build Forge server URL: `http://localhost:82/fullcontrol`.
3. Log in as Dana, who is also the Deployment Engineer, using the following parameters:
   - **User Name:** dana

- **Password:** dana

D. Deploy the project

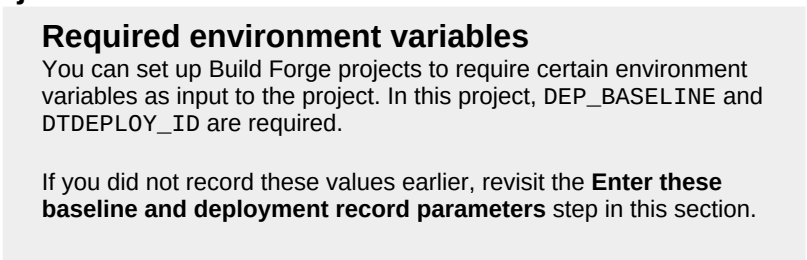
In this step, the Deployment Engineer is deploying the application by using a preconfigured Ant task run from a Build Forge project.

Tip:
Rational Build Forge is highly versatile and can be used to manage more than just traditional compile or build stage of an application. If you can execute it from the command line, you can run it from Build Forge.

1. On the left side of the console, click the **Projects** link.
2. Click the **RatlBankWeb Dep** project.
3. Take a look at the steps used to deploy the **RatlBankWeb** application.
4. Click **Start Project**.
5. Enter these baseline and deployment record parameters (entries are case sensitive), as shown in Figure 64:
   - `RATLBANKWEB_REL_48_INT_deploy` as the label for the **DEP_BASELINE** environment variable. The baseline label is used to create a view with the correct RatlBankWeb WAR file to deploy.
   - `RBU00000256` for the **DTDEPLOY_ID**. The Deployment Record ID is necessary to advance the state of the deployment record.
7. Save the information that you just entered, because you need it later.

Important:
If the baseline and deployment record created previously were different, use those values instead.

Figure 64. Project Environment Variables screen



7. Click **Execute** to start the deployment.

8. When the state of the execution changes from Waiting to **Running**, as Figure 65 shows, click the Build Tag (**44_DEP**) to display the steps as they run (see Figure 66).

Figure 65. Deployment state has changed to Running
Figure 66. Build Tag screen showing the steps as they run
Tip:
You can drill down into any of the steps to see more detail about what Build Forge is doing.

9. Click the **Refresh** drop-down menu, and select a refresh rate of **10 seconds** (Figure 67). Observe the progression through the build project.

Figure 67. Selecting the refresh rate

10. After the project has completed, click the **Deploy Application** step in the **Project Run** (Figure 68).

Figure 68. Selecting Deploy Application

11. Scroll down in the **Step Log** to observe that the application has been deployed successfully (Figure 69).

Figure 69. Checking results in the Step Log

12. **Log out** of the Rational Build Forge Management Console.

E. Review the deployment record

1. Navigate to the **Rational ClearQuest** perspective.

2. Select **ClearQuest > Database > Connect > dana,7.0.0@RBU** to log in as the deployer, Dana.

3. Expand **Public Queries > DeploymentQueries** and double-click the **MyDeployments** query.

4. Select **RatlBank_2.0** for the **Release Name** in the **Dynamic Filters** dialog, and then select **OK**.

5. In the **Results** set, click the entry for the record representing the

deployment that you just completed.

6. Observe that the **State** of the record has changed from Ready to **Deployed**.

F. Test the deployed application

Now that you have deployed the new release of RatlBankWeb to the system test environment, you can test it to find out whether the defect has been resolved properly.

1. Navigate back to the Web browser in the **J2EE perspective**, and enter the URL from the RaltBank application: `http://localhost:9090/RatlBankWeb/`
2. Log in to application by entering the following parameters:
   - **IBM ID:** `101`
   - **Password:** `101`
4. Click **Submit**.
5. Select the account number **101-1001**, and then click **Submit**.
6. Select **List Logged transactions**, and click **Submit**. A screen similar to what you see in Figure 71 should display. Notice that the transaction date is now formatted correctly, which confirms that the defect is resolved.

Figure 70. Verifying that the defect is now resolved

G. Close the defect

The final step in this evaluation is to close the defect by performing **Validate action** on it, thus indicating it has been verified as corrected.

1. Navigate back to the **Rational ClearQuest** perspective
2. If you are not already connected, select **ClearQuest > Database > Connect > dana,7.0.0@RBU** to log in as the Tester, Dana.
3. Find the defect that you created earlier by double-clicking **Public Queries > All Defects**.
4. Scroll to the bottom of the list to find the defect report that you submitted

previously, and double-click that **defect record**.

5. Select the **triangle** next to the **Change State** button, and then select **Validate** to close the defect.

6. Click **OK**.

Section 7. What's next

As you have seen in Part 1 of this two-part evaluation guide, IBM Rational change and release management products help software and systems development teams empower businesses by accelerating software delivery, making global teams more efficient, and governing end-to-end software processes.

