INTRODUCING RATIONAL SUITE CONTENTSTUDIO™

Version 2001A.04.00

Windows/UNIX Edition

Rational®
the **e-development** company™

800-024921-000

Introducing Rational Suite ContentStudio
Document Number 800-024921-000 June 2001
Rational Software Corporation 20 Maguire Road Lexington, Massachusetts 02421

Rational®
the e-development company™

Contents

Figures

Preface

Rational Suite ContentStudio is a content management environment and Web application deployment solution that supports the life cycle of Web application development.

About This Manual

This document presents the concepts behind and a usage model for Rational Suite ContentStudio. It also includes some procedures to configure and use components of the product.

Before you can use ContentStudio, you must install, license, and configure all its components. For details, see *Installing Rational Suite ContentStudio.*

Using Rational Suite ContentStudio Documentation

The documentation for this product consists of the following:

- *Introducing Rational Suite ContentStudio*
- *Installing Rational Suite ContentStudio*
- *Rational Suite ContentStudio Release Notes*
- *Rational Suite ContentStudio Sample Application Guide*
- *Using the Rational Suite ContentStudio Console* (online help)
- *Using the Rational Suite ContentStudio Administration Utility* (online help)
- *Using the Rational Suite ContentStudio ClearCase/CMS Integration* (online help)

Using Vignette Documentation

The Vignette V/5 documentation, which is available only online, describes some features that are not available in ContentStudio:

- Personalization server
- Observation Manager
- TCL page generation
- Multi-Channel Server
- Solaris platform support
- Web server support (other than IIS)

You can ignore these manuals entirely:

- *Visitor Services Guide*
- *Net Perceptions Recommendation Engine*
- *Resonate Central Dispatch User's Guide*
- *Internationalization Guide*
- *Vignette Content Manager Generator User's Guide*
- *Vignette CMS Explorer Guide*
- *Template Command Reference*

To identify which sections of the documentation you can ignore, consult the table.

Vignette manual	Sections or chapters to Ignore
Administration Guide	➤ In Chapter 4: *Viewing OMS Information*, *Viewing OMS Process Information*, *Viewing MCS Information*, and *Viewing MCS Process Information* ➤ In Chapter 8: *Obtaining Process Status—Solaris only* ➤ Chapter 11 ➤ Appendix D ➤ Appendix F
Security Guide	➤ In Chapter 2: The OMS and MCS information in *Configuring Security for the CMS, CDS, or MCS* ➤ In Chapter 6: *Specifying Securing Web Servers* and *Security for Multiple, Remote Web Servers*
Template Cookbook	Chapter 6 and Chapter 9

Using Rational Suite ContentStudio (Base Features)

The ContentStudio documentation refers to manuals that are available on the CD titled *Rational Solutions for Windows: Online Documentation*.

One manual that is not available on that CD is *Developing Software with ClearCase*. The information in this manual is available in the online help for ClearCase LT.

Typographical Conventions

This manual uses the following typographical conventions:

➤ *cs-home-dir* represents the directory into which Rational Suite ContentStudio has been installed. By default, this directory is *drive*:**\Program Files\Rational\ContentStudio**.

➤ **Bold** is used for names the user can enter; for example, all command names, file names, and branch names.

➤ *Italic* is used for variables, document titles, glossary terms, and emphasis.

➤ `A monospaced font` is used for examples. Where user input needs to be distinguished from program output, **bold** is used for user input.

➤ Key names and key combinations are capitalized and appear as follows: SHIFT, CTRL+G.

1 Rational Suite ContentStudio Concepts

Rational Suite ContentStudio is a content management environment and Web application deployment solution that supports the life cycle of Web application development. The ContentStudio environment provides an integrated model for Web application development and deployment that you can use to manage content and code.

Rational Suite ContentStudio has multiple components:

- Vignette V/5
- Rational ContentStudio Page Generator and Rational NetDeploy
- Rational Suite ContentStudio (Base Features) products:
  - Rational ClearCase LT
  - Rational ClearQuest
  - Rational Unified Process
  - Rational RequisitePro
  - Rational SoDA
  - Rational TestManager

In general, Vignette V/5, Rational ContentStudio Page Generator and Rational NetDeploy, and ClearCase LT and ClearQuest from Rational Suite ContentStudio (Base Features) have a client/server architecture.

In this manual, all references to ClearCase refer to ClearCase LT as well.

1.1 ContentStudio Life Cycle

Planning, designing, and implementing a new Web application is a complex process. This section describes the tasks in this process and how the components of ContentStudio are used to complete them.

Planning

The business analyst defines requirements for the application. A useful method for evaluating and managing business requirements is to define use cases in Rational RequisitePro. The analyst can also design the data object model required to support these requirements with Rational Rose. Rational Rose is available as a separate product from Rational Software. For more information, see *Planning a Web Application* on page 11.

When the use cases have been elaborated and validated, the design of a site architecture should support them. For example, if one of the use cases for an e-business application is to process credit card transactions, the site architecture requires an interface to an accounts receivable processing system. For more information, see *Model of a Web Application* on page 7.

Installing, Licensing, and Configuring

The administrator installs and configures the infrastructure necessary to support the application development environment. You can install ContentStudio on a single server host for evaluation or on multiple server hosts to support production work. For more information, see the manual *Installing Rational Suite ContentStudio.*

Code and Content Creation

The architecture for the Web site is created during the planning phase. Template developers can then create templates, using the ClearCase / CMS integration, to support the format and navigation of content for the application software developers to create the software infrastructure. Content contributors create new material, and editors manage the approval process. For information on developing templates, see *Coding Rules for Templates* on page 19 and for more information on the ClearCase / CMS integration, see Chapter 5, *Using the ClearCase/CMS Integration*.

Page Generation

The site manager defines an interval (for example, nightly) at which approved templates and content records are generated as HTML pages and merged with the software infrastructure to build a complete staging environment for the application as it evolves. Working in the staged environment, testers can verify new development work. As they encounter problems, testers create a feedback loop to template developers, software developers, and content contributors in the form of change requests.

The project manager can create reports from these change requests to monitor the progress toward project milestones. Application development can be managed as a software development process. For more information, see Chapter 3, *Content Change Requests in ClearQuest*.

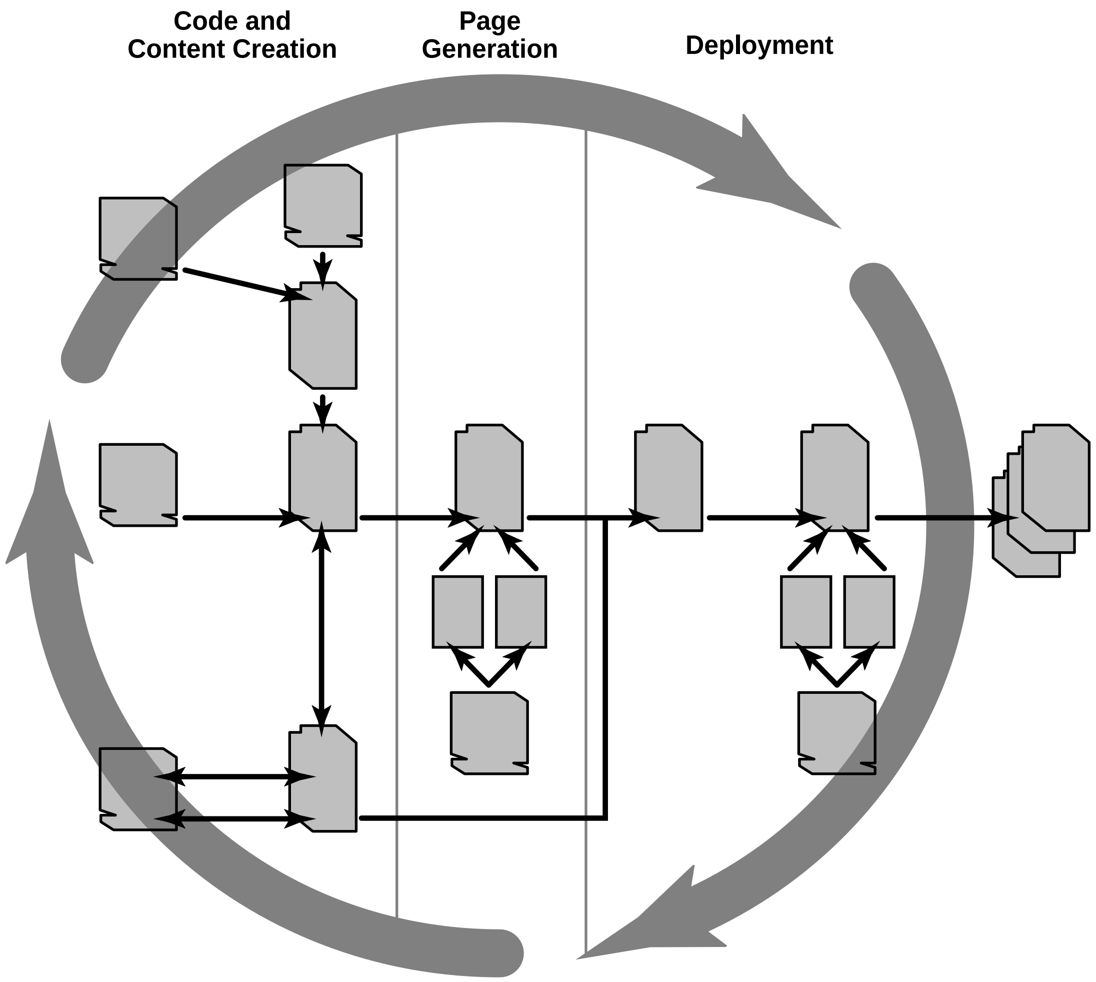
Deployment

When all bugs are resolved and the application is ready to deploy, the site manager uses Rational NetDeploy to define a deployment task to copy the Web application files to one or more target Web servers (inside or outside a firewall). For more information, see *Deployment* on page 24.

In addition, ContentStudio supports the use of Rational NetDeploy triggers that run when executing deployment tasks. Rational NetDeploy triggers are user-defined routines for processing before or after a specified operation. For more information, see Chapter 4, *Using Rational NetDeploy Triggers*.

Figure 1 shows the ContentStudio process. The further information on the process details for Figure 1, see *Code and Content Creation*, *Page Generation*, and *Deployment* in Chapter 2. For complete information on the installation phase, see *Installing Rational Suite ContentStudio;* for information on the planning phase, see *Planning a Web Application* on page 11.

Figure 1 ContentStudio Model



After the Web application is up and available, the need for changes—in the form of updates and new additions to the site—becomes apparent. This iterative process of developing code and content, generating HTML pages, internal testing in a staging area, and deployment keeps a site fresh. By streamlining this process, ContentStudio helps you focus on the content of these changes, not on the mechanics of creation, review, approval, and posting.

1.2 ContentStudio System Environment

After ContentStudio is installed, the host environment includes development servers and production servers. The recommended installation model for the development servers is to use four server hosts (Figure 2).

Figure 2 System Environment

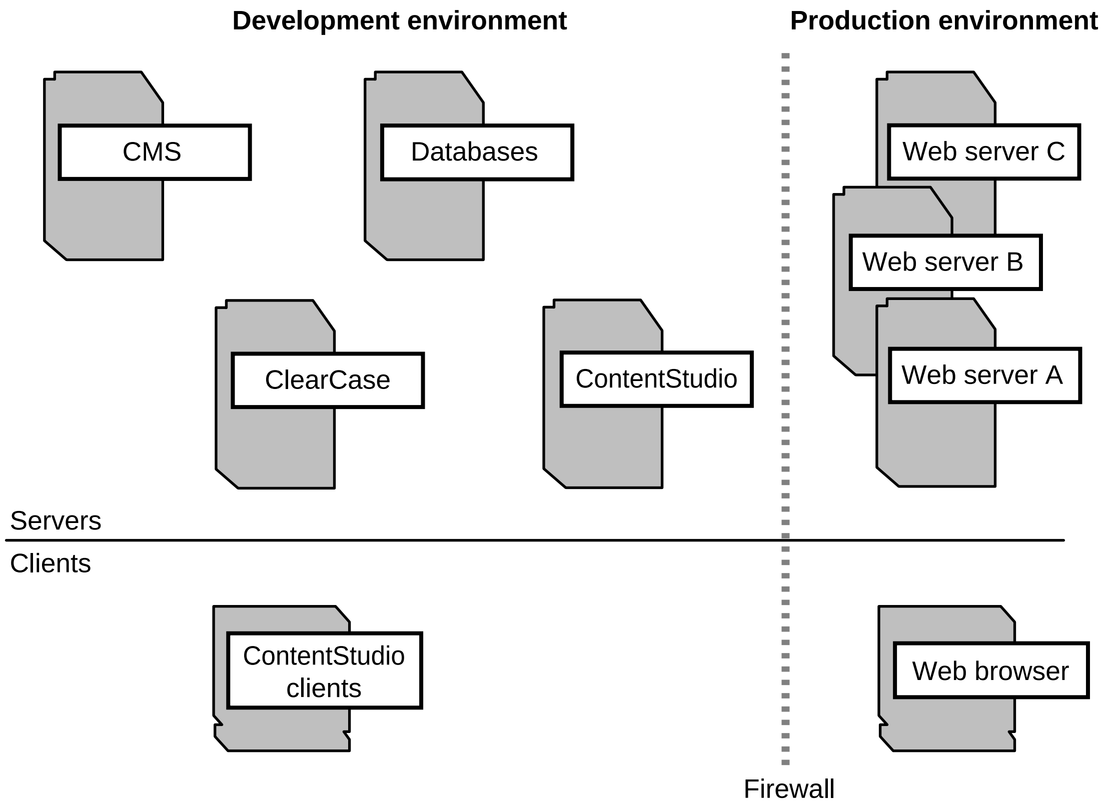


Table 1 describes the clients and servers in Figure 2.

Table 1 ContentStudio Clients and Servers

Name of client or server	**Description**
CMS server	Host on which Vignette CMS is installed for all V/5 projects.
Database server	Host on which the CMS database and the ClearQuest database reside. The CMS database stores all content items; the ClearQuest database is configured with the ContentStudio package.
ClearCase server	Host on which ClearCase is installed and the content VOB and code VOB reside. The code VOB is used for template development and for software development on the Web application.
ContentStudio server	Host on which the Page Generator, Rational NetDeploy, and the ClearQuest integration back-end reside.
Production Web server	Web host on which Rational NetDeploy agent is configured. This server is a target for deployments.
ContentStudio clients	Client host that includes the following client software: ➤ ContentStudio Console, used to define and manage the Page Generator and Rational NetDeploy. ➤ ClearQuest client, used to access the ClearQuest database for change requests. These requests include content change requests for the ClearQuest integration with the V/5 task list. ➤ V/5 tools client, used to manage the Vignette environment, including the V/5 task list, which is integrated with the content change requests in the ClearQuest database.
Web browser	Client that accesses HTML pages for the Web application deployed to the production Web servers.

For more information on installing the system environment, see *Installing Rational Suite ContentStudio.*

After communication between the development servers is set up, Web team members can use the V/5 tools and ContentStudio Console to work with the development servers. ContentStudio clients enable Web teams to create, edit, review, and test applications in the development server environment and deploy approved files to production servers. Typically, the production servers are Web servers that are available to your site's visitors. These servers can reside inside or outside a firewall.

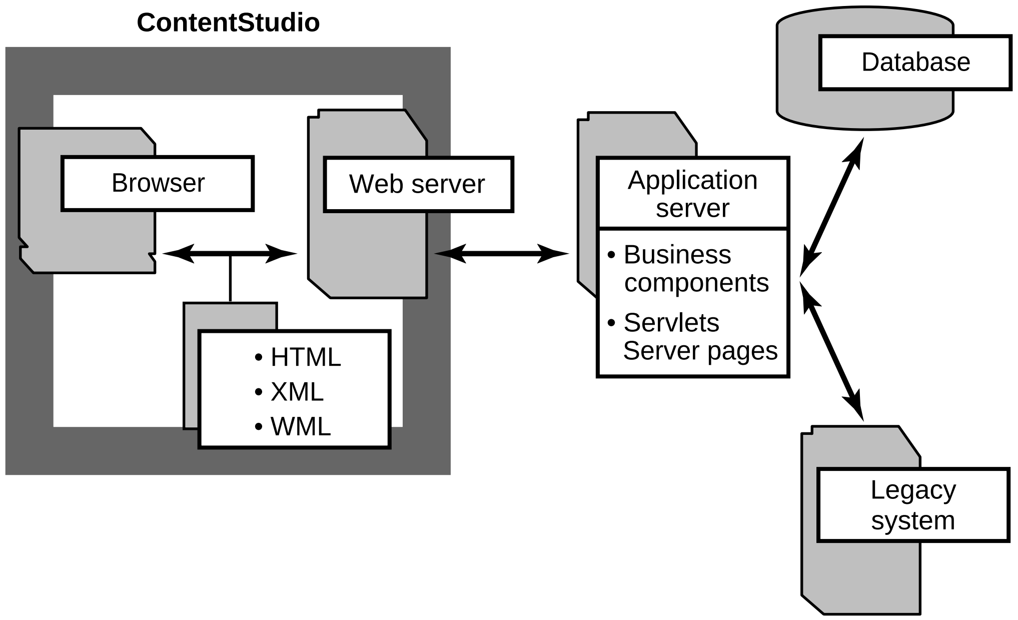
1.3 Model of a Web Application

A Web application combines content and code. When an ASP or JSP template from a code VOB is processed through ContentStudio page generation, it becomes an HTML page that resides in a content VOB. After the page is tested and deployed to a Web server, users can download it as a URL from a Web server. That same page can also contain scripts that run when users initiate events (clicking an object on the page, for example). The script may contain a reference to an application that resides on the server. The code on the server is designed to run there or in a virtual machine environment on a visitor's desktop.

Figure 3 shows how ContentStudio is used to build the run-time environment for a production Web server. Web teams can design code to interact with back-end business components or servlets, which in turn may access additional databases or legacy systems.

The performance of e-business sites depends on the effective management of content and code. ContentStudio enables Web teams to develop content with V/5 tools and to develop ASP or JSP template code and Web application code in a programming IDE. Both the content VOB and code VOB are source controlled by Rational ClearCase.

Figure 3 Web Application Model

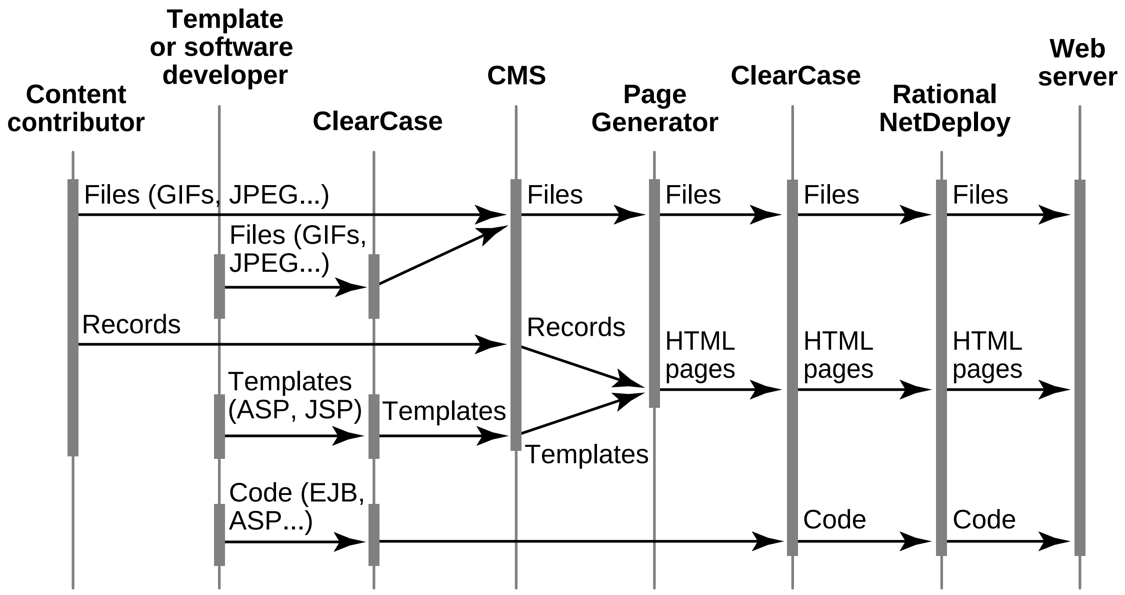


ContentStudio also provides a single mechanism, Rational NetDeploy, to transfer all changes to all live Web sites at the same time. This mechanism eliminates the tedious task of identifying individual files to be copied. It also permits Web teams to easily recognize which changes were made and when, and to synchronize work from multiple projects. In addition, you can design Rational NetDeploy triggers to perform such actions as starting and stopping a Web server, registering a COM or EJB object, and sending e-mail notification when a deployment task completes.

1.4 Flow of Artifacts

The Rational Unified Process defines an artifact as a final or intermediate work product that is produced and used during a project. Figure 4 illustrates how artifacts for a Web application move from creation to deployment. The Web pages that users access are created from content and code artifacts in the ContentStudio environment.

Figure 4 Flow of Application Artifacts



The V/5 environment utilizes three types of artifacts:

- Files (for example, .**gif**, .**jpeg**)
- Templates for a content development or content management application
- Content records (rows in the CMS database schema)

ContentStudio enables you to create and manage static files (such as .**gif** and .**jpeg**) and templates (ASP or JSP) in ClearCase. Using the ClearCase/CMS integration, you can use a programming IDE to develop ASP or JSP templates residing in the code VOB. When you check in templates or static files from ClearCase, a copy of the checked-in version of the element is imported to the content management server (CMS). For more information, see Chapter 5, *Using the ClearCase/CMS Integration*.

All V/5 artifacts are stored in the database for the CMS. Typically, contributors create content by means of a content management application (CMA), which places files and content records (text and graphic files) in the CMS database.

V/5 allows you to separate the creation of content from its presentation. V/5 templates control the format and navigation of content on the Web page. As the V/5 artifacts pass through the Page Generator, static Web pages are created in the content VOB. The content VOB contains HTML pages and static files. Interactive features for the Web application are added to these Web pages by deploying code, from a code VOB, along with them.

The code VOB can contain ASP and JSP template source files and static files and Web application source code (for example, .**java** or .**c** files) and derived objects (for example, .**jar**, .**dll**, or .**exe** files).

For information on developing a Web application, see the *Rational Suite ContentStudio Sample Application Guide*, which is located in the **\Samples** directory in *cs-home-dir* on the ContentStudio client or server host. For information on coding ASP templates, see the Vignette V/5 manual *COM Extension Programmer's Guide and Reference*. For information on coding JSP templates, see the Vignette V/5 manual *Java Extension Programmer's Guide and Reference*.

‘

ContentStudio Process Model

2

This chapter describes the phases of Web application development and explains how the components of Rational Suite ContentStudio work together during each phase.

2.1 Planning a Web Application

This section describes the important aspects of ContentStudio that you ought to consider when you do the planning for the Web application.

Define and Analyze the Business Requirements

You need to define the business requirements for the Web application. The Rational Unified Process recommends that you do so through use cases. You can use Rational RequisitePro to manage your use cases. RequisitePro is a product in the Rational Suite ContentStudio (Base Features) for requirements management that is also a component of ContentStudio. An analysis of the business requirements may indicate that new code is required to support functions described by a use case. In this situation, you may need to construct a data object model for a production server database to interface with the Web application code.

Define a Data Object Model for the CMS Database

You need to define a data model for the content records stored in the content management server (CMS) database that supports the business requirements.

Define the CMA

A content management application (CMA) enables content contributors to create, edit, and delete content records in the CMS database. Before template developers can begin to create V/5 templates for the CMA, you must define both the functions that content contributors will need and the user interface for the CMA.

For more information on defining and coding a CMA template, see the *Rational Suite ContentStudio Sample Application Guide*, which is located in the **\Samples** directory in *cs-home-dir* on the ContentStudio client or server host.

Define the CDA

A content delivery application (CDA) is a set of V/5 templates that process data from the CMS database, including the data records submitted through the CMA and files (such as **.gif** and **.jpeg**).

Before template developers can begin to create V/5 templates for the CDA, you must define the following:

➤ CMS project hierarchy

➤ Security model for users and roles

➤ V/5 workflow model

➤ Template architecture and the set of templates required

➤ Path information for templates, which is stored as a property of each template

➤ Templates that need **<applet>** tag references or JavaScript stubs to reference Web application code that resides on the production Web server

In addition, you can use the ClearCase/CMS integration to develop V/5 templates and files (such as **.gif** and **.jpeg**) for the CDA. For more information, see Chapter 5.

Define Web Application Code

If any business requirements defined in use cases will need code that resides on a production Web server, you must define the functions supported by this application code.

For more information on a Web application model, see *Model of a Web Application* on page 7.

2.2 Code and Content Creation

Content contributors create content artifacts in the V/5 environment using a CMA or the V/5 tools client. Template developers and software developers create code artifacts using a programming tool, such as an integrated development environment (IDE) or an editor.

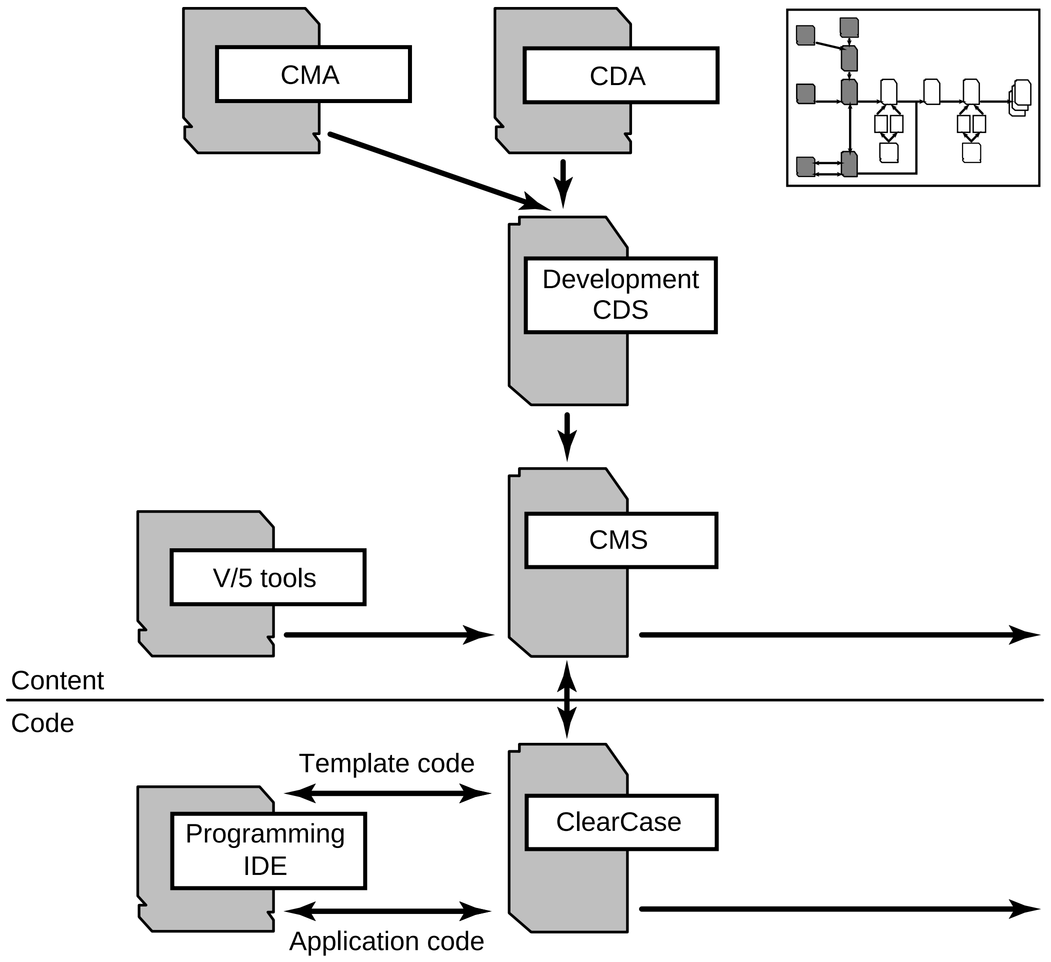
The V/5 environment supports the management of content artifacts through workflow events. The V/5 task list is used to manage workflow for a CMS project, including all subprojects. All content artifacts inherit the workflow defined for the CMS project, but a workflow defined for an individual content record overrides the project workflow. Typically, in V/5 workflow, content contributors create artifacts that editors must review and approve. (For more information on using the V/5 workflows , see the Vignette V/5 manual *Production Center Guide*.)

When a content artifact is launched, it is in a live state. In this state, a copy of the artifact in the Development CDS is mirrored in the Live CDS. The usage model for ContentStudio assumes that all CDA artifacts in the Live CDS are candidates for asset lists and for deployment to a production Web server. (CMA artifacts are not mirrored in the Live CDS; all of them should be marked **Internal use only**.)

The programming team may choose to use the ClearCase UCM model to manage the software development workflow for Web application code and V/5 templates (using the ClearCase/CMS integration). For more information on the ClearCase UCM model, see the manual *Developing Software with ClearCase*.

Figure 5 represents the content and code creation model for ContentStudio. Using a CMA, a content contributor can enter content records into the CMS database. A template developer uses a programming IDE (for example, Visual Basic or Forte for Java) and the ClearCase/CMS integration to create ASP or JSP templates. When a template is checked in to the code VOB, a copy is mirrored in the CMS database. In addition, a software developer uses a programming IDE (for example, Visual Age for Java) to create Web application code that is also stored in the code VOB. In the ContentStudio environment, the infrastructure for content creation in Vignette is separate from the infrastructure for code creation in ClearCase. Consequently, software developers and template developers can use programming tools that are specific to their coding needs.

Figure 5 Code and Content Creation Model



Approval Workflow Model Based on a CMS Project

All content items (files, templates, and content records) entered into the CMS database are subject to a workflow model defined in the CMS project. Content records are entered into the

CMS database using a CMA and templates are entered into the CMS database using the ClearCase / CMS integration. Files can either be entered into the CMS database by using a CMA or the ClearCase / CMS integration. Using workflow, all created or modified content artifacts are routed through an approval step before the content artifact is in a live state. The V / 5 task list displays the state of all content items in workflow. ContentStudio supports an integration between V / 5 and ClearQuest (that is using a ContentChangeRecord record). For more information on the ClearQuest integration, see Chapter 3, *Content Change Requests in ClearQuest*.

2.3 The ContentStudio Console

To create and manage page generation and deployment tasks, Rational NetDeploy trigger definitions, and a mapping for the ClearCase / CMS integration, you use the ContentStudio Console. Any V / 5 user who is a member of **CS_Role** (the ContentStudio role) can use the console. The ContentStudio Console is installed as part of ContentStudio Server or ContentStudio Client. For more information, see *Installing Rational Suite ContentStudio*.

To start ContentStudio Console, click **Start**>**Programs**. Depending on which product you installed the console from, click one of the following:

- **Rational Suite ContentStudio Server**>**Rational ContentStudio Console**
- **Rational Suite ContentStudio Client**>**Rational ContentStudio Console**

For information when using the ContentStudio Console, click **Help**>**Help Topics**.

2.4 Page Generation

Page generation is the process of converting the content artifacts (templates, content records, and files) created in the CMS database to HTML files and files (for example, **.jpeg**), and storing these files in ClearCase. This process consists of three steps:

1. The ContentStudio Console user creates a list of the CMS content artifacts to convert.
2. The ContentStudio user defines a page generation task for the conversion.
3. The Page Generator stores both new and changed HTML pages and files in a ClearCase content VOB.

The list of artifacts to convert is called an *asset list*; each item on the list is an *asset item*. The items on the asset list that are actually processed during page generation are those that are in a live state on the Live CDS.

Page generation tasks run in the page generation view, which selects the latest version of each element on the **main** branch in the content VOB. This is the configuration specification (config spec) for the page generation view:

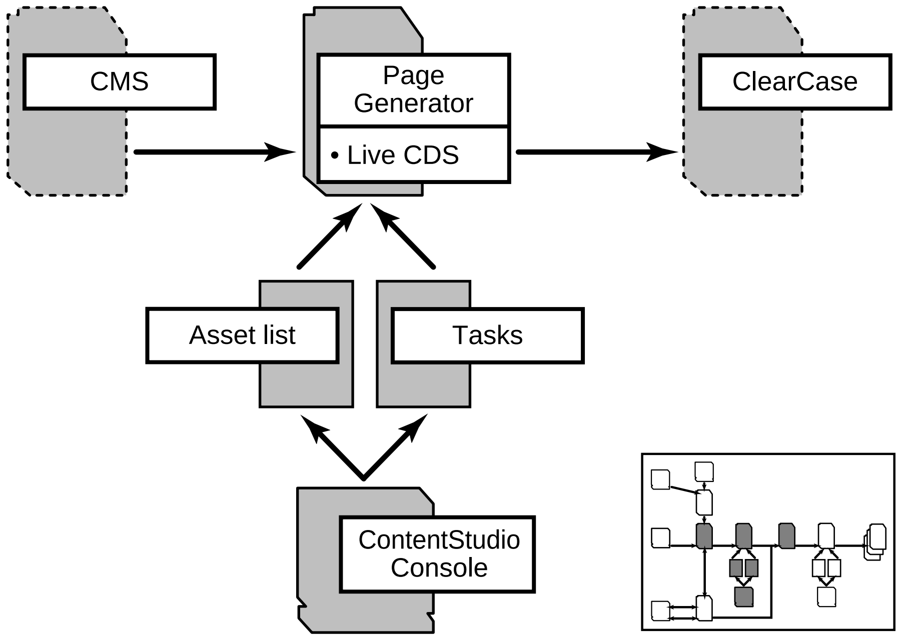
```
element * CHECKEDOUT
element *\main\LATEST
load \contentVOB
```

When a page generation task runs, the Page Generator creates a new element in the content VOB for new items on the asset list. If the HTML page or file already exists in the content VOB but has changed, a new version of the element is created. The directories and files that result represent the accumulation of all page generation tasks run up to that point.

WARNING: The page generation view is a ClearCase view that is intended for use by ContentStudio only. Page Generation errors will occur if you change the config spec, create view-private files, or check out files in this view.

Figure 6 illustrates how page generation works.

Figure 6 Page Generation Model



Managing Asset Lists

You can include any of these items on an asset list:

- ➤ A V/5 project or subproject
- ➤ A V/5 file (**.gif**, **.jpeg**)
- ➤ A V/5 template (ASP or JSP coded only)
- ➤ A V/5 content record

Synchronizing Asset Lists

You can synchronize the CMS database with the content VOB. If you delete content items from the CMS project or change the path information for a template or file, you need to create and run a synchronization task.

NOTE: Using a **cleartool cmelem**, **cleartool rmname**, **cleartool rmver**, or **cleartool move** command when working in the ClearCase / CMS integration changes the CMS project. You must run a synchronization task after executing any of these commands. For more information on the ClearCase / CMS integration, see Chapter 5.

When a synchronization task runs, the content items in the CMS database are compared with all elements that are currently in the content VOB. If content items have been removed from the CMS database but exist in the content VOB, all corresponding elements in the content VOB are removed by means of the ClearCase **cleartool rmname** command.

Managing Page Generation

If page generation is to work properly, the associated CDA templates must satisfy the following conditions:

- ➤ They must conform to ContentStudio coding rules. (See *Coding Rules for Templates* on page 19.)
- ➤ They must be included on an asset list (directly or indirectly by including the V / 5 project to which the asset belongs) that is scheduled for page generation.
- ➤ They must have a status of live and be in the Live CDS.

NOTE: If a content item is on an asset list but not in the Live CDS, the asset item name appears in a page generation logging error. To correct the error, run the page generation task after the content item is in the Live CDS.

Establishing page generation Schedules

After you create an asset list, you can establish a page generation schedule for it. You can define the schedule as one of the following:

- ➤ Immediately
- ➤ Once only at a specified date and time
- ➤ At specified intervals starting at a specified date and time
- ➤ After another task completes

For more information, see the online help for ContentStudio Console.

Coding Rules for Templates

ContentStudio supports CMA and CDA templates. Because each has different functions in ContentStudio, these templates have different coding rules. The CMA templates are not processed by the Page Generator, but CDA templates are.

All CMA and CDA templates used with ContentStudio must be coded in ASP or JSP.

The coding rules for CDA templates are more restrictive to ensure reliable page generation:

- The template must be cached (first argument for CURL = 0).
- The template must have a path defined.
- The template must not be designated for internal use only.
- The template cannot support any user input.
- The template cannot support browser variations (fourth argument for CURL = 00).
- The template must use a standard CURL; you cannot append any data to the end of a CURL.
- The template must not depend on a variable in the HTTP header.
- If a record ID is defined in the CURL, the template must specify a primary table. The converse also applies: if a record ID is not defined in the CURL, the template must not specify a primary table.
- All content records must be registered with the CMS.

If these rules are not followed, an error appears in the page generation log.

To make the testing and debugging easier, we recommend that each template in your V/5 CMS project include an HTML comment that specifies the name of the template or provides information to identify it. After you run a page generation task and are testing content in a staging area, if you detect a problem in a specific HTML page, you can determine where the problem is coming from by referring to the HTML comment in the HTML source.

Page Generator Processing

The actions of page generation are as follows:

- ➤ If the CDA template has defined a primary table, an HTML page is created for each record in that table that is in a live state and for each path specified by the template.
- ➤ If the CDA template has not defined a primary table, a single HTML page is created for each path (defined in the template).
- ➤ For all component templates, when ContentStudio page generation runs, **/component** is prepended to the template's path.
- ➤ If an asset list contains an asset defined as a V/5 file content artifact, the file is passed through to page generation output.
- ➤ Page generation for the same asset list may run many times in the course of project development. Each time a page generation task executes, a new version of an HTML page or a file is checked in to the ClearCase content VOB only if the latest version of the HTML page or file is different from its predecessor.
- ➤ All files, content records, and templates must be in a live state to be page generated by ContentStudio.
- ➤ If a content record is in the CMS database in a live state, an HTML page is generated for each template with a primary table equal to that of the record for each path specified by the template.
- ➤ When page generation runs, the file name for the HTML page that is created is the CURL with an extension as specified in the template.

When a content item for a template or a file is both on an asset list scheduled for page generation and in a live state, the Page Generator creates a ClearCase element in the content VOB in the same directory location specified in the **Path(s)** box in the **V/5 Details** dialog box.

NOTE: When using the ClearCase/CMS integration, you must create and modify **Path** information in a ClearCase view. For more information, see Chapter 5.

In Figure 7, the path for this template is *docroot***/ClassicsCD/review/detail**, where *docroot* is the doc root of the Web server (for example, **C:\Inetpubs\wwwroot**).

The HTML page that is created resides in the content VOB at *content-VOB-root***\ClassicsCD\review\detail**; the file name is the CURL. The format of a CURL is

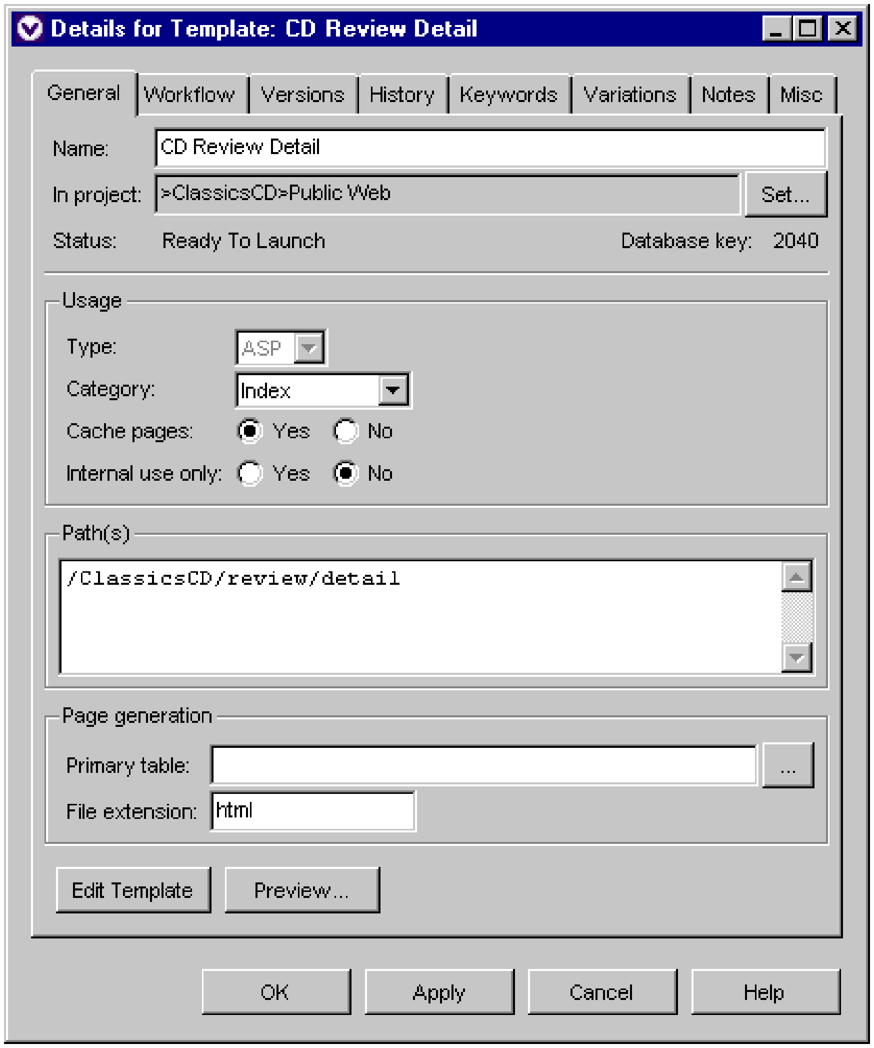
uc, *ttt*, *rrr*, *vvv*

where

- *uc* is 0 for a cached template and 1 for an uncached template. This value is always 0 for ContentStudio.
- *ttt* is the template ID number.
- *rrr* is the record ID number.
- *vvv* is the template variation. This value is always 00 (no variation) for ContentStudio.

For more information on CURLs, see the Vignette *Template Cookbook*.

Figure 7 V/5 Path Details for Templates and Files


Details for Template: CD Review Detail

General | Workflow | Versions | History | Keywords | Variations | Notes | Misc

Name: CD Review Detail
In project: >ClassicsCD>Public Web
Status: Ready To Launch
Database key: 2040

Usage
Type: ASP
Category: Index
Cache pages: [✓ Yes] No
Internal use only: Yes [✓ No]

Path(s)
/ClassicsCD/review/detail

Page generation
Primary table:
File extension: html

Model for Staging and Testing

After a page generation task completes and the versions of ClearCase elements in the content VOB are updated, you can stage the content for testing. You can also stage and test Web application code (from the code VOB) and content (from the content VOB) together by deploying the Web application to a staging area (typically inside the firewall). For more information, see *Deployment*.

Figure 8 Code and Content VOBs

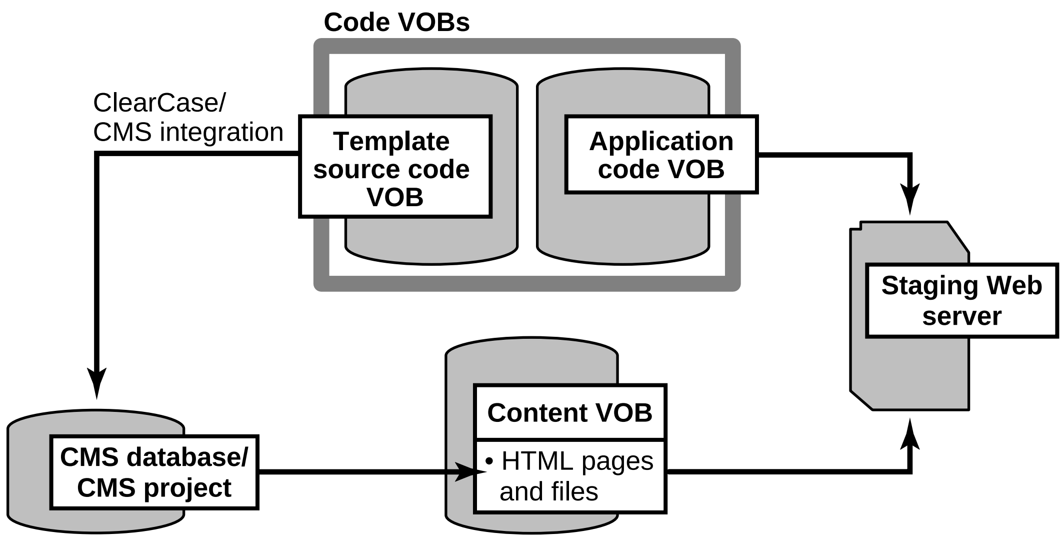


Figure 8 demonstrates the three stages for staging a Web application for testing:

1. Application code and template code are created and developed in the code VOB. When template code is checked in from the ClearCase / CMS integration, a representation of the checked-in version is mirrored in a CMS project.

2. When an asset list is created and a page generation task is scheduled using ContentStudio Console, the resulting HTML pages and files (such as **.jpeg** and **.gif**) reside in the content VOB.

3. A deployment task is created to copy application code derived objects from the code VOB and HTML pages and files from the content VOB to a staging Web server for testing.

In either case, you can use the ClearQuest integration to track requests for changes. You can synchronize content change requests in ClearQuest with the V / 5 task list. In addition, you can submit defects in ClearQuest and create a child association with a ContentChangeRequest record. You can also use ClearQuest uniform reporting to generate reports of code and content defects. For more information on the ClearQuest integration, see Chapter 3, *Content Change Requests in ClearQuest*.

The development cycle continues until the Web application achieves an acceptable level of quality. At this point, the Web application last deployed to the staging area can be redeployed to target Web servers. For more information, see *Rollback, Redeploy, and Label Conversion Tasks* on page 29.

2.5 Deployment

The deployment process transfers files from code and content VOBs to target Web servers. A deployment task specifies the code and content artifacts to be deployed to servers on which the Rational NetDeploy agent is installed and configured. For information on the installing and configuring a Rational NetDeploy agent on supported Windows and UNIX platforms, see *Installing Rational Suite ContentStudio.*

Rational NetDeploy uses a ClearCase view to select the code and content artifacts that are copied to the target Web servers. This deployment view is created during the deployment task execution. It is not a public view, but it is configured with task information that you provide when you create deployment tasks with ContentStudio Console.

When a deployment task runs, the artifacts selected by the deployment view are copied to Web servers specified in a topology list. If Rational NetDeploy triggers are enabled for the deployment task, associated trigger actions may fire during the deployment task. (For more information, see Chapter 5.) Also, as part of the deployment, the target directories are cleaned up so that only deployed files reside in them; any files deleted or renamed in the production environment are removed.

NOTE: If you have copied code or content to the target Web server without using a deployment task, Rational NetDeploy does not create entries in the ContentStudio database for such files and directories. To include them on your Web site, you should delete them before you attempt to re-create them by means of a deployment.

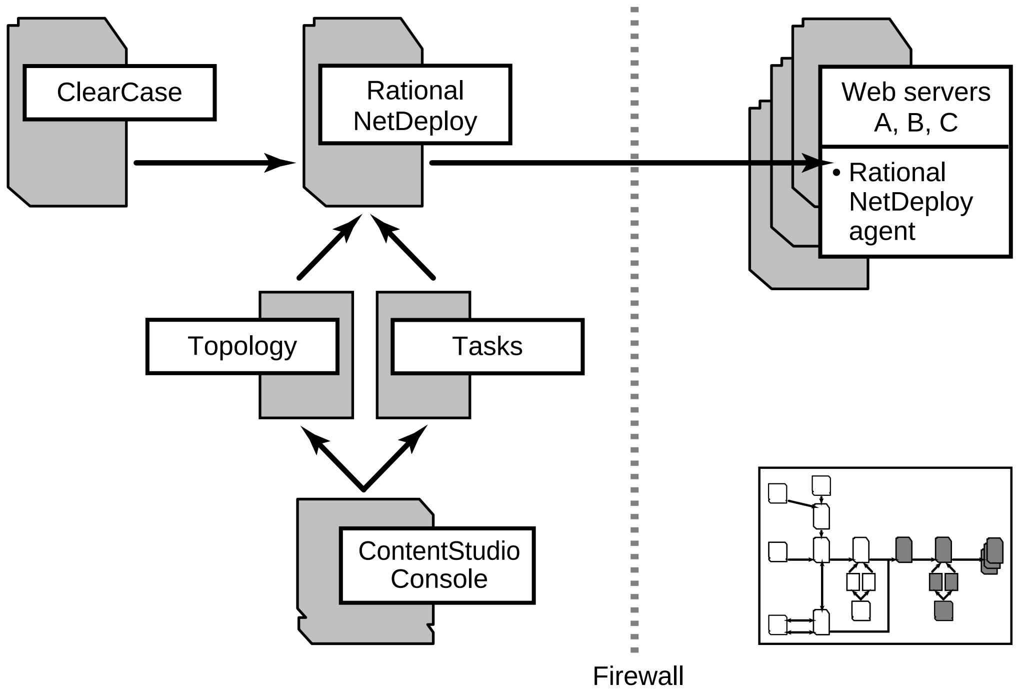
Before you can deploy the Web application, you must define the following items in the ContentStudio Console:

➤ A topology that includes all possible target Web servers and specifies whether the communications to each target Web server will be secure by means of SSL

- A mapping of directories in all possible content and code VOBs to directories on all possible target Web servers
- A deployment task that defines the following:
  - A directory map (or a subset of the topology)
  - A copy policy
  - Which versions to select / deploy
  - A deployment schedule
  - Whether Rational NetDeploy triggers are enabled

Figure 9 illustrates the ContentStudio deployment model. The ContentStudio Console is used to manage server topology information and deployment tasks. Rational NetDeploy accesses artifacts stored in ClearCase content and code VOBs and manages the copying of those artifacts defined in a deployment task. Target Web servers can reside inside or outside the firewall; secure SSL communications are supported to Web servers outside the firewall.