In Part 2, you will get an in-depth look at creating and executing projects, user administration, scheduling, multiple process threading and more. By the end of the evaluation, you will see the value that the core features of Build Forge can bring to your team.

IBM provides the most flexible and scalable change and release management solution, enabling global teams to stay productive and deliver projects on time. Working with the leader in change and release management products means that you have a reliable, proven solution.

Resources

Learn

- Get more acquainted with Build Forge by reading IBM Rational Build Forge Evaluation Guide, Part 2.
- ClearQuest users and administrators can find more resources in the ClearQuest area of the developerWorks Rational zone, including ClearQuest hooks, Eclipse plug-ins, scripts and triggers, product documentation, articles and whitepapers, with links to training courses, discussion forums, and support.
- ClearQuest and ClearCase training: SCM215, Essentials of IBM Rational ClearCase UCM for Developers This online course introduces you to the concepts of Unified Change Management. You learn to use ClearCase UCM and ClearQuest to perform common, day-to-day software development tasks.
- ClearQuest training: SCM010, Getting Started with IBM Rational ClearQuest If you are new to IBM Rational ClearQuest, this Short Learning Module is for you. The Web-based training Rational ClearQuest SLM emphasizes the practical uses of Rational ClearQuest for both the Web and Windows and introduces you to operational basics of using Rational's defect-tracking tool.
- ClearQuest training: SCM010, Getting Started with IBM Rational ClearQuest If you are new to IBM Rational ClearQuest, this Short Learning Module is for you. The Web-based training Rational ClearQuest SLM emphasizes the practical uses of Rational ClearQuest for both the Web and Windows and introduces you to operational basics of using Rational's defect-tracking tool.
- ClearQuest training: SCM110, Principles of Defect and Change Tracking with IBM Rational ClearQuest Take this online introduction to defect and change request tracking to learn how to use IBM Rational ClearQuest to submit change requests, track requests from submission to resolution, and generate charts and reports for change and defect tracking results.
- ClearQuest training: SCM205, Essentials of IBM Rational ClearQuest This course introduces students to using IBM Rational ClearQuest to track change requests. It covers the basic concepts involved in change management and how ClearQuest facilitates creating change records, controlling the change management process, and communicating status to project stakeholders by using queries, reports, and charts.
- ClearCase training: SCM101, Principles of Software Configuration Management with IBM Rational ClearCase UCM This online course introduces the software developer to the principles of software configuration management, as well as to IBM Rational's software configuration management solution: IBM Rational ClearCase UCM. Conceptual explanations, typical user scenarios, and self-check exercises introduce you to using UCM to maintain and track software

changes.

- ClearCase training: SCM210, Essentials of IBM Rational ClearCase for Developers (for Microsoft Windows) This online course teaches developers how to perform common day-to-day tasks in Rational ClearCase. It focuses on teaching the concepts and skills developers need to successfully manage source code changes in their development environments.
- ClearCase training: SCM250, Essentials of IBM Rational ClearCase LT for UCM and Windows After taking this 4- to 6-hour Web-based training course, you'll understand the fundamentals of using IBM Rational ClearCase LT and applying the UCM approach to the software development process.
- Agile configuration management for large organizations From *The Rational Edge* newsletter (15 March 2007). Because of their size and organizational complexity, large companies need to embrace agile development principles. Learn how hundreds, even thousands, can work together efficiently within an agile configuration management environment.
- To learn more about IBM Rational products, visit the developerWorks Rational zone. You'll find technical documentation, how-to articles, education, downloads, product information, links to training courses, and more.
- Getting Started with the Eclipse Platform (developerWorks article, November 2002) provides a history and overview of Eclipse, including details on how to install Eclipse and plug-ins.
- Eclipse in Action: A Guide for Java Developers (Independent Pub Group, 2003) is a must-read for Java developers using Eclipse.
- Browse the technology bookstore for books on these and other technical topics.

Get products and technologies

- Download IBM product evaluation versions and get your hands on application development tools and middleware products from DB2®, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- Check out developerWorks blogs and get involved in the developerWorks community.

About the authors

Paul Boustany
Paul Boustany is a change and release management Marketing Engineer with IBM Rational Software. He has 7 years of experience with configuration management and

10 years experience in the software development industry. Paul has created many self-running demonstrations of the IBM change and release management solutions, as well as a hosted trial featuring Rational ClearCase and Rational ClearQuest.

Tim Feeney
Tim Feeney is the Change and Release Management Practice Lead for Rational Software in IBM's Americas TechWorks organization. He has 7 years of experience with Rational software development and 14 years of experience in the software development industry. Tim has created many hands on-workshops for change and release management (C&RM) and other solutions, including the recent proof of technology based on the build- and deployment-tracking features of IBM Rational ClearCase, ClearQuest, and Build Forge and for IBM Tivoli Provisioning Manager.