Figure 9 Deployment Model



Managing Server Topology Lists

You must define a topology list for the Web servers on which you have configured a Rational NetDeploy agent. For instructions on configuring a Rational NetDeploy agent, see *Installing Rational Suite ContentStudio.*

Use the ContentStudio Console to create, edit, or delete servers from the server topology list.

For each server defined, you must specify a port number and whether Rational NetDeploy communications to the Web server should be secure or not. Optionally, you can choose to use SSL to define secure communications for a Web server.

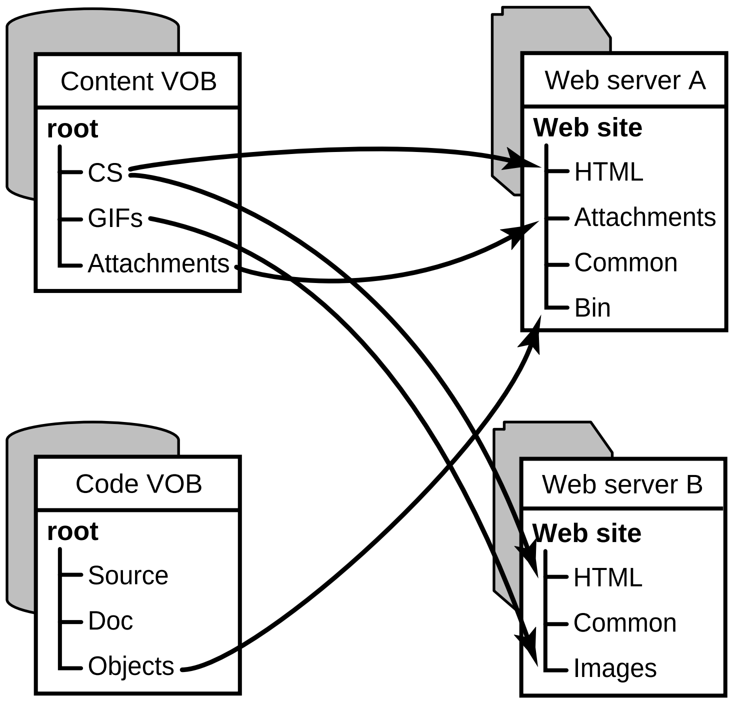
For more information on defining and managing a server topology list, see the online help for ContentStudio Console.

Managing Directory Maps

You must define mappings for all directories in your VOBs from which you will deploy artifacts. The source directories in the content and code VOBs are mapped to a destination directory on each target Web server to which the files are copied when a deployment task runs.

Directory mapping is controlled by the site manager using the ContentStudio Console. In Figure 10, Web server A and Web server B each have an existing target directory **\HTML**. The site manager can define the directory mapping for **\CS** directory in the content VOB to **\HTML** on both Web servers. The **\Objects** directory in the code VOB maps only to the **\Bin** directory on Web server A.

Figure 10 Directory Mappings



Managing Versions to Deploy

When you define a deployment task, you can use one of these methods to identify which versions of which artifacts to include:

➤ Use the latest version on the **main** branch. This is the default method.

➤ Use a version identified by a label.

NOTE: To use this method, you must label both the specific artifacts and all intervening directories in the VOB directory hierarchy up to the VOB root.

➤ Use the versions selected by a config spec.

If you want to deploy code artifacts that have been developed using ClearCase UCM, you may prefer to use the config spec for an integration view to select the versions.

Config spec from an integration view

```
element * CHECKEDOUT
element * ...branch2\LATEST
element * ...branch1\LATEST -mkbranch branch
element * CSLABEL -mkbranch branch1
element * \main\LATEST
```

If you want to include content artifacts in this deployment, you must add one rule to this config spec after the **element * CHECKEDOUT** rule.

Updated config spec

```
element * CHECKEDOUT
element \contentVOB\... \main\LATEST -time <TIMEVAL>     #New rule added
element * ...branch2\LATEST
element * ...branch1\LATEST -mkbranch branch2
element * CSLABEL -mkbranch branch1
element * \main\LATEST
```

This new rule selects the latest versions of elements on the **main** branch in the content VOB.

Deployment Using Shadow Directories

Rational NetDeploy supports a shadow directory option that is enabled on a per-server basis. If this feature is enabled for a server in the topology list, the deployed files are copied to a directory named *target-directory***-temp@@** on the target server, where *target-directory* is the designated target directory, defined using ContentStudio Console directory map for each target Web server. When a deployment, redeploy,or rollback task is executed, for every server that is enabled for shadow directories, the Rational NetDeploy agent running on the target Web server creates a temporary directory named *target-directory***-temp@@**. When all files are successfully transferred, the files in *target-directory***-temp@@** are moved to *target-directory.*

To configure the shadow directory option for each server in the topology list:

1. Click one of the following:

   ➢ **Start**>**Programs**>**Rational Suite ContentStudio Server**>**Rational Suite ContentStudio Console**

➢ **Start**>**Programs**>**Rational Suite ContentStudio Client**>**Rational Suite ContentStudio Console**

2. Click **Rational NetDeploy**, click the **Topology** tab, and then click **Help**>**Topology Help** for information on how to specify shadow directory support for each server.

Rollback, Redeploy, and Label Conversion Tasks

ContentStudio Console supports four types of Rational NetDeploy tasks: deployment, rollback, redeploy, and label conversion. Only the deployment task has been discussed so far.

A rollback task enables you to return to any previous edition of the Web application on any specified server. A redeploy task enables you to use any existing deployment task for any server and rerun that task for a different server. Rollback and redeploy tasks are possible because labels are created for all files that are sent to Rational NetDeploy agents during a deployment task.

Rollback and redeploy tasks perform different operations that are valuable for internal and external deployments. An internal deployment is useful for testing artifacts in a staging area; an external deployment transfers tested files to a production Web server, residing either inside or outside a firewall.

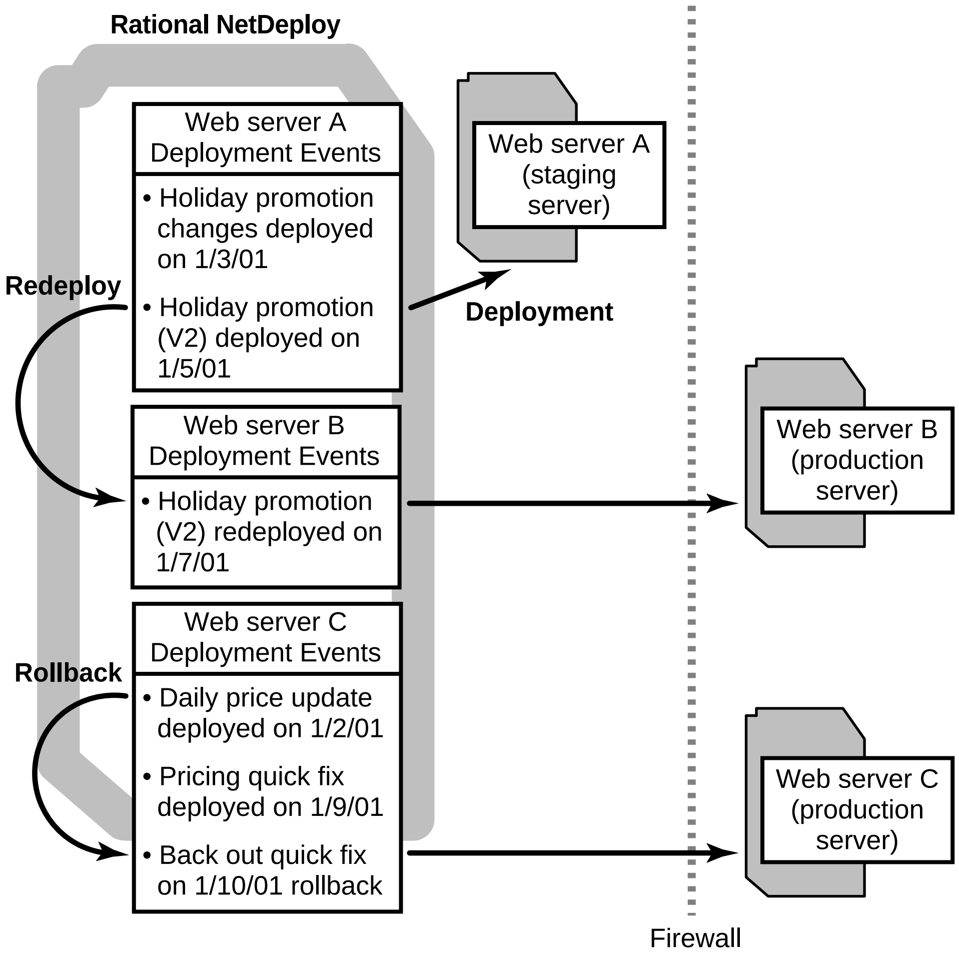
To test a Web application, artifacts from the content and code VOBs must be deployed to an internal Web server that serves as a staging area. When all problems with content and code are resolved, you can deploy the corrected versions of the artifacts on the staging server to the target Web servers.

The ContentStudio server keeps a history of all deployment tasks. If you rerun a deployment task from a specific date and time, Rational NetDeploy creates a ClearCase view that selects all files for the date and time specified.

Figure 11 illustrates rollback and redeploy. On January 3, 2001, the holiday promotion changes were initially deployed to an internal staging area on Web server A. The holiday promotion changes were tested on Web server A and contained format problems. On January 5, 2001, the corrections were deployed to Web server A. Further testing indicated that the problems were corrected and the changes were ready for production. On January 7, 2001, the deployment task for January 5 was redeployed to production on Web server B.

On January 2, 2001 a price fix was deployed to Web server C and was updated by a subsequent deployment on January 9. The pricing data deployed on January 9 was incorrect, so a rollback to the January 2 deployment task was executed for Web server C.

Figure 11 Deployment, Rollback. and Redeploy Tasks


Rational NetDeploy
Web server A Deployment Events
• Holiday promotion changes deployed on 1/3/01
• Holiday promotion (V2) deployed on 1/5/01
Redeploy
Web server B Deployment Events
• Holiday promotion (V2) redeployed on 1/7/01
Web server C Deployment Events
Rollback
• Daily price update deployed on 1/2/01
• Pricing quick fix deployed on 1/9/01
• Back out quick fix on 1/10/01 rollback
Web server A (staging server)
Deployment
Web server B (production server)
Web server C (production server)
Firewall

To define a label conversion task, you must specify a server and a previous deployment. When the label conversion task runs, all files that have been deployed to the server (up to the point in time of the specified previous deployment task) are labeled. For example, Table 2 demonstrates that after the initial full deployment, Deploy0, to server Victory, label **D0** was created in the ContentStudio code VOB for all versions deployed. At the time of Deploy1, label **D1** is created for the updated version of file A. Then Deploy2 runs and label **D2** is created for the updated version of file B. When Deploy 3 runs, label **D3** is created for the updated version of file C. At this point, a label conversion task for server Victory, designating the Deploy3 deployment, creates and applies label **D3** for all files that have been deployed to Victory up to the point in time. The result of the label conversion task is all files that have been deployed to server Victory, with respect to a user-defined point in time, can be specified by using label **D3**.

Table 2 Example Label Conversion Task

Deployment sequence	**Task name**	**Files deployed to server Victory**	**Current label for deployed files in content VOB**
1	Deploy0	A, B, C	**D0**
2	Deploy1	A	**D1**
3	Deploy2	B	**D2**
4	Deploy3	C	**D3**
5	Run label conversion for serverVictory for previous deployment Depl0y3	N / A	After the label conversion task is complete, the elements have these labels: A: **D0**, **D1**, **D3** B: **D0**, **D2**, **D3** C: **D0**, **D3**

Archiving Deployment Tasks

After a deployment task executes, Rational NetDeploy creates and applies a ClearCase label to every deployed element in a code or content VOB. Rational NetDeploy uses these labels as a means of tracking deployment tasks to support redeployments and rollbacks.

When a VOB contains large numbers of labels, its performance eventually suffers. To preserve your deployment history, you will need to back up an archive version of your VOBs before you remove old label types. If you run lots of deployment tasks and do so frequently, you will need to devise a backup and archiving strategy for your VOBs and a strategy for removing label types from them. For more information on backing up ClearCase LT and ClearCase VOBs, see *Administering ClearCase LT* and *Administering ClearCase*.

The ContentStudio database contains records for your entire deployment history. Before you remove label types from your VOBs, you must synchronize the backup of the ContentStudio database with the backup of your VOBs. When you back up the VOBs and the ContentStudio database, we recommend that you lock them and back them up together. If you ever need to restore them, they will be synchronized.

3 Content Change Requests in ClearQuest

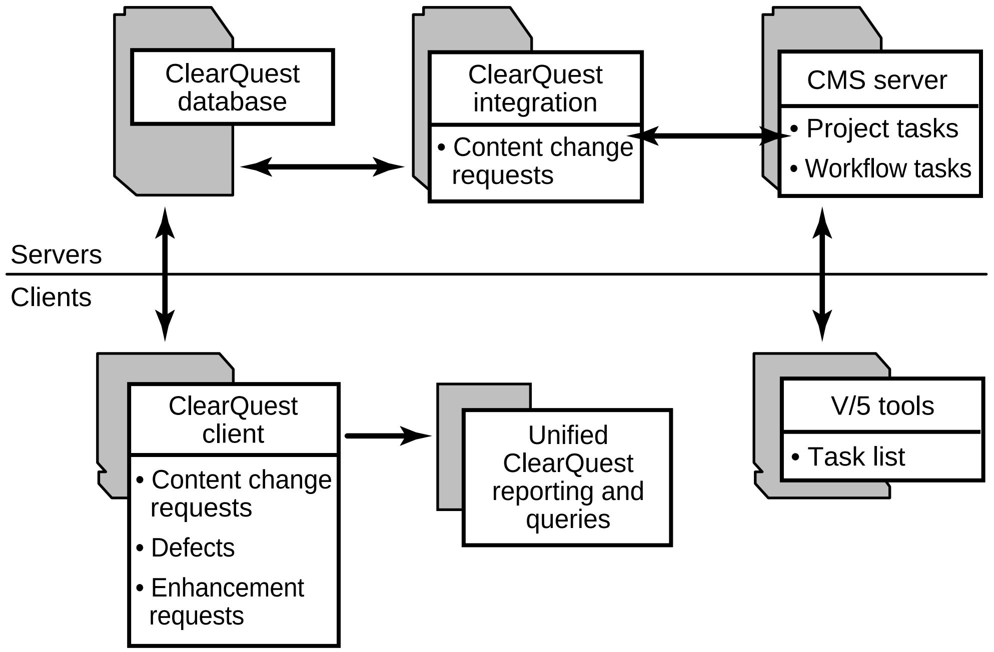
This chapter describes the Rational ClearQuest integration with the V/5 task list and a ContentChangeRequest record. It also explains how to use these records as children of other ClearQuest record types.

3.1 Overview of the ClearQuest-V/5 Integration

Rational ClearQuest allows you to manage requests for changes to code and content in a unified fashion. During staging and testing, there are two usage model options for using ClearQuest to assign change requests to content contributors. You can submit and assign standalone ContentChangeRequest records from ClearQuest and then use the ClearQuest integration with V/5 task list to track the state of these records. Or, you can associate one or more child records with a parent record type (such as a defect or enhancement request). The child records use the ClearQuest integration with the V/5 task list.

Figure 12 illustrates the interaction between ContentChangeRequest records that are submitted to ClearQuest with the V/5 project task list on the CMS server. To use these records as child records, you must define and configure parent record types at installation time. For more information on configuring parent record types, see *Installing Rational Suite ContentStudio*.

Figure 12 Content Change Model



Model of State Changes

In Figure 13, the solid lines indicate direct manipulation of state changes and the broken lines represent the system synchronization. Synchronization begins when a ContentChangeRequest record is created in ClearQuest and assigned to a V/5 user.

Figure 13 ClearQuest Integration State Model

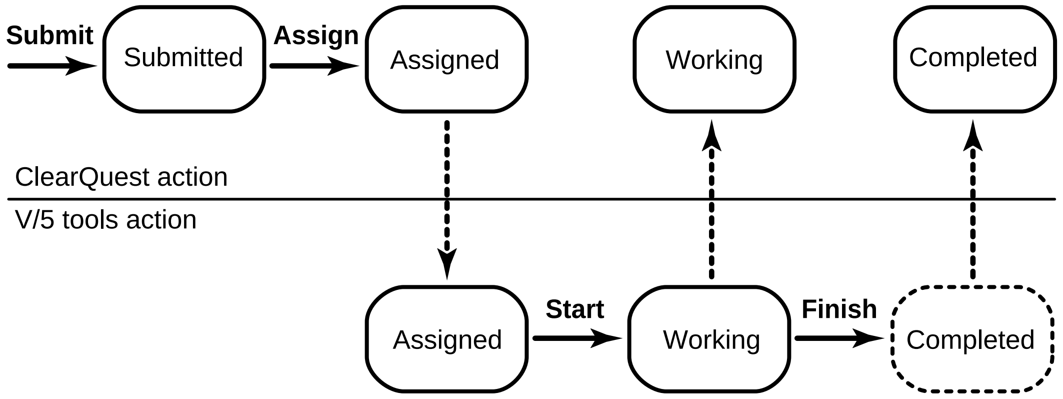
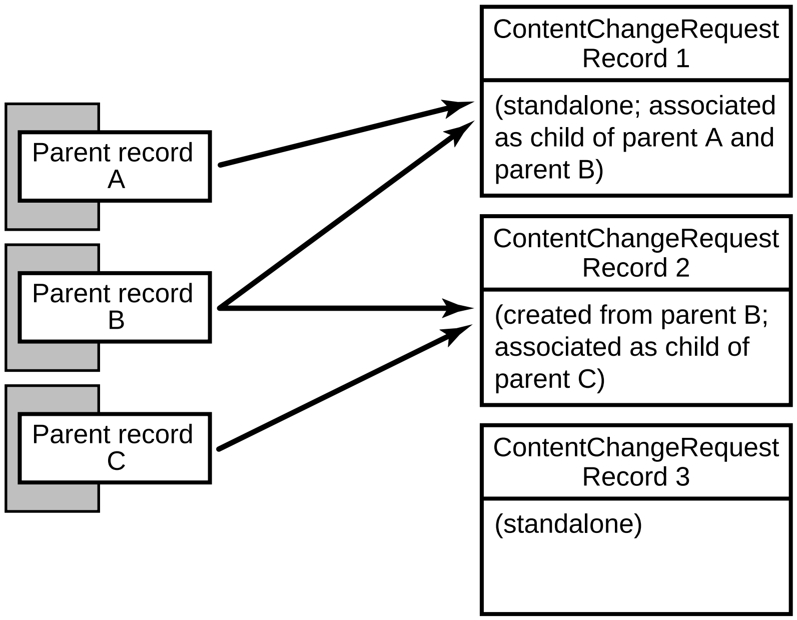


Figure 14 represents ContentChangeRequest records created using both the standalone and child models. After a ContentChangeRequest record is submitted, it can either exist independently (as in ContentChangeRequest Record 3), or it can be defined as a child to a parent record (as in ContentChangeRequest Record 1). When a ContentChangeRequest record is submitted as a child (as in ContentChangeRequest Record 2), it is associated with the parent record that submits it. This ContentChangeRequest record can also be defined as a child by other parent records (as in Parent record C).

Figure 14 ContentChangeRequest Records for the Standalone and Child Options


Parent record A
Parent record B
Parent record C
ContentChangeRequest Record 1
(standalone; associated as child of parent A and parent B)
ContentChangeRequest Record 2
(created from parent B; associated as child of parent C)
ContentChangeRequest Record 3
(standalone)

To use the ClearQuest integration, you must first install and configure these components:

- ContentStudio Package
- ContentStudio Client
- ClearQuest client
- V/5 platform tools

For complete installation and configuration instructions for the ClearQuest integration, see *Installing Rational Suite ContentStudio*.

ContentChangeRequest Records

ContentStudio provides a new ClearQuest record type, ContentChangeRequest, to manage changes to V/5 content records. When a ContentChangeRequest record is created and assigned in ClearQuest, it is mirrored in the V/5 project task list. Content contributors see these assignments as V/5 project tasks; they don't have to use ClearQuest directly.

For example, a tester finds an incomplete sentence in an HTML page after page generation. The tester uses ClearQuest client to submit a ContentChangeRequest record that describes the content problem and identifies the URL for that page. After the ContentChangeRequest record is assigned to a V/5 user, it appears as a project task on the V/5 task list. From the task list, the assigned content contributor can transition the task to a **Working** state, make the correction, and then transition the V/5 task to a **Complete** state. As the ContentChangeRequest record in the task list is transitioned to different states, the corresponding ContentChangeRequest record in ClearQuest is updated to reflect the current state in the V/5 task list.

ContentChangeRequests as Child Records

Using ClearQuest to manage workflow for software development, testers can submit defects and enhancement requests during the system test cycle. Software developers and template developers can use ClearQuest directly to record progress on the defects and enhancement requests that testers submit. (For more information on using ClearQuest as a defect tracking system for software development, see *Introducing Rational ClearQuest*.)

When using ClearQuest with ContentStudio, testers submit all problems encountered in staging as ClearQuest defects (or enhancement requests). When a tester creates a ClearQuest defect for an HTML page that is page generated from ContentStudio, a child ContentChangeRequest record can be created in the context of the parent defect (or enhancement request). The child record is mirrored in the V/5 task list. Content contributors can see these assignments as V/5 project tasks; they don’t have to use ClearQuest directly. The ClearQuest parent record (such as a defect or enhancement request) can be closed only when the associated child record is in a **Complete** state.

For example, a tester detects broken links in an HTML page after page generation from ContentStudio. The tester uses ClearQuest client to submit a defect that identifies the URL for this page. In ClearQuest, this defect is assigned to a template developer. Upon further examination of the URL, the tester finds a typographical error in the content of the HTML page. The tester then submits and assigns a child ContentChangeRequest record that is associated with the parent defect. This child record is mirrored in the V/5 task. A content contributor then sees the newly assigned record in the V/5 project task list. The content contributor transitions the V/5 task to a **Working** state, corrects the file, and then transitions the V/5 task to a **Complete** state. When the tester verifies that the parent defect for the broken link is also corrected, the defect can be closed. However, if the ContentChangeRequest record has not been closed previously in the V/5 project task list, the parent defect (or enhancement request) cannot be closed in ClearQuest.

3.2 Using the ClearQuest Integration

You can use the ClearQuest integration for a variety of tasks:

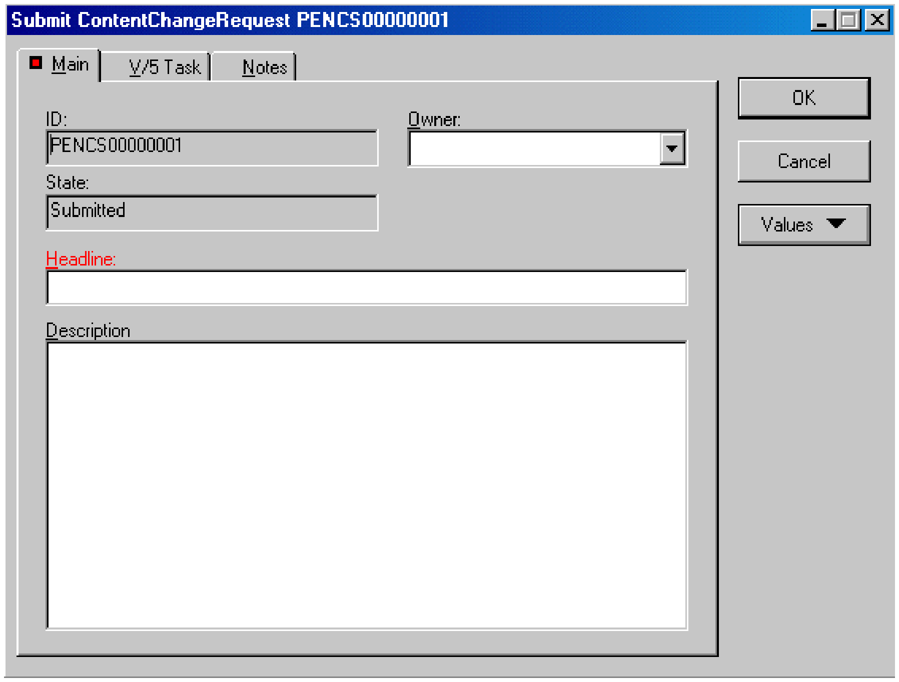
- ➤ Submit a New ContentChangeRequest record
- ➤ Assign a ContentChangeRequest record to a V/5 User
- ➤ Manage V/5 ContentChangeRequest records in a Task View
- ➤ Create Reports of Code and Content Change Requests
- ➤ Delete a Completed ContentChangeRequest record

Submit a New Content Change Request

After the ClearQuest integration has been installed and configured, you can create a new content change request:

1. On any host where the ClearQuest client is installed, start the ClearQuest client.
2. Click **Actions**>**New** to open the **Choose Record Type** dialog box. Click **ContentChangeRequest** and click **OK**.
3. In the dialog box (Figure 15), complete the **Headline**, **Owner**, and the **Description** boxes. For information about each box, right-click to display a shortcut menu and click **Help**.
4. Click **OK** to submit the ContentChangeRequest record to the ClearQuest database.

Figure 15 Submit ContentChangeRequest Dialog Box


Submit ContentChangeRequest PENCS00000001

ID: PENCS00000001

State: Submitted

Owner:

Headline:

Description

Assign a Content Change Request to a V/5 User

1. On any host where the ContentStudio Client and ClearQuest client are installed, start the ClearQuest client.

2. Run a query to select unassigned ContentChangeRequest records. (For more information on running queries, see *Introducing Rational ClearQuest*.)

   NOTE: The ContentStudio Package includes a **V5 Tasks - Unassigned** query in the **CS User Queries** folder. This query retrieves all unassigned V/5 Tasks for ContentChangeRequest records that the current user owns or that have no assigned owner.

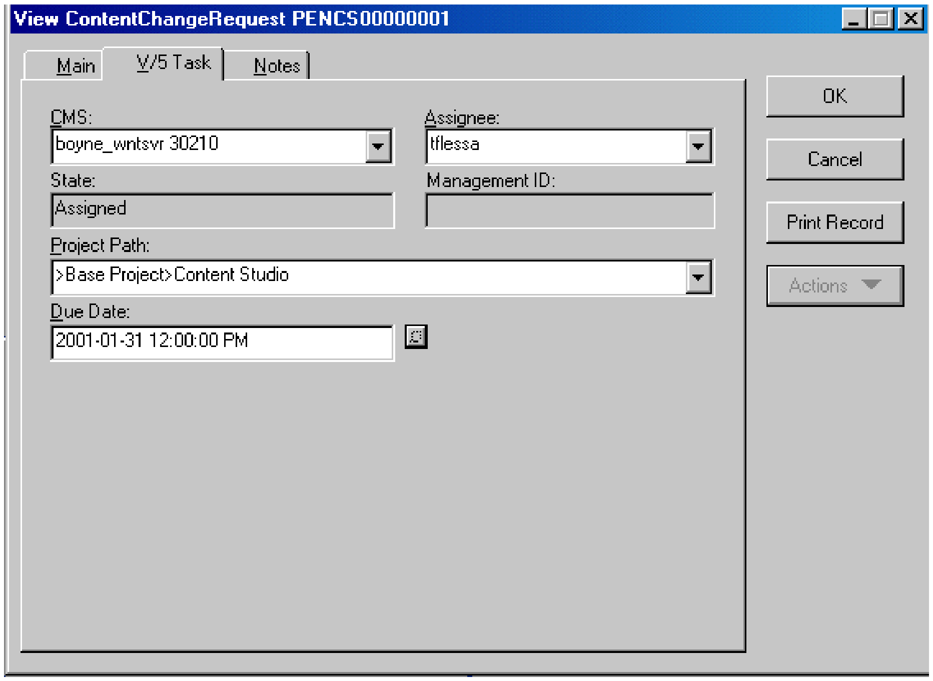
3. Select the ContentChangeRequest record that you want to assign and click **Actions**>**Assign**.

4. Complete the information on the **V/5** tab (Figure 16). For information about each field, right-click to display the shortcut menu and click **Help**.

- ➢ To select the name of your CMS server, click **CMS**.
- ➢ To select a V/5 owner, click **Assignee**.
- ➢ Verify that the information in the **Project Path** box is correct. This is where the corresponding V/5 project task will be stored.
- ➢ Optional. To select a completion date, click **Due Date**. This date is mirrored in the Vignette Task **Due Date** box.
- ➢ Optional. To provide additional information about the ContentChangeRequest record, click the **Notes** tab. This information is mirrored in the **Notes** tab in the **V/5 Task Details** dialog box (Figure 17).

5. Click **OK**.

Figure 16 The Assign ContentChangeRequest Dialog Box


View ContentChangeRequest PENCS00000001

CMS: boyne_wntsvr 30210

Assignee: tflessa

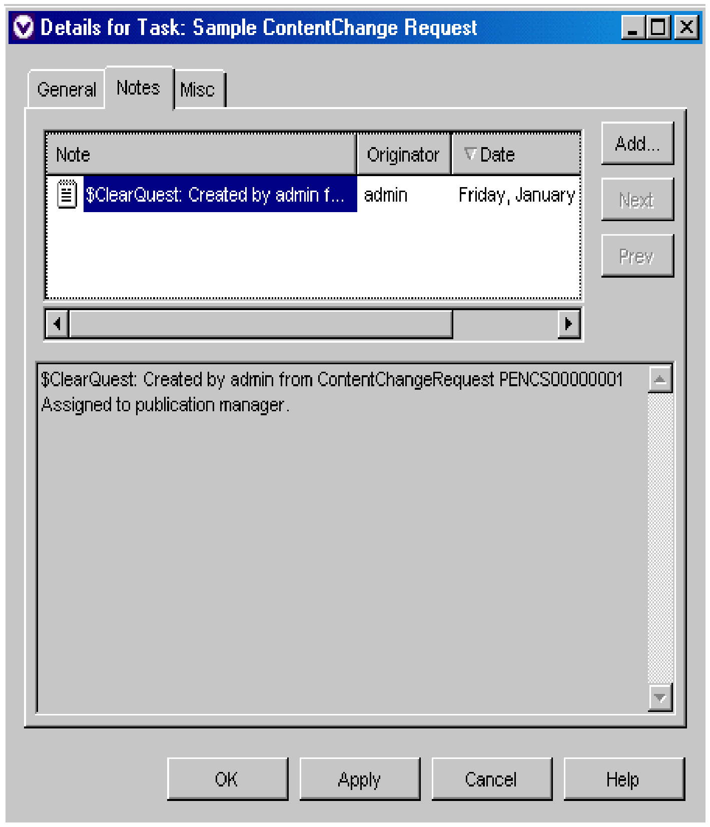
State: Assigned

Management ID:

Project Path: >Base Project>Content Studio

Due Date: 2001-01-31 12:00:00 PM

Figure 17   Notes Tab in V/5 Task Details



Note	Originator	Date
$ClearQuest: Created by admin f...	admin	Friday, January

$ClearQuest: Created by admin from ContentChangeRequest PENCS00000001
Assigned to publication manager.

Manage Content Change Requests in the V/5 Task View

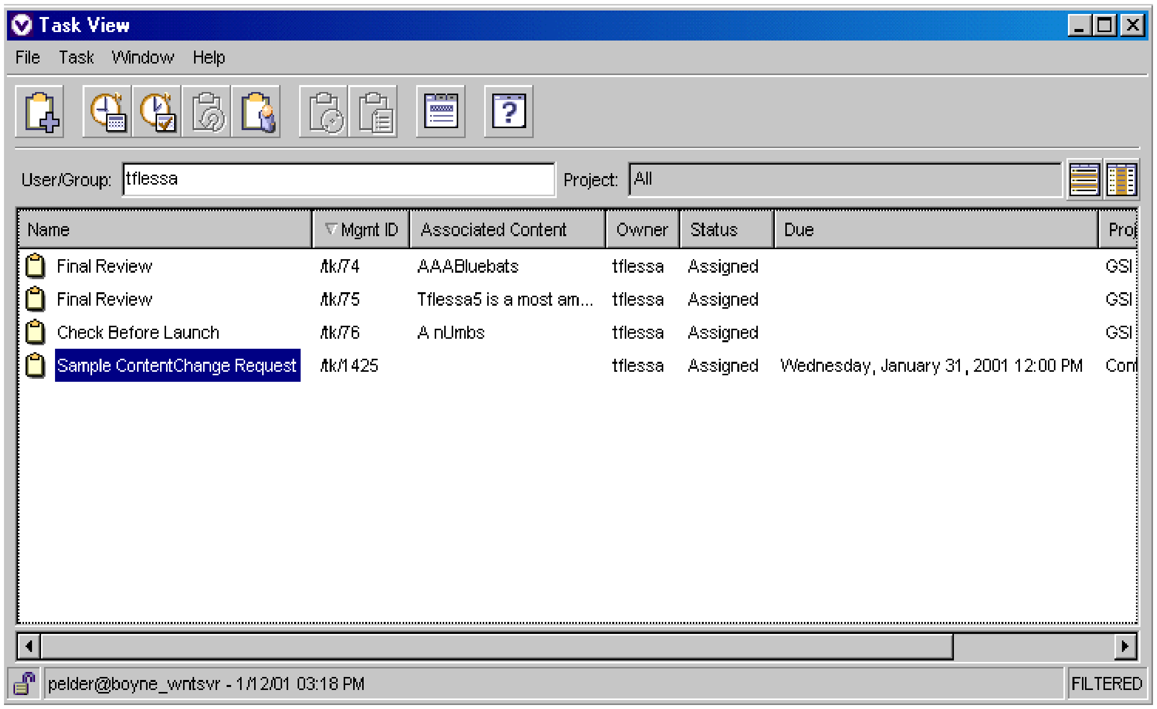
1. On a V/5 Platform Tools client, start V/5 Platform Tools and log in to the CMS.
2. Click **Window**>**Production Center**>**Task View** to open the Task View window (Figure 18).

A copy of the assigned ContentChangeRequest record is created in the V/5 task list. An authorized user working in the V/5 Task View window can change the state of a task list record from **Assigned** to **Working** to **Complete**.

As an item in the V/5 task list changes state, the state change information is mirrored in the associated ContentChangeRequest record in ClearQuest. For more information, see *Model of State Changes* on page 34.

For information on using the V/5 task list, see the Vignette V/5 manual *Production Center Guide*.

Figure 18 Task View Dialog Box



User/Group: tflessa
Project: All

Name	Mgmt ID	Associated Content	Owner	Status	Due	Proj
Final Review	/tk/74	AAABluebats	tflessa	Assigned		GSI
Final Review	/tk/75	Tflessa5 is a most am...	tflessa	Assigned		GSI
Check Before Launch	/tk/76	A nUmbs	tflessa	Assigned		GSI
Sample ContentChange Request	/tk/1425		tflessa	Assigned	Wednesday, January 31, 2001 12:00 PM	Con

pelder@boyne_wntsvr - 1/12/01 03:18 PM
FILTERED

Create Reports of Code and Content Change Requests

1. Start the ClearQuest client.

2. Create a query to generate the report. ClearQuest users who have **Super User** or **Schema Designer** rights can save queries in the **Public Queries** folder, where other ClearQuest users can access them. All ClearQuest users can create personal queries. For more information on creating queries, see *Introducing Rational ClearQuest*.

The ContentStudio Package provides the following to facilitate reporting in ClearQuest:

- A ClearQuest Record Family called **All_CS_Requests**.

  By default, this family includes one record: ContentChangeRequest. *Installing Rational Suite ContentStudio* contains instructions for adding a Defect and EnhancementRequest to this family. In addition, the ClearQuest administrator can revise the schema and add additional record types to the family. For more information, see *Administering Rational ClearQuest*. Creating queries and reports against **All_CS_Requests** allows you to monitor progress on all changes to code and content.

- The ContentStudio Package includes these public queries:

  - **V5 Tasks - Active**: Selects all ContentChangeRequest records that are in the **Assigned** or **Working** state and are either owned by the current user or have no owner.

  - **V5 Tasks - Late**: Selects all ContentChangeRequest records whose Due Date has passed that are in the **Assigned** or **Working** state and are either owned by the current user or have no owner.

  - **V5 Tasks - Unassigned**: Selects all ContentChangeRequest records in the **Submitted** state that are either owned by the current user or have no owner.

  - **All My CS Requests**: Selects all records in the **All_CS_Requests** record family that are either owned by the current user or have no owner.

Delete a ContentChangeRequest Record

To delete a ContentChangeRequestRequest record from the ClearQuest database:

1. On any host where the ClearQuest client is installed, start the ClearQuest client.

2. Find the ContentChangeRequest record. You can use use an existing query, create a new query by clicking **Query**>**New Query**, or enter a record ID by clicking **Edit**>**Find Record**.

3. Select the ContentChangeRequest record that you want to delete. If this record is in an **Assigned** or **Working** state, click **Action**>**Cancel**. This moves the record to a **Complete** state and terminates the associated Vignette Task.

4. In the ContentChangeRequest record form, click **Action**>**Delete** to remove the record from the ClearQuest database.

3.3 Enabling ContentChangeRequest Records as a Child Option

When using the ContentStudio Package, you have the option to use any records as parents of ContentChangeRequests records along with any ClearQuest schema.

To enable this option, you must map the state model used by the parent record type to a simplified state model understood by the ContentStudio Package. (This is called the ContentStudio Package state-type model in ClearQuest.) The state-type model for ContentChangeRequest records consists of three state-types:

➤ **TasksNotCreated**: The ContentChangeRequest record does not exist in the V/5 task list.

➤ **TasksInProgress**: The ContentChangeRequest record exists in the V/5 task list in an **Assigned** or **Working** state.

➤ **TasksComplete**: The ContentChangeRequest record exists in the V/5 task list in a **Complete** state.

The state-type mapping determines whether and when the ContentStudio Package performs automated actions and validations between the parent record and the child records.

When configuring the ContentStudio Package, a ClearQuest schema designer must select the parent record types and map the states for the parent records to one of the three state types supported by the ContentStudio Package. This mapping is the responsibility of the schema designer and depends on the states supported by a site's ClearQuest schema. For more information on configuring the ContentStudio Package, see *Installing Rational Suite ContentStudio*. To illustrate the mapping process, Figure 19 represents the mapping for all states supported by a defect in the ClearQuest Enterprise schema to the three state types supported by the ClearQuest package.

Figure 19 State to State-Type Mapping for the Enterprise Schema

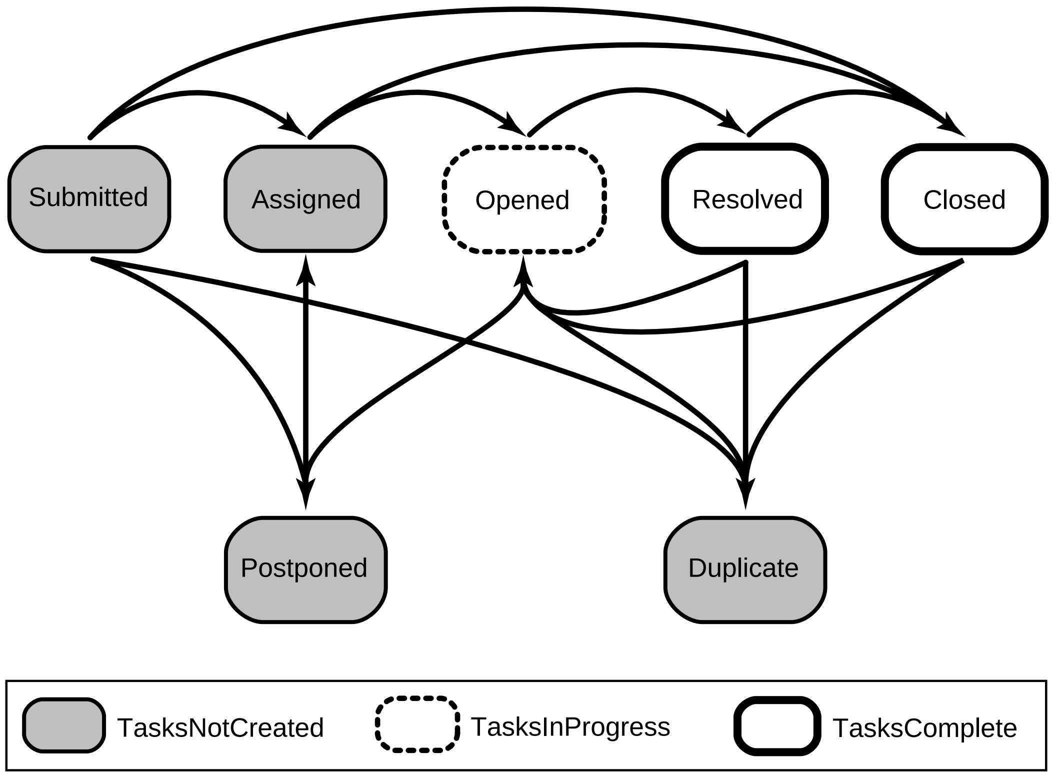


Table 3 maps the defect states (based on the Enterprise schema) to the three state types supported by ContentChangeRequest records.

Table 3 Example State to State-Type Mapping

Parent defect state	**Associated child ContentChangeRequest state**
Submitted	TasksNotCreated
Assigned	TasksNotCreated
Opened	TasksInProgress
Resolved	TasksComplete

Table 3 Example State to State-Type Mapping

Parent defect state	Associated child ContentChangeRequest state
Closed	TasksComplete
Postponed	TasksNotCreated
Duplicate	TasksNotCreated

Assuming that you define the state to state-type mapping as in Table 3, there is an automation for the ClearQuest integration when:

➤ A parent record type is transitioned from the **Assigned** to **Opened** state; all associated child ContentChangeRequest records are transitioned to an **Assigned** state, if not already in an **Assigned** state.

➤ A parent record type is transitioned from the **Opened** to **Resolved** state; all associated ContentChangeRequest records must be in a **Complete** state, or the transition of the parent record to **Resolved** will fail.

➤ A parent record type is transitioned to **Resolved** if the **Perform default action when all CCRs complete** option on the **Content Changes** tab is selected.

After the child ContentChangeRequest option has been configured for parent record types, the ClearQuest form for each parent record has a new tab for **Content Changes** as in Figure 17. Using the **Content Changes** tab, you can do the following:

➤ Associate the parent record with a new ContentChangeRequest record

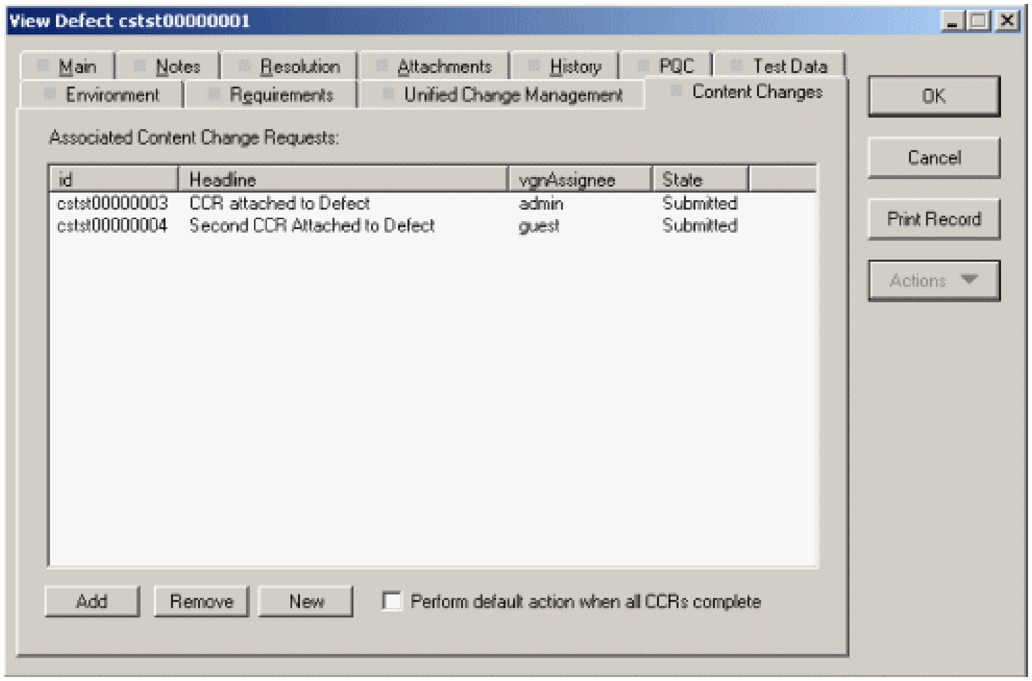
➤ Associate the parent record with an existing ContentChangeRequest record

➤ Remove an association for the parent record with a ContentChangeRequest record

➤ View ContentChangeRequest record states in the V/5 project task list

➤ View ContentChangeRequest record assignments

Figure 20 Content Changes Tab in the ClearQuest Defect Form



Associated Content Change Requests:

id	Headline	vgnAssignee	State
cstst00000003	CCR attached to Defect	admin	Submitted
cstst00000004	Second CCR Attached to Defect	guest	Submitted

If the ClearQuest integration with the child ContentChangeRequest option is enabled, you can do the following:

- Submit a defect (or other parent record type)
- Submit and assign a child ContentChangeRequest record
- Manage child ContentChangeRequests in the V/5 Task View
- Create reports of code and child Content Change Requests
- Remove a child ContentChangeRequest record
- Delete a child ContentChangeRequest record
- Resolve the defect

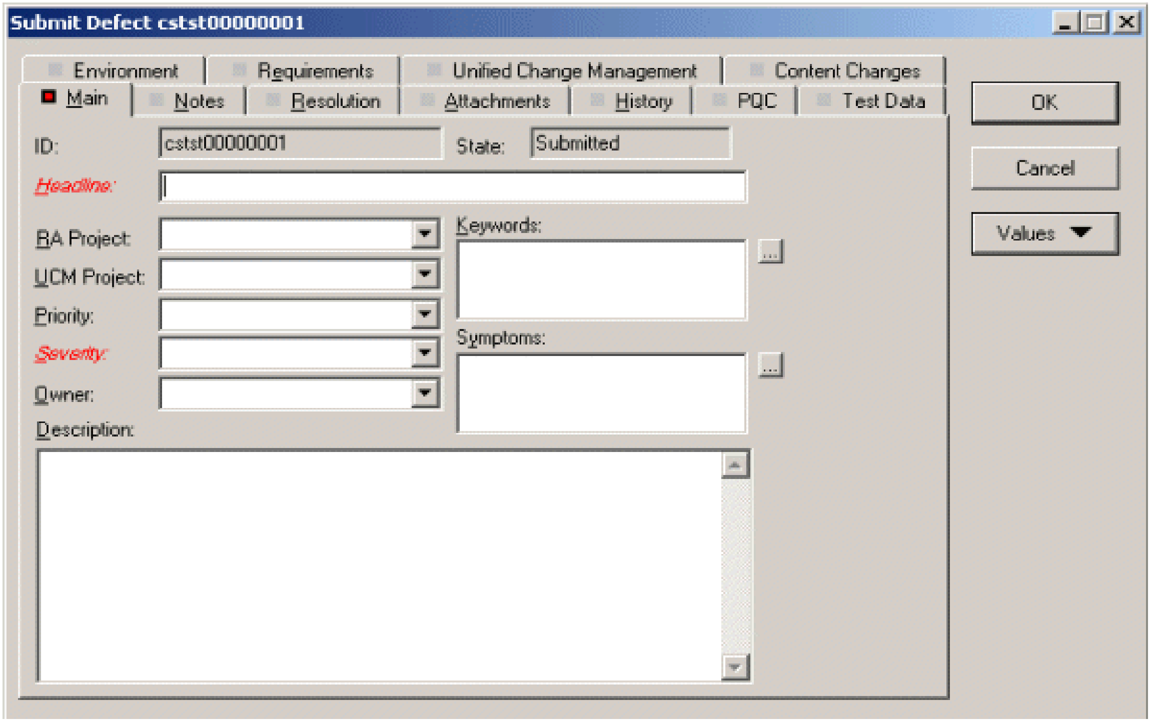
Submit a Defect

After the ClearQuest integration for child ContentChangeRequest records is installed and configured, you can submit a defect that is enabled for those child records.

1. On any host where the ClearQuest client is installed, start the ClearQuest client.
2. Click **Actions**>**New Defect** to open the **Submit Defect** dialog box as in Figure 25. Complete the required fields for the defect.
3. Click **OK** to submit the defect to the ClearQuest database.

   At the same time you submit a defect, you can associate ContentChangeRequest records. For more information on adding an association, see *Add a Child Association for a New or Existing Content Change Request* on page 49.

Figure 21 ClearQuest Defect Submission Form


Submit Defect cstst00000001

ID: cstst00000001
State: Submitted

Add a Child Association for a New or Existing Content Change Request

While submitting or after submitting a parent record type, you can:

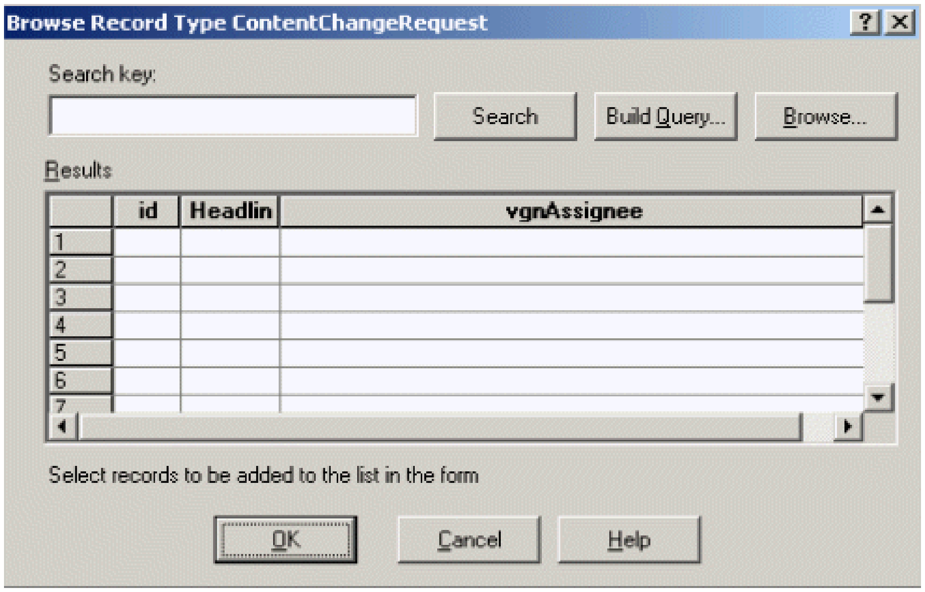
- Add a child association with an existing ContentChangeRequest record
- Add a child association for a new ContentChangeRequest record

To Add a Child Association for an Existing ContentChangeRequest Record

1. On any host where the ClearQuest client is installed, start the ClearQuest client.
2. Find a defect (or other parent record type) by using an existing query, creating a new query by clicking **Query**>**New Query**, or entering a record ID by clicking **Edit**>**Find Record**. Select a defect (or other parent record type) for which you want to add an existing child record.
3. Using the ClearQuest form for the selected defect, click the **Content Changes** tab.
4. Click **Actions**>**Modify**. Click **Add** to open the **Browse Record Type ContentChangeRequest** dialog box (see Figure 19) to create an association for an existing ContentChangeRequest record.
5. Using the **Browse Record Type ContentChangeRequest** dialog box, you have three options for locating existing ContentChangeRequest records to associate with the parent record:
   - Search by ContentChangeRequest record ID. Enter a record ID in the **Search key** box and click **Search**.
   - Use a predefined query. Click **Browse** to select an existing query.
   - Build a new query. Click **Build Query**.
6. After locating existing ContentChangeRequest records, from the **Browse Record Type ContentChangeRequest** dialog box, select one or more ContentChangeRequest records to add to the **Content Changes** tab.

   The selected ContentChangeRequest records are now associated with the parent record. The defect cannot be closed until all child records are in a **Completed** state.

Figure 22 Browse Record Type ContentChangeRequest Dialog Box


Browse Record Type ContentChangeRequest
Search key:
Search
Build Query...
Browse...
Results
id
Headlin
vgnAssignee
Select records to be added to the list in the form
OK
Cancel
Help

To add a Child Association for a New ContentChangeRequest Record

1. On any host where the ClearQuest client is installed, start the ClearQuest client.

2. Find a defect (or other parent record type). To do so, you can use an existing query, create a new query by clicking **Query**>**New Query**, or enter a record ID by clicking **Edit**>**Find Record**. Select a defect you want to add a child association of a new ContentChangeRequest record.

3. Using the ClearQuest form for the selected defect, click the **Content Changes** tab.

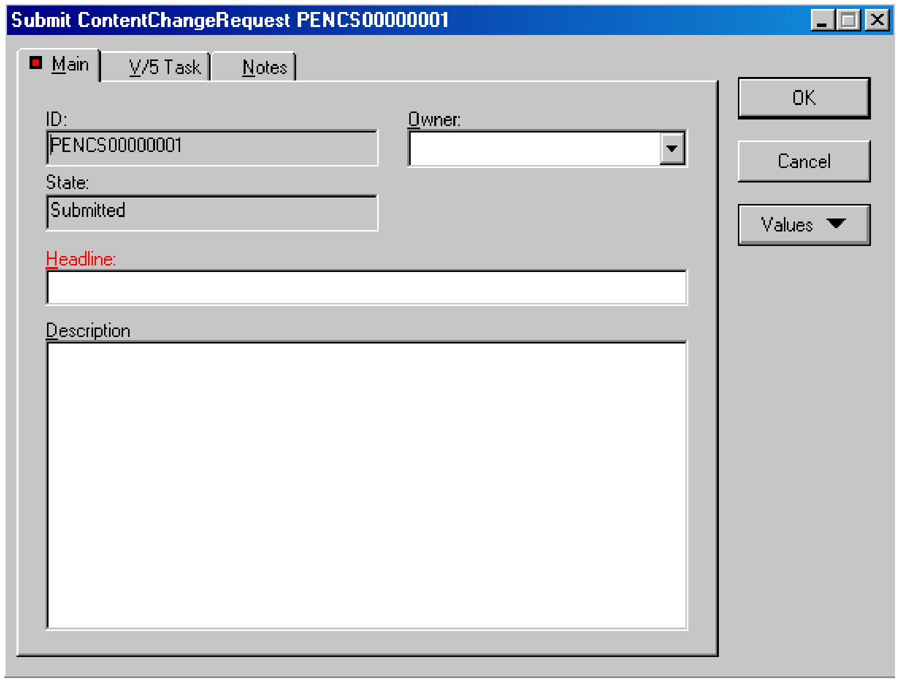
4. Click **Actions**>**Modify**, then click **New** to create an association for a new ContentChangeRequest record. The **ContentChangeRequest** dialog box opens (see Figure 23).

   At this point, the steps for submitting and assigning a new ContentChangeRequest record are identical to the standalone model for a ContentChangeRequest.

   For more information, see *Submit a New Content Change Request* on page 38 and *Assign a Content Change Request to a V/5 User* on page 39.

NOTE: During the submit action, you can complete the **V/5 Task** tab in the **ContentChangeRequest** dialog box so that the ContentStudio Package assigns the task (that is, creates the V/5 Task) when the parent record enters the **Opened** state (or any state mapped to the **TasksInProgress** state type). For more information on state to state-type mapping, see Table 3.

Figure 23 Submit ContentChangeRequest Dialog Box Accessed from Content Changes tab



ID: PENCS00000001

State: Submitted

Manage Child ContentChangeRequest Records in a Task View

After an association is created to a new or an existing ContentChangeRequest record, you can use the V/5 Task view to manage the ContentChangeRequest record in the same manner as the standalone model.

For more information on using V/5 Platform Tools to manage ContentChangeRequest records, see *Manage Content Change Requests in the V/5 Task View* on page 41.

Create Reports of Code and Child Content Change Requests

You can use ClearQuest to create reports for code and child ContentChangeRequest records in the same manner as the standalone model.

In addition to the queries for ContentChangeRequest records described in *Create Reports of Code and Content Change Requests* on page 42, the following queries are also available for reporting on child ContentChangeRequest records:

- **Parent CCRs with Tasks Submitted**: Finds all parent records that have submitted child ContentChangeRequest records.
- **Parent CCRs with Tasks Assigned or Working**: Finds all parent records that have submitted and assigned child ContentChangeRequest rcords.
- **Parent CCRs with Tasks Completed**: Finds all parent records that have completed child ContentChangeRequest records.

Remove a Child ContentChangeRequest Record

To remove a child ContentChangeRequest record:

1. On any host where the ClearQuest client is installed, start the ClearQuest client.
2. Find a defect (or other parent record type) by using an existing query, creating a new query by clicking **Query**>**New Query**, or entering a record ID by clicking **Edit**>**Find Record**. Select a defect that you want to remove an association with a child ContentChangeRequest record.
3. Using the ClearQuest form for the selected defect, click the **Content Changes** tab.
4. Click to select one or more ContentChangeRequest records on the **Content Changes** tab; click **Actions**>**Modify**, and then click **Remove**.

   NOTE: Clicking **Remove** only removes the association between the child record and the parent record; it does not remove or change the state of the ContentChangeRequest record in the ClearQuest database.

Delete a Child ContentChangeRequest Record

To delete a child ContentChangeRequest record from the ClearQuest database:

1. On any host where the ClearQuest client is installed, start the ClearQuest client.
2. Find a defect (or other parent record type) by using an existing query, creating a new query by clicking **Query**>**New Query**, or entering a record ID by clicking **Edit**>**Find Record**. Select a defect (or other parent record type) where you want to delete an existing child ContentChangeRequest record.
3. Using the ClearQuest form for the selected defect, click the **Content Changes** tab.
4. Double-click to select the ContentChangeRequest record that you want to delete. If the ContentChangeRequest record is in an **Assigned** or **Working** state, click **Action**>**Cancel**. This moves the ContentChangeRequest to a **Complete** state and terminates the associated Vignette Task.
5. From the ContentChangeRequest record form, click **Action**>**Delete** to remove the ContentChangeRequest from the ClearQuest database.

Resolve the Defect

When using the child ContentChangeRequest record, you have two options for resolving the defect:

➤ Resolve the parent defect when child ContentChangeRequest records are in a **Completed** state.

  To resolve a parent record, click **Actions**>**Resolve**. If any child ContentChangeRequest record is not in a **Completed** state, then ClearQuest displays an error message.

➤ Enable the parent defect to resolve automatically when last ContentChangeRequest record associated with the parent completes.

  To do so, click the **Perform default action when all CCRs complete** option from the **Content Changes** tab. When the last ContentChangeRequest record reaches a **Completed** state, the automatic resolving of the parent record only succeeds if:

  ➢ **Resolve** is the default action for the **Open** state.

- All required fields for **Resolved** state have values.

4 Using Rational NetDeploy Triggers

Rational Suite ContentStudio supports the use of Rational NetDeploy triggers that run when executing deployment, redeploy, or rollback tasks. Rational NetDeploy triggers are user-defined routines for processing before or after a specified operation.

4.1 Overview

Rational NetDeploy triggers support processing by user-defined routines. In general, when using NetDeploy triggers with deployment, redeploy, or rollback tasks, you can design trigger routines to:

- ➤ Perform notification functions during the course of a task
- ➤ Perform housekeeping activities on a Web server on which a Rational NetDeploy agent is installed
- ➤ Perform operating system functions on a Web server on which a Rational NetDeploy agent is installed

Rational NetDeploy triggers enable you to design a precise set of pre-operation or post-operation processing actions that can be executed either on Rational NetDeploy agents or the ContentStudio server.

NOTE: Rational NetDeploy triggers are not based on the ClearCase trigger mechanism.

Rational NetDeploy triggers are user-defined actions that can execute using any of the following:

- A DOS or UNIX command.
- A batch or shell script.
- Any executable on either the ContentStudio server or a Web server on which a Rational NetDeploy agent is installed.

Rational Suite ContentStudio ships with a set of sample triggers that perform the following actions:

- Register a COM object on a Rational NetDeploy agent
- After a deployment task completes, send an e-mail message from the ContentStudio server

After you install the Rational Suite ContentStudio Server or Client, the sample triggers are located in <*drive*>**:\Program Files\Rational\ContentStudio\Samples\NetDeploy Triggers**.

In addition to the sample triggers, you can develop user-defined triggers to perform pre-operation or post-operation actions such as:

- Before a specific type of file is deployed to a specific set of Rational NetDeploy agents, execute a script on the Web server for a user-defined action
- After a deployment task completes, stop each target Web server, run an update procedure, and then restart the target Web server.
- After a rollback task completes, send an electronic page to a site manager.

If a trigger action is to run on specific Web servers where a Rational NetDeploy agent resides, the user-defined routine must reside on the target Web servers before a trigger-enabled task can run and use the trigger.

4.2 Defining a Trigger

A NetDeploy trigger is defined using the Rational Suite ContentStudio Console. To define a trigger, you must specify:

- A unique trigger name
- A trigger description (optional)

- The trigger type
- The trigger firing conditions (which depend on the trigger type)
- The command or location of the script that supports trigger action
- The action to take if the trigger fails

For more information on using the ContentStudio Console to define a trigger, click **Rational NetDeploy**, click the **Triggers** tab, and then click **Help** > **Deployment Trigger Help**.

Defining Trigger Types

Rational NetDeploy supports different kinds of triggers for different pre-operation and post-operation functions. Rational NetDeploy supports four trigger types:

Task trigger

A task trigger executes an action for all ContentStudio tasks that are enabled for triggers. A trigger action associated with a task trigger type runs only on the ContentStudio server. For example, you can use a task trigger type to send an e-mail message at the completion of a deployment task.

Host trigger

A host trigger executes an action for each host in a list. For example, you could use a host trigger type to perform a specific action every time that a task is executed on a Rational NetDeploy agent where the Web server is running UNIX.

File trigger

A file trigger executes an action for each host, path, file type, or operation type (such as, create file, update file, or delete file) in a list. The firing conditions for a trigger action associated with a file trigger type are evaluted one time for each file operation in the task. For example, you can use a file trigger type to create a log of all **.com** file names that are deployed.

Directory trigger

A directory trigger executes an action for each host, path, or operation type (such as, create directory or delete directory) in a list. The firing conditions for a trigger action associated with a directory trigger type are evaluted, one time for each directory operation in the task. For example, you can use a directory trigger type to send an e-mail message to the product manager whenever a product information directory is updated.

Defining Trigger Firing Conditions

After a trigger type is defined, you must specify the firing conditions for the trigger. There are three parameters to specify when defining the firing conditions for a NetDeploy trigger.

Where

The task trigger type can run only on the ContentStudio server. For all other trigger types, you must define whether the trigger will run on the ContentStudio server or the Rational NetDeploy agent.

When

You must specify if a trigger action will execute pre-operation or post-operation; the operation is a deployment, redeploy, or rollback task.

Filter

Depending on what the trigger type is, you can specify the filters (firing conditions) in Table 4:

Table 4 Trigger Types and Filters

Trigger type	**Filters**
Task	None.
Host	You can specify the host names for Rational NetDeploy agents as a firing condition for executing a trigger action on the ContentStudio server or the Rational NetDeploy agent.

Table 4 Trigger Types and Filters

Trigger type	**Filters**
File	You can specify any of the following as firing conditions for executing a trigger action on the ContentStudio server or Rational NetDeploy agent locations: ➤ Host names for Rational NetDeploy agents ➤ Paths at the Rational NetDeploy agent locations ➤ Extensions for files sent to the Rational NetDeploy agent locations ➤ Operations (create file, update file, or delete file) at the Rational NetDeploy agent locations
Directory	You can specify any of the following as firing conditions for executing a trigger action on the ContentStudio server or Rational NetDeploy agent locations: ➤ Host names for Rational NetDeploy agents ➤ Paths at the Rational NetDeploy agent locations ➤ Operations (create directory or delete directory) at the Rational NetDeploy agent locations

NOTE: When you define a host, file, or directory trigger type, if you specify all hosts for the value of **Hosts** for Rational NetDeploy agent locations, the evaluation for all possible hosts occurs at run time for the deployment, redeploy, or rollback task.

Defining the Trigger Action

There are two dimensions to define for a trigger action:

➤ Command or script for Windows or UNIX

➤ Options for a trigger action failure

NOTE: When a trigger command or script executes, the trigger command or trigger script will execute as the user identity of the servlet engine running on the ContentStudio server or Rational NetDeploy agent. For more information on configuring a servlet engine for the ContentStudio server or a Rational NetDeploy agent, see *Installing Rational Suite ContentStudio.*

Command or Script for Windows or UNIX

You must specify the command or script location to support the trigger action.

➤ If you have specified a **Where** parameter of ContentStudio server, you must enter the location of the script or command to be executed on the ContentStudio server host running a Windows operating system.

➤ If you have specified a **Where** parameter of Rational NetDeploy agent, you must enter the location of the script or command to be executed on a Rational NetDeploy agent host running either a Windows or UNIX operating system.

If you specify a command for execution on the ContentStudio server or Rational NetDeploy agent, the command may contain arguments. ContentStudio supports three variables:

➤ *%p*

➤ *%s*

➤ *%u*

The *%p* and *%s* variables represent the pathname of an operation. For example, when this trigger action executes, the *%p* variable is

replaced by a pathname like **webfile1.dat**.

```
trigger_10.bat %p
```

Use the *%s* variable if you are using shadow directories, and you want to perform a trigger action based on the name of a file in a shadow directory. For more information on using the shadow directory mechanism in a deployment, redeploy, or rollback, see Section 2.5, Deployment.

If you are using shadow directories with a task that has an associated trigger action, the execution of the trigger action may require that a deployed file be at its final destination, and not the shadow directory destination. For more information on using a NetDeploy trigger with shadow directories, refer to **README** at <*drive*>**:\Program Files\Rational\ContentStudio\Samples\NetDeploy Triggers**.

The *%u* variable represents the name of the task and a date. For example, if a trigger action executes using the *%u* variable, it will be replaced by a task name and date like **fred-07-May-2001.12.38.25**. If a trigger action is performing a logging function, you can use this variable to create a unique name for the log.

Location of NetDeploy Trigger Scripts

If you specify a user-defined script, you must define the location of the script on the ContentStudio server or Rational NetDeploy agent either by using a full path, or by adding the script directory path to a **PATH** environment variable.

NOTE: User-defined scripts only run in batch mode; trigger scripts requiring user interaction at trigger execution are not supported.

Trigger scripts must previously reside at the appropriate location (ContentStudio server or Rational NetDeploy agent). You cannot use a deployment task to deploy the trigger scripts, and then run that trigger script as part of the same deployment task. If you want to use ContentStudio to place the trigger scripts at remote Rational NetDeploy agent locations, you must first create a deployment task to deploy the trigger scripts; then, you can run a deployment task associated with the trigger action.

Options for a Trigger Action Failure

Rational NetDeploy can simultaneously start deployment task threads to multiple Web servers (where Rational NetDeploy agents reside. These multiple deployment threads run asynchronously. When defining a trigger, you must specify how Rational NetDeploy will handle a trigger action failure that can occur on any of the deployment threads. You can either:

➤ Continue the deployment threads to any Rational NetDeploy agent targets where an error occurred

➤ Or cancel the deployment threads to any Rational NetDeploy agent targets where an error occurred

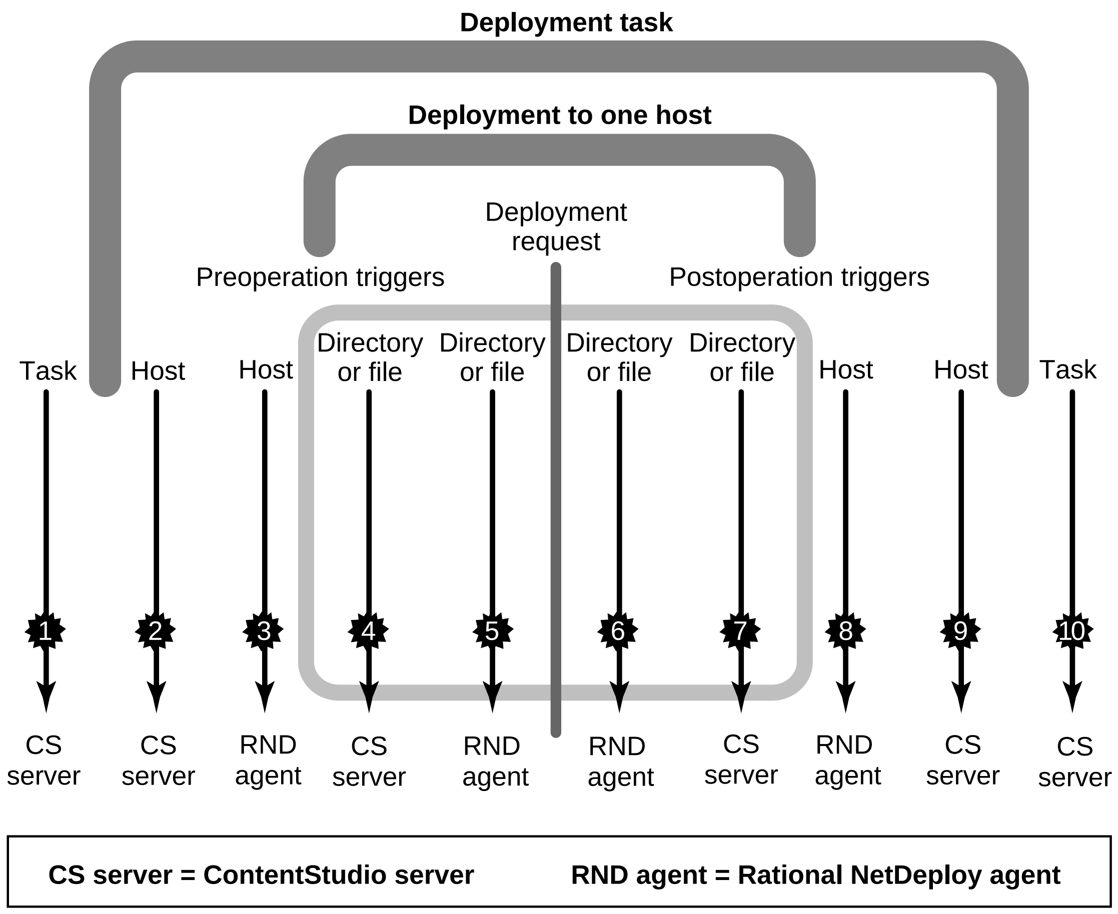
When you design user-defined routines, you must code error-handling logic to return a non zero value. If you do not include error-handling logic in user-defined routines, the **Continue** and **Cancel** failure actions will have no effect because ContentStudio server cannot determine that there has been a trigger execution failure. Conversely, any user-defined routine that returns a non zero value will be evaluated as having failed. If your trigger script needs to return a value for some other reason than to indicate a failure, you must return a value of zero.

NOTE: Trigger actions run synchronously; that is, a trigger action must successfully complete before the next deployment operation can occur. A task will wait indefinitely for a trigger action to successfully complete or indicate failure through a non zero return value.

NetDeploy Triggers Usage Model

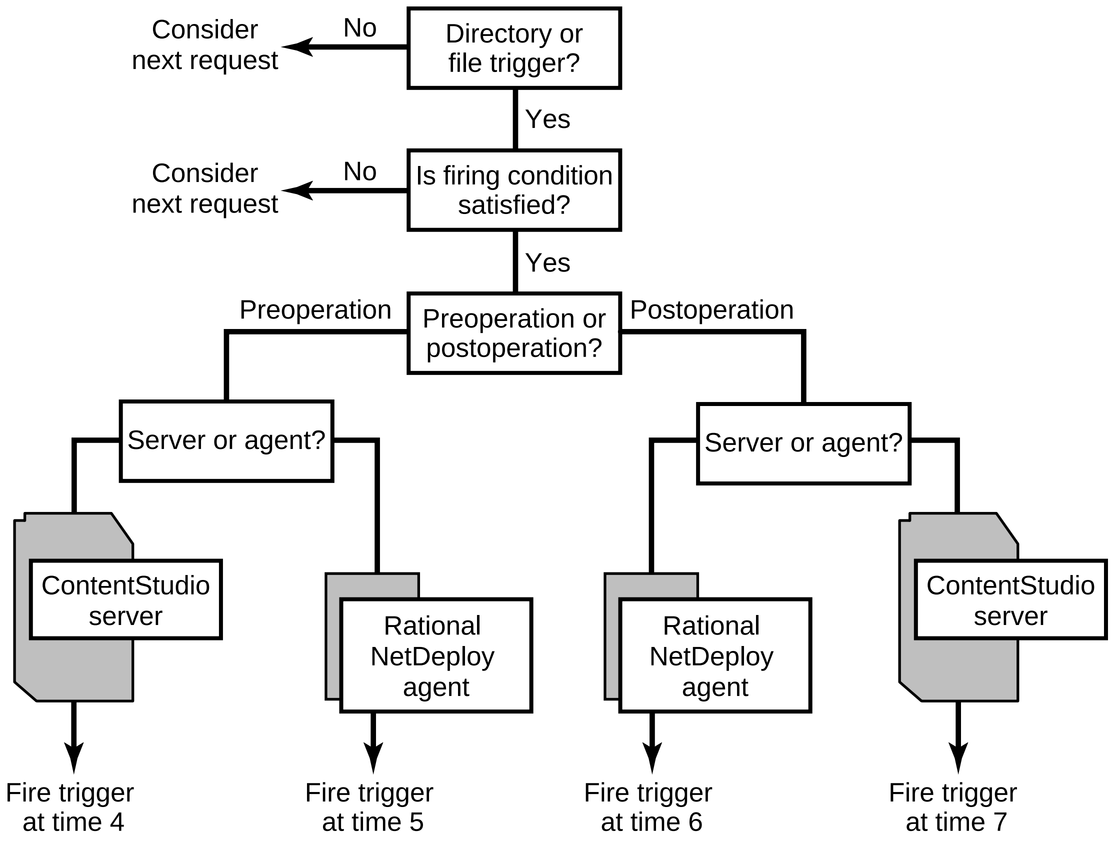
Figure 24 demonstrates the general usage model for working with Rational NetDeploy triggers. Before using ContentStudio Console to define a trigger, you must decide exactly which actions you will need to code. Figure 24 defines the precedence of pre-operation and post-operation trigger actions in the context of a deployment, redeploy, or rollback task. By specifying the trigger type (task, host, file, or directory), where (ContentStudio server or Rational NetDeploy agent), and when (pre-operation or post-operation), you can designate the point in the task processing sequence that a trigger action fires.

Figure 24 Trigger Sequence Model



If you specify a file or directory trigger type, when the task executes, the trigger can fire either pre-operation or post-operation, on the ContentStudio server or Rational NetDeploy agent, for each file or directory deploy operation. For example, Figure 25 represents the detail for the shaded area in Figure 24. Figure 25 describes the processing flow for evaluating each file or directory operation in a deployment, redeploy, or rollback task.

Figure 25 Processing Flow When Using a File or Directory Trigger (Sequence 4-7 from Figure 24)



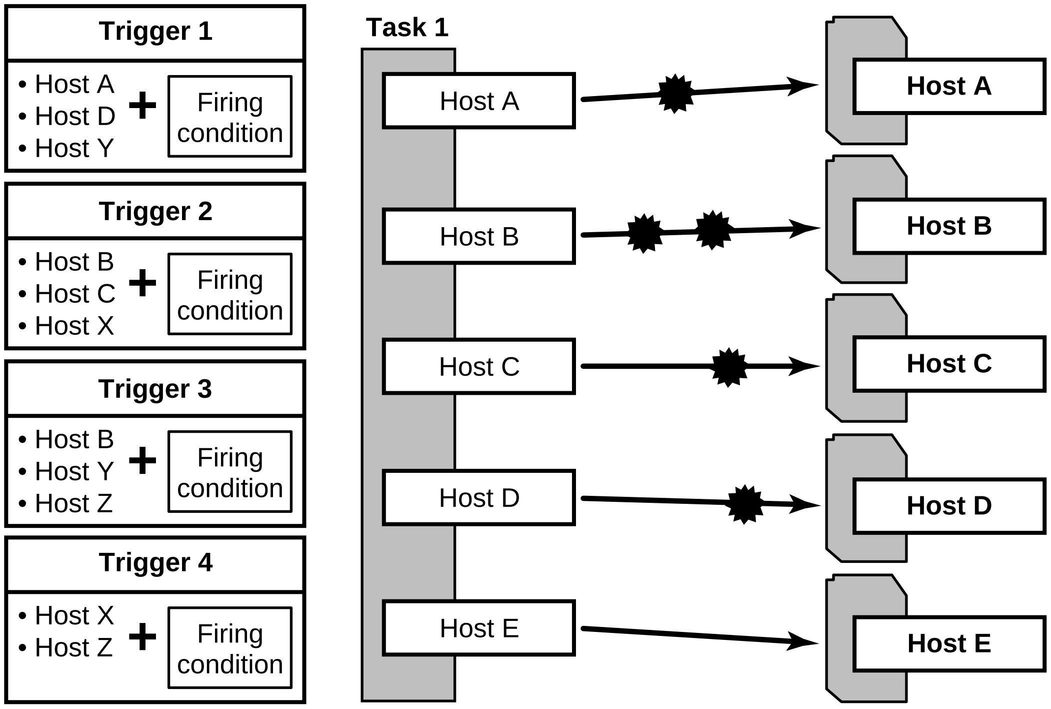
4.3 Behavior of Triggers

When you define a trigger, you must specify if the trigger is to run on the ContentStudio server or Rational NetDeploy agents. If a host, file, or directory trigger type is defined, the trigger definition must specify a **Hosts** parameter in the trigger firing conditions for the Rational NetDeploy agents where the trigger will execute.

Any deployment, redeploy, or rollback task can be enabled for triggers by clicking the **Yes** or **No** option for the **Use Triggers** field on the associated task definition dialog box. When a trigger-enabled task executes, ContentStudio determines if the **Hosts** parameter for the trigger definition matches any of the Rational NetDeploy agent locations that are defined for the task. If there is a match for the **Hosts** parameter, a trigger action will execute, on either the ContentStudio server or matching Rational NetDeploy agents, based on the specified firing conditions.

For example, Figure 26 demonstrates that for the deployment task definition of Task 1, the deployment operation will send files to Rational NetDeploy agents on Hosts A,B, C, D, and E. There are trigger definitions in the ContentStudio database for Triggers 1, 2, 3, and 4. If the deployment task is enabled for triggers, at the time the task executes, the deployment task will execute the trigger action for Triggers 1, 2, and 3 on Rational NetDeploy agents for Hosts A, B, C, and D. The triggers will fire on the deployment task thread for Hosts A, B, C, and D because there are three trigger definitions in the ContentStudio database that have a **Hosts** parameter that matches the Rational NetDeploy agent locations that are defined for Task 1. When Task 1 executes, no triggers will execute on Host E because there is not a current trigger definition in the ContentStudio database that has a Hosts parameter of Host E. In addition, when the deployment thread to Host B executes, two triggers will fire because there are two trigger definitions (Trigger 2 and Trigger 3) that define a trigger action for Host B.

Figure 26 Associating Trigger Definitions with a Deployment Task



Using ContentStudio Console, you can edit or delete trigger definitions. If a trigger definition is deleted, the trigger definition is removed from the ContentStudio database. Any pending tasks that are enabled for triggers will execute using only existing trigger definitions in the ContentStudio database. If you edit a trigger definition, any recurring deployment tasks associated with that trigger definition will use the edited version of the trigger definition.

Logging Trigger Actions

Logging for trigger events is in the associated deployment, redeploy, or rollback task logs that are accessible from the ContentStudio Console. Table 3 describes the location for NetDeploy logging information.

Table 5 Trigger Log Message Types.

Log message type	**Location**
Rational NetDeploy agent action (including action **STDOUT** or **STDERR**)	Task log on ContentStudio server (if all actions for the operation finish)
ContentStudio server action (including action **STDOUT** or **STDERR**)	Task log on ContentStudio server (regardless of whether the action terminates)
Task trigger actions	Task log on ContentStudio server (regardless of whether the action terminates)

NOTE: Any output from a trigger executing on a Rational NetDeploy agent must be returned to the ContentStudio server; therefore, no trigger information is in a task log until after a response returns from a Rational NetDeploy agent to the ContentStudio server. In other words, if a trigger action hangs on a specific Rational NetDeploy agent, you will not see a task log message to indicate that the trigger is executing on that specific Rational NetDeploy agent. In this case, you will only see the task remain in the executing queue for an extended period of time. To correct this, you must abort the task in the executing queue to clear the queue for any other pending deployment tasks still waiting to execute.

For more information on accessing task logs, from the ContentStudio Console, click **Task Schedule**, click the **Completed** tab, and then click **Help** > **Completed Tasks Help**.

Combining Triggers to Define Multiple Actions

To more efficiently administer control Web server environments when you deploy, redeploy, or rollback, you can use a combination of triggers to perform a sequence of actions, such as:

1. Stop the web server before beginning a deployment. (When the web server is stopped, to run the deployment task, a second Web server must be configured to use the Rational NetDeploy agent.)
2. Deploy files defined in a task.
3. Register all **.com** objects on the Web server.

4. Restart the Web server.
5. Send e-mail notification of successful completion to the site manager.

To complete these processing steps on single or multiple Rational NetDeploy agent locations, you must define the following triggers:

1. Host trigger to run as a pre-operation to stop the Web server
2. File trigger to run as a post-operation for all **.com** files to register **.com** objects on the Web server
3. Host trigger to run as a post-operation to restart the Web server
4. Task trigger to run as a post-operation on ContentStudio server to send an e-mail message to the site manager

If there is more than one pre-operation or post-operation trigger action for an operation, ContentStudio does not control the order of execution. If you want to perform more than one action in a certain order, you can do one of the following:

➤ Place all trigger actions in a single script.

➤ For a set of simple Windows NT or Windows 2000 commands, list them all when defining the trigger action, separated by **&&**. For example:

```
net stop "World Wide Web Publishing Service" && net start "World Wide Web
Publishing Service"
```

Setting Up Triggers

In general, to define triggers with Rational NetDeploy, you will need to

1. Design the trigger and write code.
2. Copy user-defined trigger scripts to predetermined locations on hosts where the trigger action will execute.
3. Debug the trigger script.
4. Redeploy final trigger scripts.
5. Define deployment, redeploy, or rollback tasks enabled for new triggers.

Design the Trigger and Write Code

You can use operating system commands or user-defined scripts to support trigger actions.

Deploy Code to a Rational NetDeploy Agent Testing Location

You will need to deploy your test code to a Rational NetDeploy agent testing location.

Debug the Trigger Script

Designing a set of trigger actions to work in a synchronized manner is a complex task. We recommend that you configure a Rational NetDeploy agent for testing your trigger actions. After executing a test deployment task associated with your triggers:

- ➤ Check the deployment task log for any possible trigger execution error messages. In addition, the deployment task log, accessible from the ContentStudio Console, displays log information for a trigger **STDOUT** or **STDERR** action. You can use this logging feature when debugging user-defined trigger routines.
- ➤ Check the Web server environment where aRational NetDeploy agent resides to verify that trigger script processing produces the results that you expect.

Redeploy Final Trigger Scripts

When your trigger script is tested and you have made all necessary corrections, you can deploy or redeploy this trigger script to the appropriate Rational NetDeploy agent locations.

Define Deployment, Redeploy, Rollback Task Enabled for New Triggers

Using ContentStudio Console, you can do either of the following:

- ➤ Define a new task with triggers enabled.
- ➤ Use an existing task and enable the task for triggers.

5 Using the ClearCase/CMS Integration

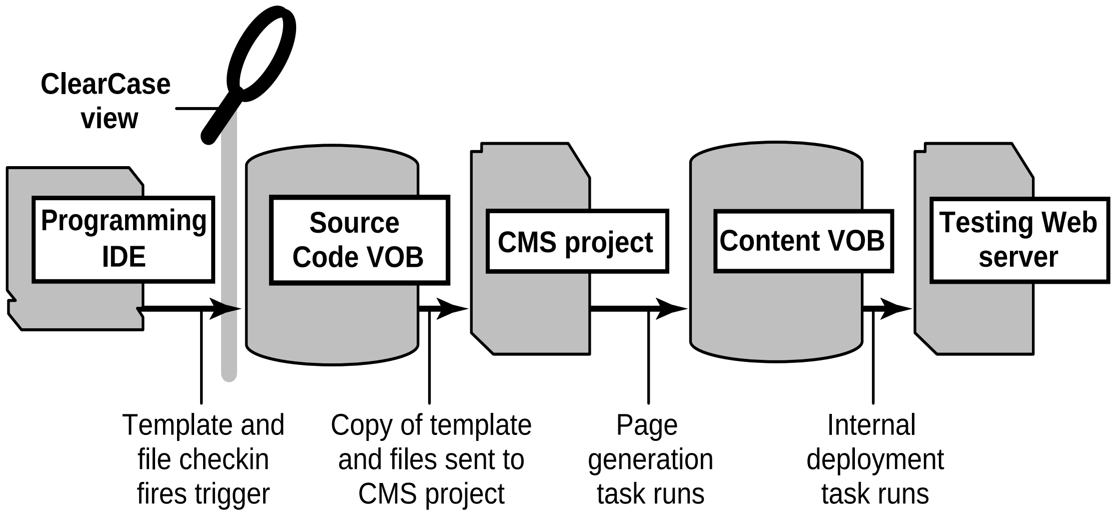
Rational Suite ContentStudio supports an integration between ClearCase and the Vignette CMS for disk-based template development. The ClearCase/CMS integration enables a template developer to use a Programming IDE for template development, while using ClearCase to source control a template (ASP or JSP) or a file (such as **.gif** or **.jpeg**).

5.1 Overview

Using the ClearCase/CMS integration, you can develop Vignette templates in source code VOBs using a ClearCase UCM or non-UCM work model. Figure 27 demonstrates that when a template, residing in a source code VOB, is checked in from a ClearCase view (or a Programming IDE using the view), the ClearCase/CMS integration imports the latest version of the template file into a Vignette CMS project. After a template is imported, the template developer can test the template by previewing the template in the development CDS. When testing of the template is completed, the template is launched using V/5 tools. Then Rational Suite ContentStudio Console is used to create an asset list and define a page generation task to generate the associated HTML pages in the content VOB.

Quality Assurance can test generated HTML pages in the content VOB or use ContentStudio Console to deploy the HTML pages in the content VOB to a staging server. As Quality Assurance detects problems during system testing, defects can be submitted using ClearQuest. For more information on using the ClearQuest integration in ContentStudio, see Chapter 3.

Figure 27 ClearCase / CMS Integration


ClearCase view
Programming IDE
Source Code VOB
CMS project
Content VOB
Testing Web server
Template and file checkin fires trigger
Copy of template and files sent to CMS project
Page generation task runs
Internal deployment task runs

For more information on using ContentStudio Console for asset list management and page generation, see Chapter 2, *Page Generation*. For more information on using ContentStudio Console to deploy HTML pages from the content VOB to a staging server, see Chapter 2, *Deployment*.

5.2 Getting Started Using the ClearCase/CMS Integration

To start using the ClearCase / CMS integration, you must

➤ Set up one or more ClearCase code VOBs for either a UCM or non-UCM work model

➤ Install ContentSudio Client and ContentStudio Server and set up the ClearCase / CMS integration triggers

➤ Set up the ClearCase / CMS integration map

➤ Plan the template directory hierarchy that will be imported into the CMS project hierarchy

➤ Decide upon which programming IDE to use for template development

Set Up a ClearCase Source Code VOB for UCM or non-UCM Work Model

After you have installed a ClearCase LT or a ClearCase server for your ContentStudio environment, you will need to create one or more source code VOBs (or use existing source code VOBs) for template development. For information on installing ClearCase LT or ClearCase, see *Installing Rational ClearCase LT* (for Windows) and *Rational ClearCase LT Installation Notes* (for UNIX) or the *ClearCase Product Family Installation Notes* (Windows and UNIX). For information on creating a code VOB, see *Adminstering ClearCase LT* or *Administering ClearCase*.

If you are using a UCM work model for template development with ClearCase, you will need to create a UCM project. For more information on creating and managing a UCM project, see *Managing Projects with ClearCase*.

Whether you use a UCM or non-UCM work model, you will also need to install ClearCase LT client or ClearCase client on all desktops where template developers are working. For more information on setting up clients for the ClearCase / CMS integration, see *Installing Rational Suite ContentStudio*.

After the ClearCase or ClearCase LT client is installed, if you are using a UCM work model, each template developer needs to join the UCM project. For more information on how developers can join UCM projects:

- ➤ If you are using ClearCase, see *Developing Software with UCM*.
- ➤ If you are using ClearCase LT, see the table of contents entry *Developing Software with UCM* in the ClearCase LT Help.

After the ClearCase client or ClearCase LT is installed, if you are using a non-UCM work model, developers must decide if they are working on the main project branch or from private branches. For more information on how developers can work:

- ➤ If you are using ClearCase, see *Developing Software with Base ClearCase*.
- ➤ If you are using ClearCase LT, see the table of contents entry *Developing Software with Base ClearCase* in ClearCase LT Help.

Install ContentStudio Client and ContentStudio Server and Set Up the ClearCase /CMS integration

To enable the ClearCase / CMS integration, you must install the ContentStudio Server and ContentStudio Client on different hosts. See *Installing Rational Suite ContentStudio* for complete installation instructions. After installing the ContentStudio Server and Client, you must set up the ClearCase / CMS integration on the ContentStudio server and client.

Setting Up ContentStudio Clients

After installing ContentStudio Client, create a local snapshot view to enable the ClearCase triggers that support the ClearCase / CMS integration. For further information on setting up the snapshot view for the ClearCase / CMS integration triggers, see *Installing Rational Suite ContentStudio*.

Setting Up the ContentStudio Server

After ContentStudio Server is installed, you must apply the ClearCase triggers to all the source code VOBs. For more information on setting up the ClearCase triggers for your source code VOBs, see *Installing Rational Suite ContentStudio*.

Set Up the ClearCase/CMS Integration Map

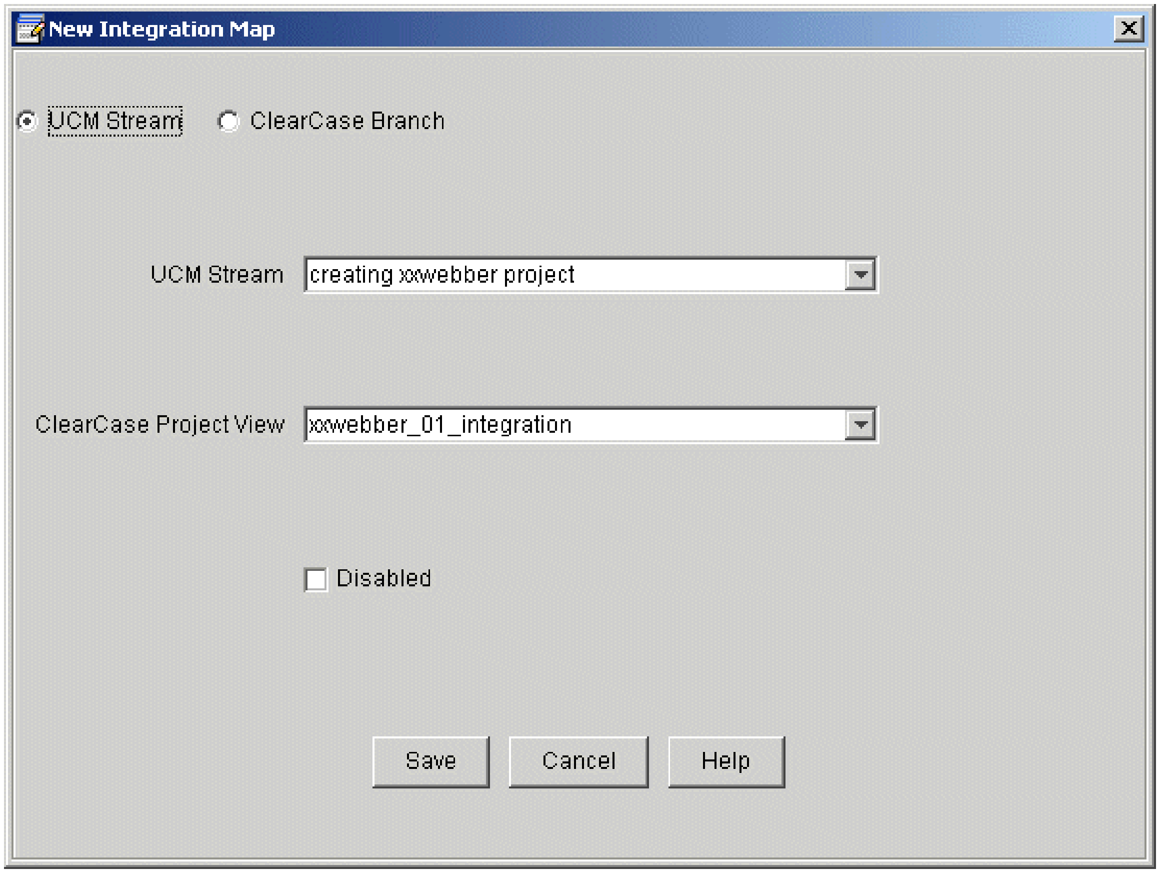
After applying the ClearCase triggers to the source code VOBs, you must use the ContentStudio Console to specify the ClearCase / CMS **Integration Map**. The **New Integration Map** dialog (see Figure 28) defines the main project stream or branch where development will occur and the CMS project view.

➤ For UCM-enabled ClearCase, the stream is an integration stream for a UCM project. The CMS project view is an associated UCM integration view whose config spec selects the specified integration stream.

➤ For non-UCM ClearCase, the branch is the main project branch in the source code VOB. The CMS project view is a ClearCase view whose config spec selects the specified main project branch.

The ClearCase / CMS integration stream or branch and CMS project view is used for evaluating supported ClearCase operations for the integration. For more information, see *ClearCase/CMS Integration Operations* on page 87 and *ClearCase/CMS Integration Usage Rules* **on page 85**.

Figure 28 ClearCase / CMS New Integration Map Dialog


New Integration Map

(● UCM Stream) ○ ClearCase Branch

UCM Stream: creating xxwebber project

ClearCase Project View: xxwebber_01_integration

☐ Disabled

To specify a ClearCase / CMS **Integration Map**:

1. Log in to ContentStudio Console.
2. Click **ClearCase/CMS Integration**. Click the **Admin** tab and then click **File>New Map**.
3. Click to select a **UCM Stream** or **ClearCase Branch**.
4. Click to select the **CMS Project View**.

For more information, see ContentStudio Console online help.

Plan the Template Directory Hierarchy

When planning the template directory hierarchy for a single or multiple source code VOBs, you must understand that the directory organization in the source code VOB is the same as your CMS project after the templates and files from the source code VOB are added or associated with the CMS project.

A limitation of the CMS is that each template and file in the CMS project must have a project leaf name that is unique with respect to all other templates and files. For example, the leaf name for a template residing in the CMS project at **ClearCase Assets\Web\frontdoor\open.jsp** would be **open**.

An individual template can have the same leaf name as a file, as long as their project paths are different; but a template cannot have the same leaf name as another template, even if their project paths are different. For example, in the CMS project, a file can reside at **ClearCase Assets\Web\frontdoor\logo.gif** and a template can reside at **ClearCase Assets\Web\sidedoor\logo.asp**. Even though the leaf name **logo** is identical, the CMS project paths are unique. However, two templates cannot share the same leaf name in the CMS project, even if there are in different CMS project paths. For example, if one template resides at **ClearCase Assets\Web\frontdoor\logo.asp** a second template with the same leaf name cannot reside at **ClearCase Assets\Web\sidedoor\logo.asp**.

If a template or file is view private in ClearCase, when you use the ClearCase / CMS integration to add a template or file to the CMS, the leaf names for the templates or files must conform to the same uniqueness constraint. The ClearCase / CMS integration ensures that all templates and files added to the ClearCase / CMS integration have a unique file name and path.

Similarly, if a template or file is previously versioned in ClearCase, when you use the ClearCase / CMS integration to associate a template or file with the CMS, the leaf names for the templates or files must conform to the same uniqueness constraint. The ClearCase / CMS integration ensures that all templates and files added to the ClearCase / CMS integration have a unique file name and path.

When adding or associating a template or file to the ClearCase / CMS integration, the default URL path for a template or file is the directory path in the source code VOB. If you retain the default URL path for a template or file, when you run page generation for the template or file assets using ContentStudio Console, the resulting HTML pages will reside in the same relative directory location (as the source code VOB) in the content VOB. For more information on changing URL paths for a template or file, see *Modify CMS Details* on page 105.

Figure 29 Relationship Between Source Code VOB Directory Organization and the CMS Project

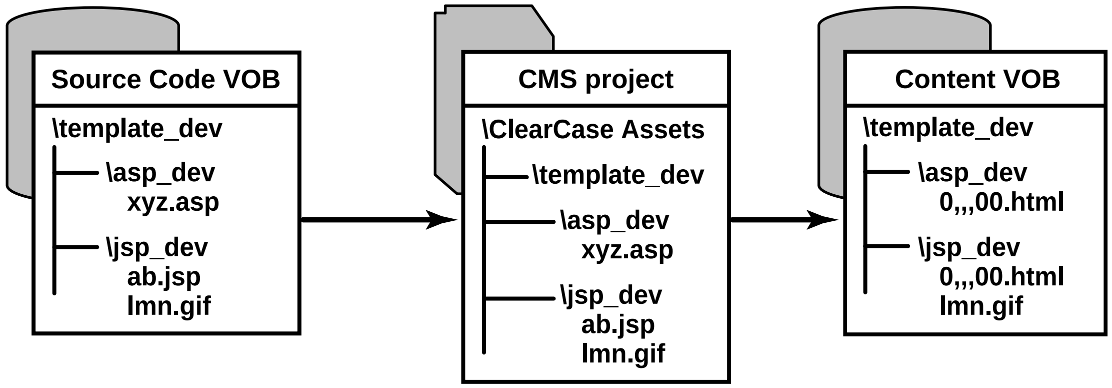


Figure 29 shows the relationship between the source code VOB directory organization and the CMS project hierarchy:

1. The source code VOB location of the ClearCase elements **ab.jsp**, **xyz.asp**, and **lmn.gif** is **template_dev\asp_dev** and **template_dev\jsp_dev**. (The default path in the **CMS Details** dialog for these elements will be **\asp_dev\xyz.jsp**, **\jsp_dev\ab.jsp** and **\jsp_dev\lmn.gif**.)

2. When these ClearCase files are associated with the CMS and then, subsequently checked in after modifications, the files are imported into CMS project. The import operation will copy the associated or checked in version of the file to the CMS project at:
   - ➢ **\ClearCase Assets\template_dev\jsp_dev\ab.jsp**
   - ➢ **\ClearCase Assets\template_dev\jsp_dev\lmn.gif**
   - ➢ **\ClearCase Assets\template_dev\asp_dev\xyz.asp**

3. The CMS assets **ab.jsp**, **xyz.asp**, and **lmn.gif** are defined for an asset list and then a page generation task is run using ContentStudio.

4. The resulting HTML pages and **.gif** file will reside at the following locations in the content VOB:
   - ➢ **\asp_dev\0,,,00.html**
   - ➢ **\jsp_dev\0,,,00.html**
   - ➢ **\jsp_dev\lmn.gif**

Decide Which Programming IDEs to Use for Template Development

ContentStudio enables you to develop ASP or JSP templates for a Vignette CMA or CDA using ClearCase and programming IDEs. Depending on your choice of programming IDE, ClearCase may support an integration with the programming IDE. For example, ClearCase suppports integrations for the following programming IDEs that you can use for ASP or JSP code development:

- ➤ Visual Age for Java for JSP development
- ➤ Forte for Java for JSP development
- ➤ Visual InterDev for ASP development

If your programming IDE does not support a ClearCase integration, you can use the ClearCase integration with Windows Explorer or the Rational ClearCase Explorer to perform the ClearCase operations for the ClearCase / CMS integration.

In addition to managing versioned Vignette templates in a source code VOB residing on a Windows or UNIX platform, the ClearCase integration with CMS enables sites using base ClearCase to use snapshot or dynamic views, based on a UCM or non-UCM work model, for template development.

5.3 Usage Models for the ClearCase/CMS Integration

The ClearCase / CMS integration enables you to develop templates residing in a source code VOB and then check in work that is imported to the CMS. Before starting to use the ClearCase / CMS integration, you must become familiar with the following:

- ➤ Templates usage with the ClearCase / CMS integration
- ➤ ClearCase / CMS integration import model
- ➤ Synchronization of the ClearCase / CMS integration
- ➤ User interfaces for the ClearCase / CMS integration
- ➤ Usage rules for the ClearCase / CMS integration

Using Templates with the ClearCase/CMS Integration

There are three scenarios for using ASP and JSP templates with the ClearCase / CMS integration:

- ➤ New templates originating in ClearCase
- ➤ Existing templates originating in ClearCase
- ➤ Existing templates originating in a CMS project

ContentStudio recommends that you start your template development in a ClearCase view. If you are planning to use the ClearCase / CMS integration, and you have existing templates in a CMS project, you must manually migrate your pre-existing work from a CMS project.

New Templates Originating in ClearCase

The ClearCase / CMS integration assumes that your Vignette template source originates in a ClearCase view. To begin, you must:

1. Use ClearCase to create a snapshot or dynamic view, based on a UCM or non-UCM usage model. (Ensure that you are using a view that defines a branch or stream that is specified in the ClearCase / CMS **Integration Map**.)
2. Copy the template source into your view or use a programming IDE to create a new template source file.

These are now view-private files in a ClearCase view. You must use the ClearCase / CMS integration to add these templates to source control with CMS. For information on adding a template or file to source control with CMS, see *Adding a Template or File to Source Control and CMS* on page 90.

Existing Templates Originating in ClearCase

To associate existing versioned templates in a ClearCase view with the CMS, you must:

1. Create a ClearCase / CMS **Integration Map** using ContentStudio Console for your integration stream or main project branch.
2. Select a template or file and click **Associate with CMS** from the shortcut menu in Windows Explorer or Rational ClearCase Explorer.

For information on associating a template or file to source control with CMS, see *Associating a Template or File with CMS* on page 97.

Migrating Templates that Originate in an Existing Vignette Project

To migrate an existing templates from a Vignette CMS project to a ClearCase view:

1. Use ClearCase to create a snapshot or dynamic view, based on a UCM or non-UCM usage model.
2. Log on to V/5 e-Business Platform Tools and click **Window**>**Production Center**>**Project Manager** to open a Project Manager window.
3. Open the CMS project and select each template that you want to place into ClearCase and click **File**>**Write to File**.
4. From the **Write to File** dialog, save a **.jsp** or **.asp** file for each template to a ClearCase view.

   These are now view-private files in a ClearCase view. You must use the ClearCase/CMS integration to add these templates with ClearCase source control and the CMS. For information on adding a template or file to source control, see *Adding a Template or File to Source Control and CMS* on page 90.

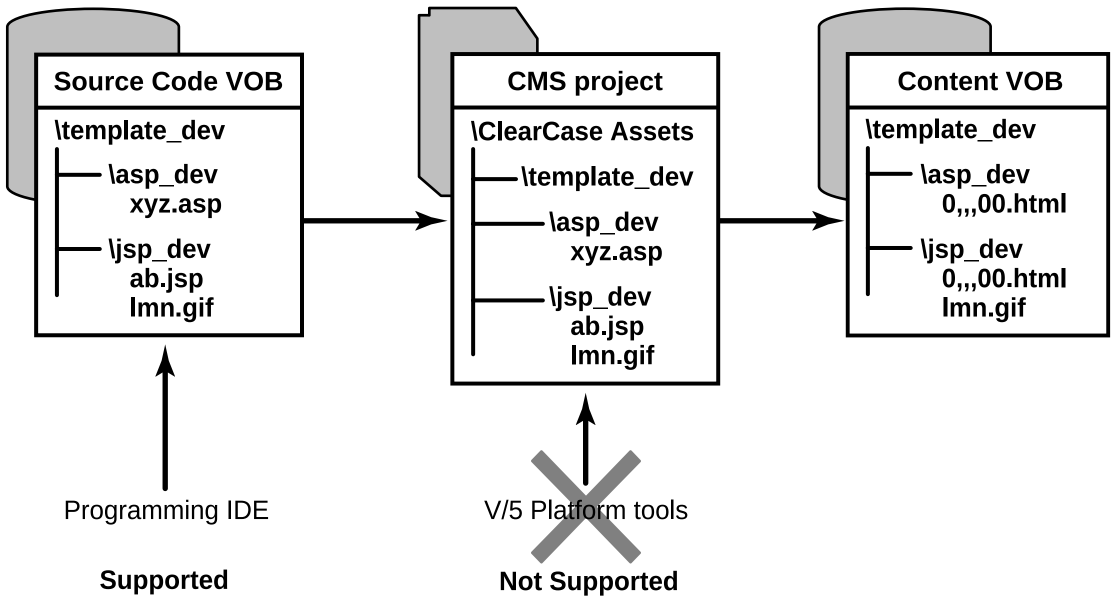
Working with Templates and Files When Using the ClearCase/CMS Integration

The representation of the version of the template or file that is imported into the CMS is not meant to be edited within the CMS. All editing operations on the template or file must take place within a ClearCase view and not within the V/5 Development Center for the CMS. If you do edit a template or file that you have added or associated with the CMS using the V/5 e-Business Platform Tools, the ClearCase/CMS integration will not be able to maintain the synchronization between the source version residing in the ClearCase source code VOB and the version imported into the CMS.

NOTE: If a template or file is changed from within V/5 Development Center for the CMS, when the ClearCase/CMS integration imports or changes a path, the integration will detect that changes have occurred outside of the integration. In this case, the integration writes a warning message to the ClearCase/CMS integration log located in the ContentStudio Console and overwrites the representation of the asset in the CMS project with the latest version in ClearCase.

As Figure 30 shows, the ClearCase/CMS integration is one-directional; there is no synchronization for changes made directly to the template version in the CMS project after it is imported to the CMS.

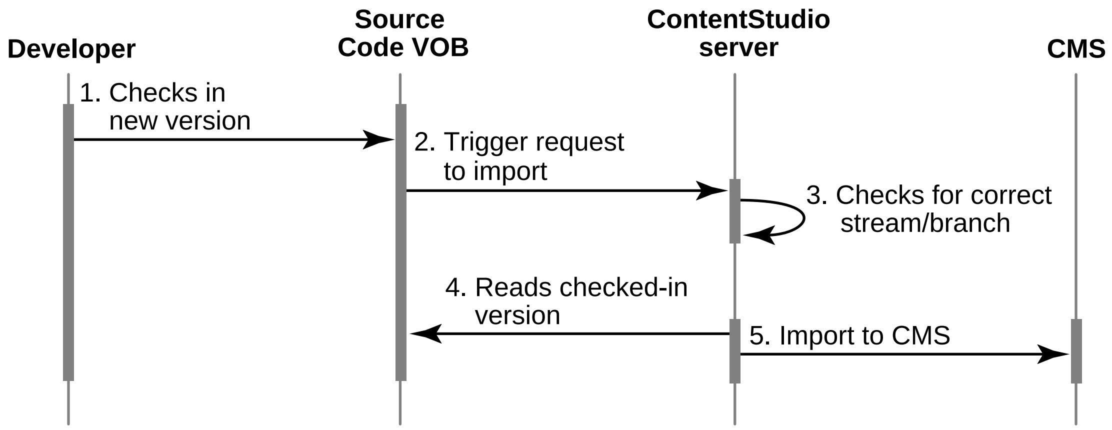
Figure 30 ClearCase / CMS Integration Synchronization



ClearCase/CMS Integration Import Model

The ClearCase / CMS integration maintains synchronization with the CMS project through an import operation. When there is a ClearCase checkin for a version of a template or file that has been previously added or associated with the CMS, the import operation will only proceed if the checkin occurs from a stream or branch that is user-defined in the ClearCase / CMS **Integration Map**. The import operation copies the new version of the template or file to the CMS project. Figure 31 shows the import model for the ClearCase / CMS integration for checkins.

Figure 31 ClearCase / CMS Import Model for Check In of a Template or File


Developer
Source Code VOB
ContentStudio server
CMS
1. Checks in new version
2. Trigger request to import
3. Checks for correct stream/branch
4. Reads checked-in version
5. Import to CMS

1. When the template developer wants to complete work on a template and wants to unit test the work, the template developer checks in a template.

2. Assuming that the template developer is working in a properly configured environment for the ClearCase / CMS integration, the check in operation from a ClearCase view for an element fires a ClearCase trigger. The trigger sends a message to the ContentStudio server.

3. The ContentStudio server checks if the check in is on the project branch or stream defined in the ClearCase / CMS **Integration Map** (by using ContentStudio Console). If so, the ContentStudio server waits for the check in operation to complete.

4. When check in is complete, ContentStudio server observes the check in and reads the checked in version from the source code VOB.

5. ContentStudio server imports the version into CMS.

Synchronizing the ClearCase/CMS Integration

ClearCase / CMS integration processing depends upon the ClearCase / CMS **Integration Map**. When using ContentStudio Console to specify the **Integration Map**, you must define the CMS project view that is associated with the main project branch or integration stream. In addition, the ClearCase / CMS integration uses a ContentStudio-specific integration view for administrative purposes. The view-tag for the ContentStudio-specific integration view is defined on **ClearCase** tab of the Rational Suite ContentStudio Administration Utility.

For the ClearCase / CMS integration to function properly, the config spec for the ContentStudio-specific integration view must remain synchronized with the config spec of the CMS project view. If you use ContentStudio to disable and then enable a different **Integration Map** or you edit an existing **Integration Map** and change the CMS project view, the ClearCase / CMS integration automatically synchronizes the config specs. However, if you edit the config spec of the CMS project view defined in the **Integration Map**, you must perform a manual synchronization.

To synchronize the ContentStudio-specific integration view with the CMS project view:

1. Log in to ContentStudio Console.
2. Click **ClearCase/CMS Integration**.
3. Click **File**>**Resynchronize Integration View**.

User Interfaces for the ClearCase/CMS Integration

After ContentStudio Client and ClearCase or ClearCase LT Client are installed on client desktops and the ClearCase triggers for the ClearCase / CMS integration are applied to all code VOBs for CMS assets, you can begin to use the ClearCase / CMS integration.

The ClearCase / CMS integration supports the following interfaces for working on templates and files residing in the code VOBs.

- Windows Explorer shortcut menu support with ClearCase views using ClearCase or ClearCase LT.
- Rational ClearCase Explorer shortcut menu support for ClearCase views using ClearCase or ClearCase LT.
- ClearCase **cleartool** support for specific commands for ClearCase views using ClearCase or ClearCase LT.
- Programming IDEs that have an existing integration with ClearCase or ClearCase LT.
- ClearCase Automation Library (CAL)

You can use any of these user interfaces when developing templates residing in source code VOBs. However, not all ClearCase / CMS integration functions are supported in each of these user interfaces. For more information on the supported functions for the ClearCase / CMS integration, see *ClearCase/CMS Integration Operations* on page 87.

Windows Explorer

From Windows Explorer, right-click a view-private file in a view to display:

- ➤ **ClearCase**>**Add to CMS**

From Windows Explorer, right-click a versioned file in a view to display:

- ➤ **ClearCase>Associate with CMS**

From Windows Explorer, right-click a versioned file to display:

- ➤ **ClearCase**>**CMS Details**

  NOTE: You should only use the **CMS Details** menu item if you have previously added or associated a template or file with the CMS.

Rational ClearCase Explorer

From Rational ClearCase Explorer, right-click a view-private file in a view to display:

- ➤ **Add to CMS**

From Rational ClearCase Explorer, right-click a versioned file to display:

- ➤ **Associate with CMS**

From Rational ClearCase Explorer, right-click a versioned file to display:

- ➤ **CMS Details**

  NOTE: You should only use the **CMS Details** menu item if you have previously added or associated a template or file with the CMS.

Cleartool

The cleartool command interface uses ClearCase triggers for the ClearCase/CMS integration when the following commands are used.

- ➤ **cleartool checkin**
- ➤ **cleartool rmname**
- ➤ **cleartool rmver**

- **cleartool rmelem**
- **cleartool mv**

For more information on using these cleartool commands with the ClearCase / CMS integration, see Table 6; refer to the *ClearCase Reference Manual* for more information on using these cleartool commands.

Programming IDEs

In some programming IDEs, there is an integration with ClearCase and ClearCase LT. In such cases, the ClearCase / CMS integration supports a checkin directly from the programming IDE (assuming that the stream or branch where the checkin occurs is appropriate).

However, if there is a **Add to Source Control** function supported by the ClearCase integration with a programming IDE, this function will not add the template or file to the ClearCase / CMS integration. If you add a template or file directly to source control from a programming IDE, you must then use the **Associate with CMS** shortcut menu item from Windows Explorer or Rational ClearCase Explorer to create CMS properties and import a copy of the template or file into the CMS project.

ClearCase/CMS Integration Usage Rules

The ClearCase / CMS integration assumes the following usage rules:

- All file leaf names in a source code VOB must be unique to the domain of templates in the CMS, if a template; or to the domain of files in the CMS, if a file. The leaf name must be unique for all templates or files, respectively, to become added or associated with the CMS project.

  Vignette requires leaf name uniqueness in a CMS project. When a template is imported to the CMS project, the template name is tested for uniqueness. If there is a name conflict, the ClearCase / CMS integration displays an error message. Any non-unique file leaf name must be changed before a template or file can be added or associated with the CMS project.

- All of the imported templates will reside in the CMS project **ClearCase Assets**.

  The directory hierarchy of templates and files in the source code VOBs is identical to the project hierarchy in CMS. When a template or file is imported to the CMS project, the project hierarchy in the CMS always begins with **ClearCase Assets** and is followed by the VOB-tag of the imported asset.

- When a template or file is imported to the CMS project, the owner of the template will be the identity that the ContentStudio server uses to log into the CMS. This identity must have the **CS_ROLE** permission assigned to it.

  When a template or file is added to ClearCase, the owner of the template or file in the source code VOB is the user identity of the template developer who adds templates or files to ClearCase source control. When a template or file is imported to the CMS project, all ClearCase / CMS integration assets are owned by the ContentStudio server identity.

- Assuming that the condition for leaf name uniqueness is met, the import operation resulting from a check in from ClearCase will occur only if the stream or branch where the checkin occurs matches the stream or branch specified in the ClearCase / CMS **Integration Map**.

  The template development model for the ClearCase / CMS integration requires a template developer to check in a template or file artifact from a view with a config spec that selects the branch or stream that is defined in the **Integration Map**. If the config spec for the view where the check in occurs selects any other branch or stream, the checkin operation can complete, but there will not be a subsequent import to the CMS project. For more information on defining the **Integration Map**, see ContentStudio Console online help.

- If you are using a non-UCM work model, the main project branch must be a global branch or **\main**.

  When using ContentStudio Console to specify the I**ntegration Map**, you can only select branches that are defined in ClearCase as global branches. For more information on defining global branches in ClearCase, see *Administering ClearCase,* and for ClearCase LT, see *Adminstering ClearCase LT.*

- The **Add to CMS** and **Associate with CMS** operations for the ClearCase / CMS integration import an initial version of the template or file into the CMS project. For the **Add to CMS** and **Associate with CMS** operations to import into the CMS project, the config spec for the view where these operations occur must either select:

  - For UCM work, the development stream associated with the integration stream that is defined in the **Integration Map**.

  - For non-UCM work, a private branch that is a subbranch of the main project branch that is defined in the **Integration Map**.

After the template or file has been added or associated with the CMS project from a development stream or a private branch, you can continue to work on the templates or files using the development stream or private branch. However, when you complete your changes, the work will only be reflected in the CMS project if you deliver the work to the appropriate integration stream, or merge your work on a private branch into the main project branch and then check in the template or file from the main project branch.

➤ The config spec for the CMS project view specified by the **Integration Map** must remain synchronized with the config spec of the ContentStudio-specific integration view.

  If the config spec of the CMS project view specified by the **Integration Map** changes, you must use the ContentStudio Console to synchronize the config spec of the ContentStudio-specific integration view with the config spec of the CMS project view. For more information on synchronizing these config specs, see *Synchronizing the ClearCase/CMS Integration* on page 82.

5.4 ClearCase/CMS Integration Operations

Table 1 shows the ClearCase operations that are supported by a ClearCase / CMS integration trigger action.

Table 6 ClearCase Supported by the ClearCase / CMS Integration

ClearCase Operation	**ClearCase / CMS Integration Trigger Action**	**ClearCase / CMS Integration Action Depends On**
Checkin for a template or file (**cleartool checkin**) For more information, see *Checkin for a Template or File* on page 105.	Import to CMS project	➤ Must be initiated by ClearCase user who is a member of the ClearCase group ➤ Name uniqueness for the file and URL path(s) ➤ Branch or stream for check in must be defined by ClearCase / CMS **Integration Map**
Checkin for a directory containing a template or file (**cleartool checkin**)	Update to CMS project if contents of the checked-in directory have changed. For example, if a file or template has been renamed, when the containing directory is checked in, the CMS project will be updated for the renamed CMS asset.	➤ Must be initiated by ClearCase user who is a member of the ClearCase group ➤ Name uniqueness for the file and URL path(s) ➤ Branch or stream for check in must be defined by ClearCase / CMS **Integration Map**
Deliver (**cleartool deliver - complete**)	The effect of a completed deliver operation for a file or template is that there is a checkout and checkin of the associated element in the integration stream. The checkin from the integration stream will automatically import a representation of the checked in version to the CMS project.	➤ Must be initiated by ClearCase user who is a member of the ClearCase group

Table 6 ClearCase Supported by the ClearCase / CMS Integration

ClearCase Operation	ClearCase / CMS Integration Trigger Action	ClearCase / CMS Integration Action Depends On
Add to CMS For more information, see *Adding a Template or File to Source Control and CMS* on page 90.	Import to CMS project (in working state)	➤ Must be initiated by ClearCase user who is a member of the Clear-Case group ➤ Name uniqueness for the file and URL path(s)
Associate with CMS For more information, see *Associating a Template or File with CMS* on page 97.	Import to CMS project (in working state)	➤ Must be initiated by ClearCase user who is a member of the Clear-Case group ➤ Name uniqueness for the file and URL path(s)
CMS Details For more information, see *Modify CMS Details* on page 105.	Import CMS Properties to CMS project, when associated element is checked in	➤ Must be initiated by ClearCase user who is a member of the Clear-Case group ➤ Name uniqueness for the file and URL path(s)
cleartool rmname For more information, see *cleartool rmname* on page 107.	Delete template or file from CMS project (occurs when the directory is checked in)	➤ Must be initiated by ClearCase user who is a member of the Clear-Case group ➤ Branch or stream for check in must be defined by ClearCase / CMS **Integration Map** ➤ The import operation will only occur when the parent directory is checked in.

Table 6 ClearCase Supported by the ClearCase / CMS Integration

ClearCase Operation	ClearCase / CMS Integration Trigger Action	ClearCase / CMS Integration Action Depends On
cleartool mv For more information, see *cleartool mv* on page 106.	Rename template or file in CMS project or rename directory containing template or file	➤ Must be initiated by ClearCase user who is a member of the Clear-Case group ➤ Name uniqueness for the file and URL path(s) ➤ Branch or stream for check in must be defined by ClearCase / CMS **Integration Map** ➤ The import operation will only occur when the parent directory is checked in.
cleartool rmver For more information, see *cleartool rmver* on page 108.	Delete specified version of template or file from CMS or delete specified version of directory containing template or file	➤ Must be initiated by ClearCase owner of ele-ment ➤ Cannot be the directory version currently selected by the CMS project view
cleartool rmelem For more information, see *cleartool rmelem* on page 109.	Delete template or file from CMS	➤ Must be initiated by ClearCase owner of ele-ment

Adding a Template or File to Source Control and CMS

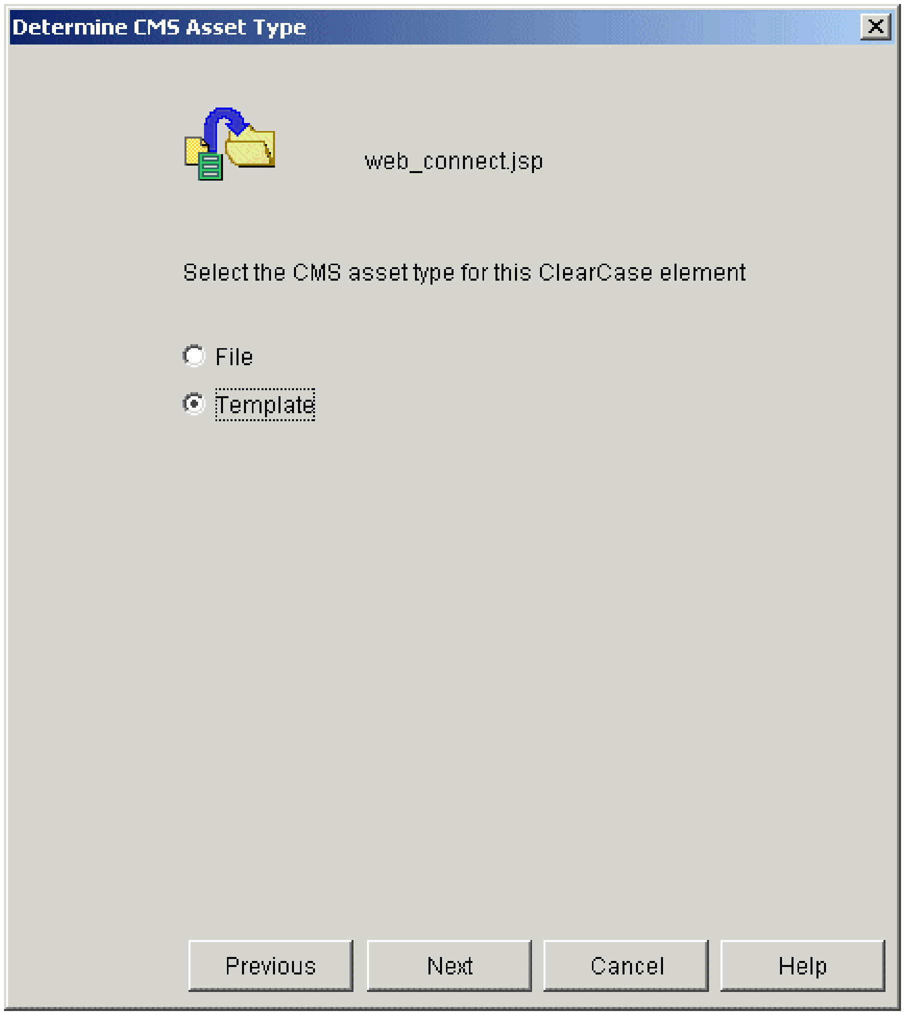
To add a view-private file (template or file) to source control and CMS:

1. Start Rational ClearCase Explorer or Windows Explorer.

2. From a ClearCase view, click to select the view-private file and then right-click and select **Add to CMS**. (If you are using Windows Explorer, right-click **ClearCase** >**Add to CMS**.)

3. When an **Add to CMS** operation is initiated, you must define whether the view-private file that you are adding is a template or a file using the **Determine CMS Asset Type Wizard** as in Figure 32.

Figure 32 Determine CMS Asset Type Wizard Screen for Adding a File or Template Asset


Determine CMS Asset Type

web_connect.jsp

Select the CMS asset type for this ClearCase element

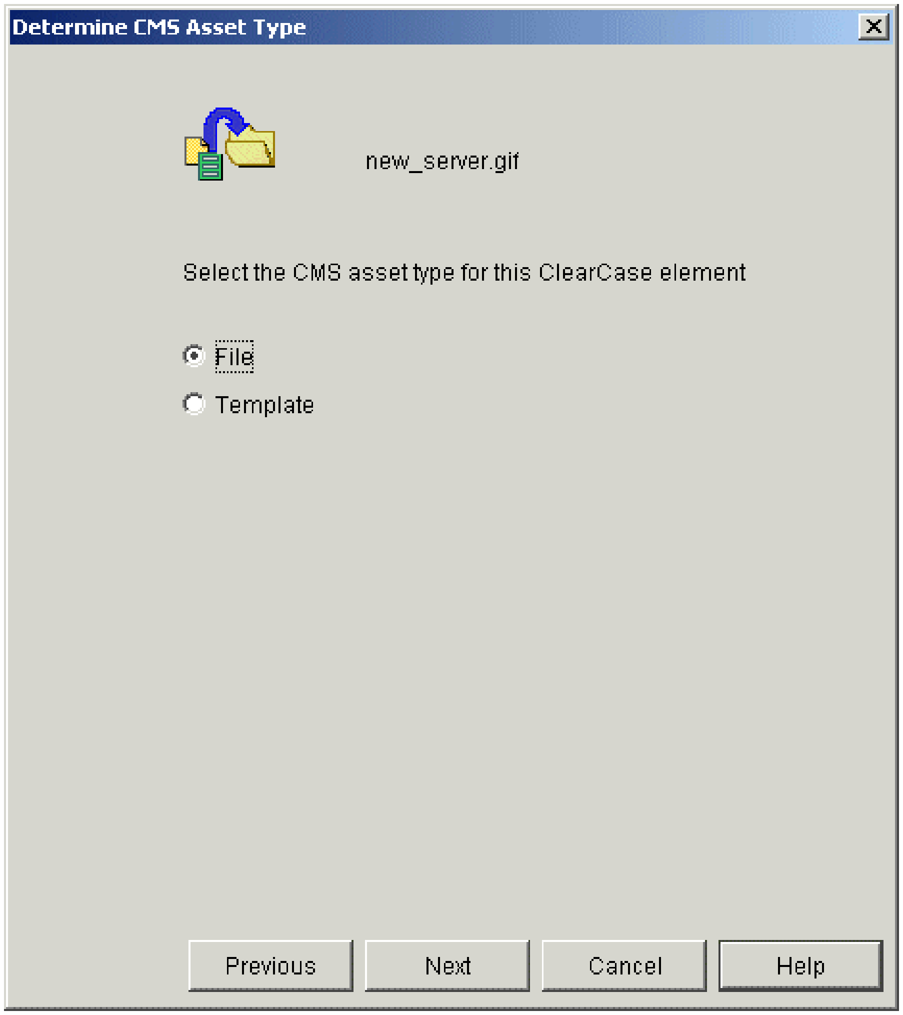
File [✓ Template]

Adding a File

As in Figure 6, using the **General** tab of the **CMS Details Wizard** you must define a name and a URL path for the file.

NOTE: The leaf name and URL path must be unique within the CMS project, if they are not, you will receive an error message.

Figure 33 CMS Details Wizard Screen, General Tab When Adding a File Asset



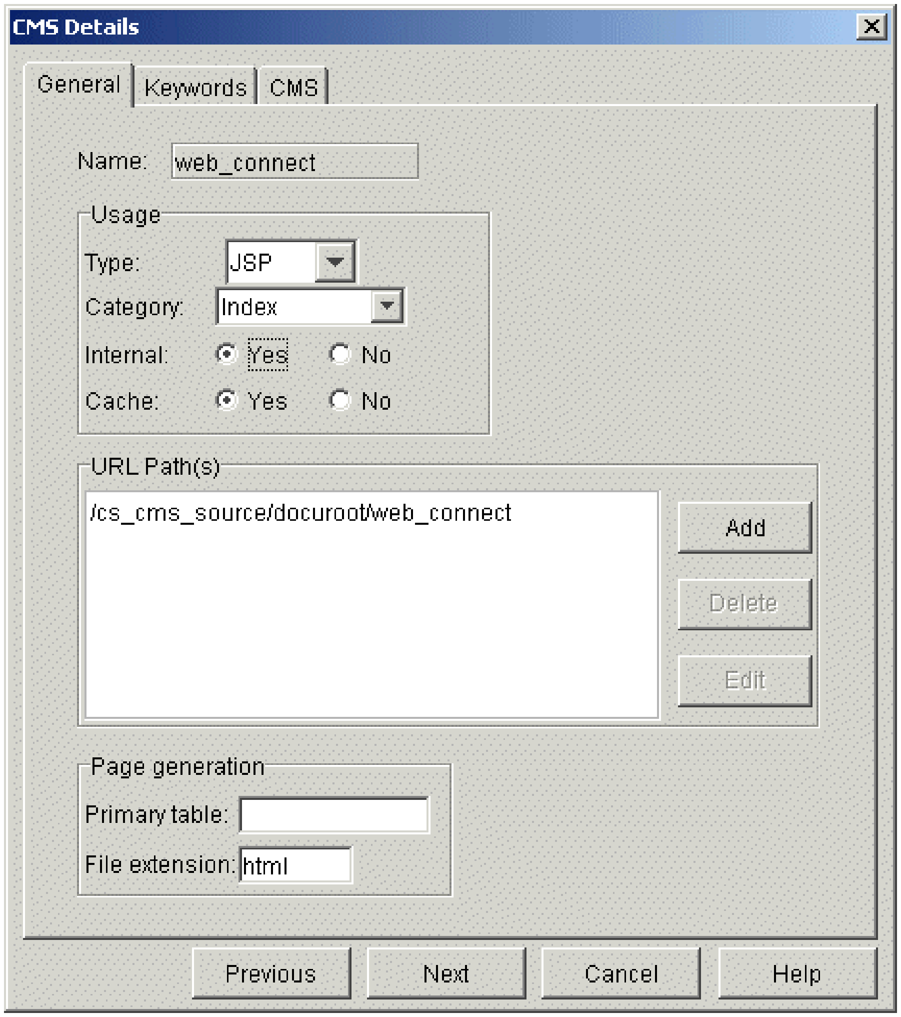
Adding a Template

As in Figure 7, using the **General** tab of the **CMS Details Wizard** you must define a name and a URL path for the template.

NOTE: The leaf name and URL paths must be unique within the CMS project, if they are not, when you click **Next**, you will receive an error message.

When adding a template to source and CMS, all fields must be completed on the **General** tab. For more information, click **Help** in the **CMS Details Wizard**.

Figure 34 CMS Details Wizard Screen, General Tab When Adding a Template Asset



Name: web_connect

Usage

Type: JSP

Category: Index

Internal: [✓ Yes] No

Cache: [✓ Yes] No

URL Path(s)

/cs_cms_source/docuroot/web_connect

Page generation

Primary table:

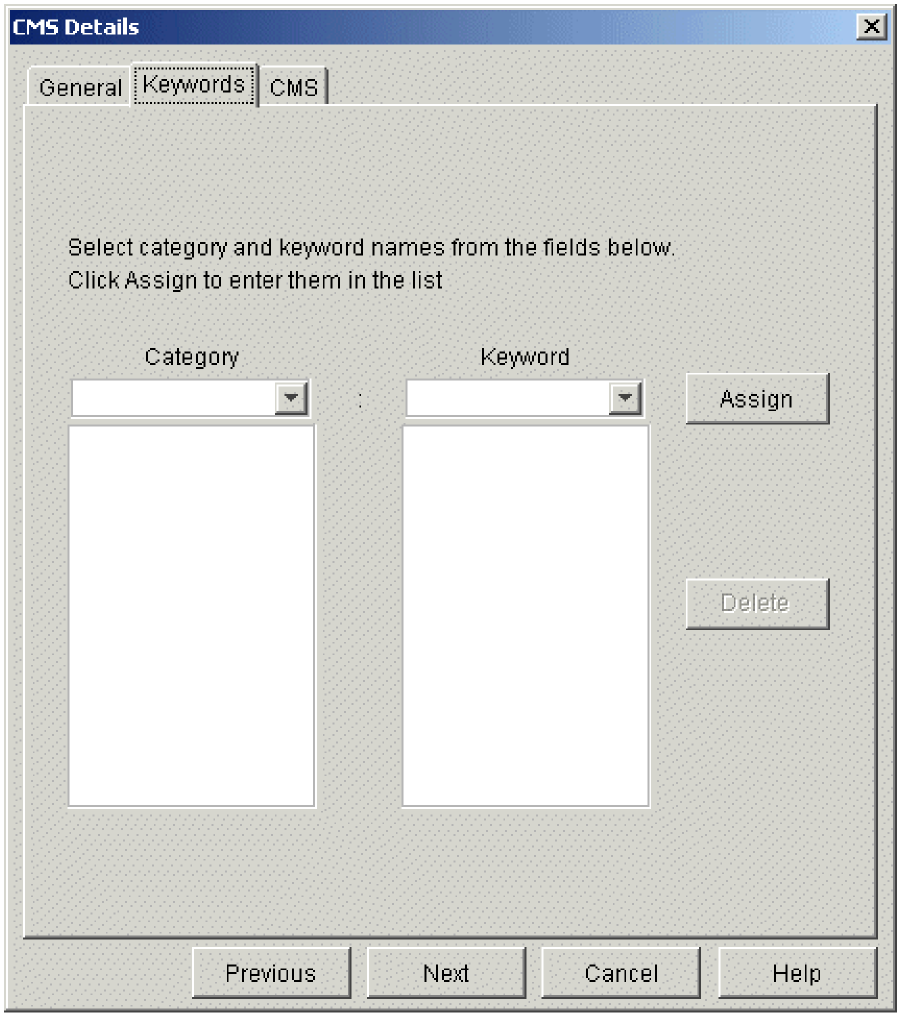
File extension: html

Defining Keywords for a Template or File

As in Figure 35, using the **Keywords** tab of the **CMS Details Wizard** you can define categories and keywords for the template or file. Categories and keywords are the metadata the Vignette runtime system uses for indexing and searches. In ContentStudio, keywords and categories are optional.

NOTE: The **Keywords** tab of the **CMS Details Wizard** will be empty unless the CMS Administrator has created categories and keywords in the CMS. For more information on using keywords and categories, see the Vignette documentation.

Figure 35 CMS Details Wizard Screen, Keywords Tab When Adding a File or Template Asset



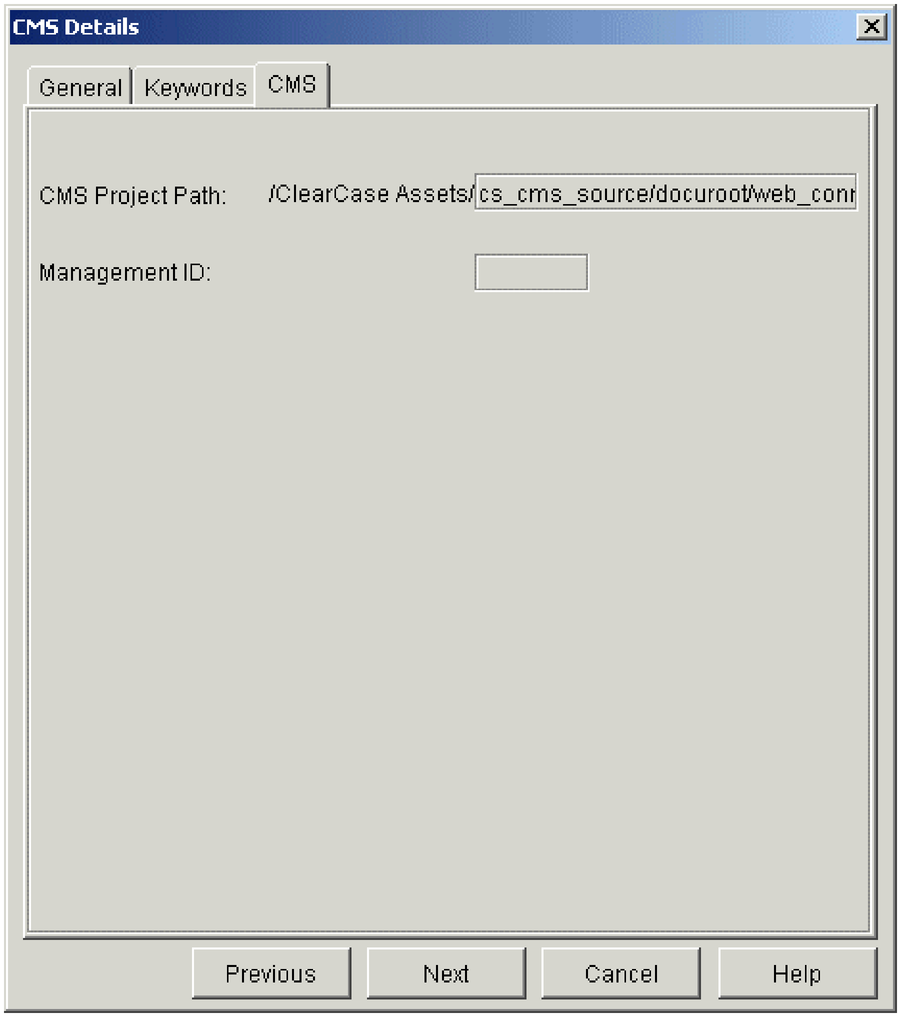
Select category and keyword names from the fields below.
Click Assign to enter them in the list

Viewing CMS Project Location for a Template or File

As in Figure 36, using the **CMS** tab of the **CMS Details Wizard** you can view the CMS project path for the template or file that will be used when imported to the CMS project. The

Management ID box is empty, a value will be assigned when the template or file is imported into the CMS. The information on the **CMS** tab is read-only.

Figure 36 CMS Details Wizard Screen, CMS Tab When Adding a Template or File Asset


CMS Details

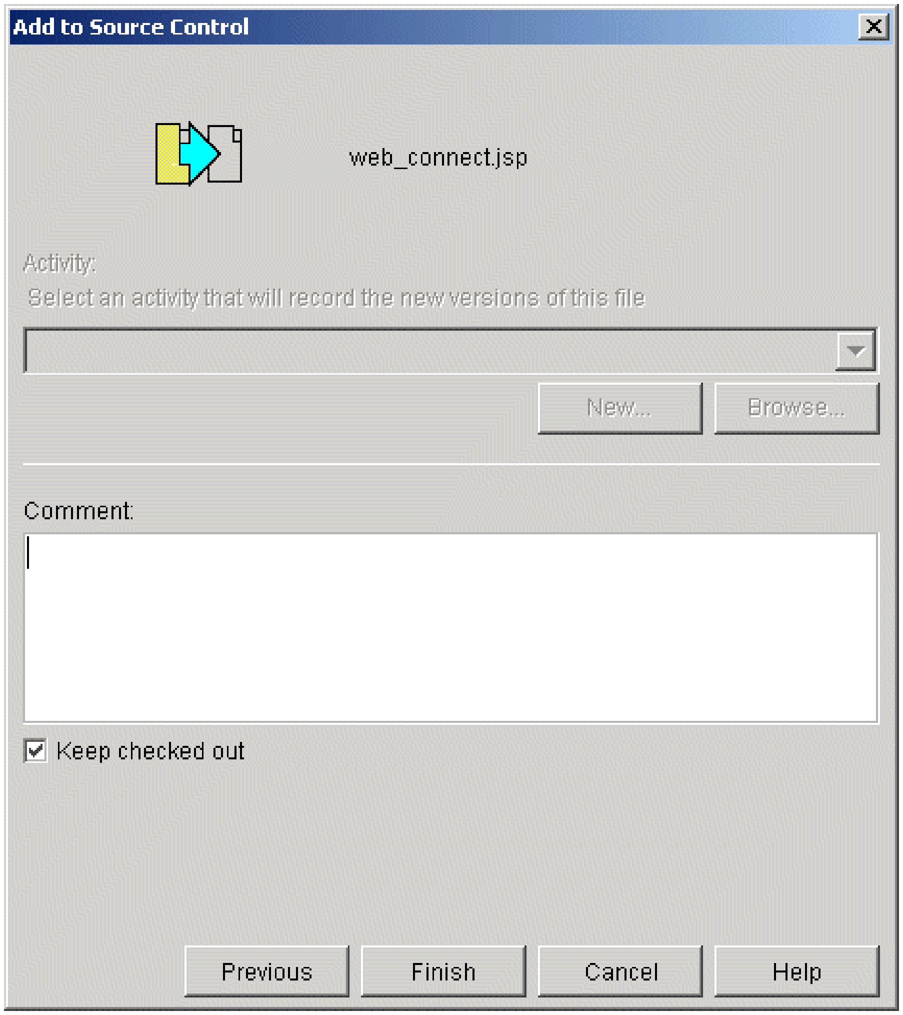
CMS Project Path: /ClearCase Assets/cs_cms_source/docuroot/web_conr

Management ID:

Entering a Comment and Assigning an Activity for a Template or File

As in Figure 37, using the **Add to Source Control Wizard** you can enter a comment and associate a UCM activity (if you are working in a UCM-enabled view). The **Activity** box will only be enabled if the template or file is being added to a UCM project.

Figure 37 Add to Source Control Wizard Screen for a Template or File Asset



Associating a Template or File with CMS

A template or file may have previously been added to source control using one of the possible methods:

➤ A **cleartool mkelem**

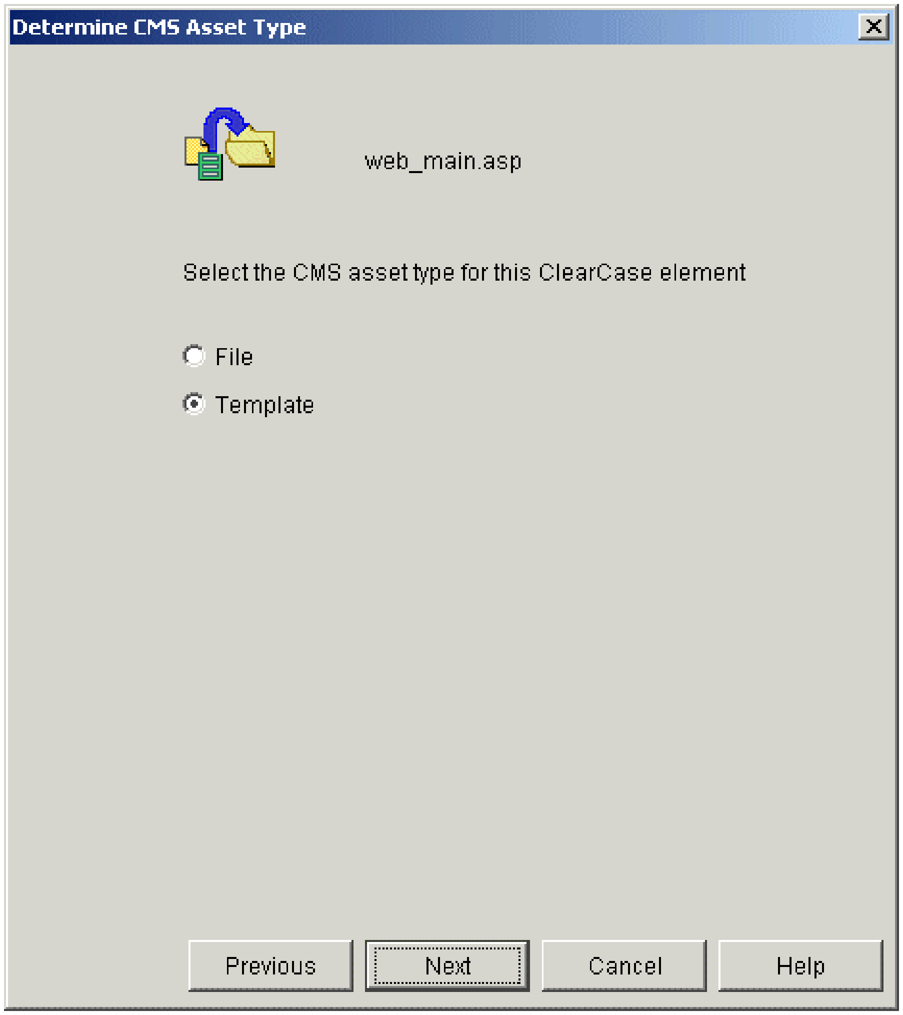
- An **Add to Source Control** operation from the ClearCase integration with a programming IDE
- An **Add to Source Control** operation from a shortcut menu in Windows Explorer or Rational ClearCase Explorer

In all of these cases, if you want to use the template or file already in ClearCase with the ClearCase/CMS integration, you must use the **Associate with CMS** operation.

To associate a template or file already in source control:

1. Start Rational ClearCase Explorer or Windows Explorer.
2. From a ClearCase view, click to select the element that you want to associate with the CMS and then right-click and select **Associate with CMS**. (If you are using Windows Explorer, right-click **ClearCase** >**Associate with CMS**.)
3. When an **Associate with CMS** operation is initiated, you must define whether the element that you are adding is a template or file using the **Determine CMS Asset Type Wizard** as in Figure 38.

Figure 38 Determine CMS Asset Type Wizard Screen When Associating a File or Template Asset



web_main.asp

Select the CMS asset type for this ClearCase element

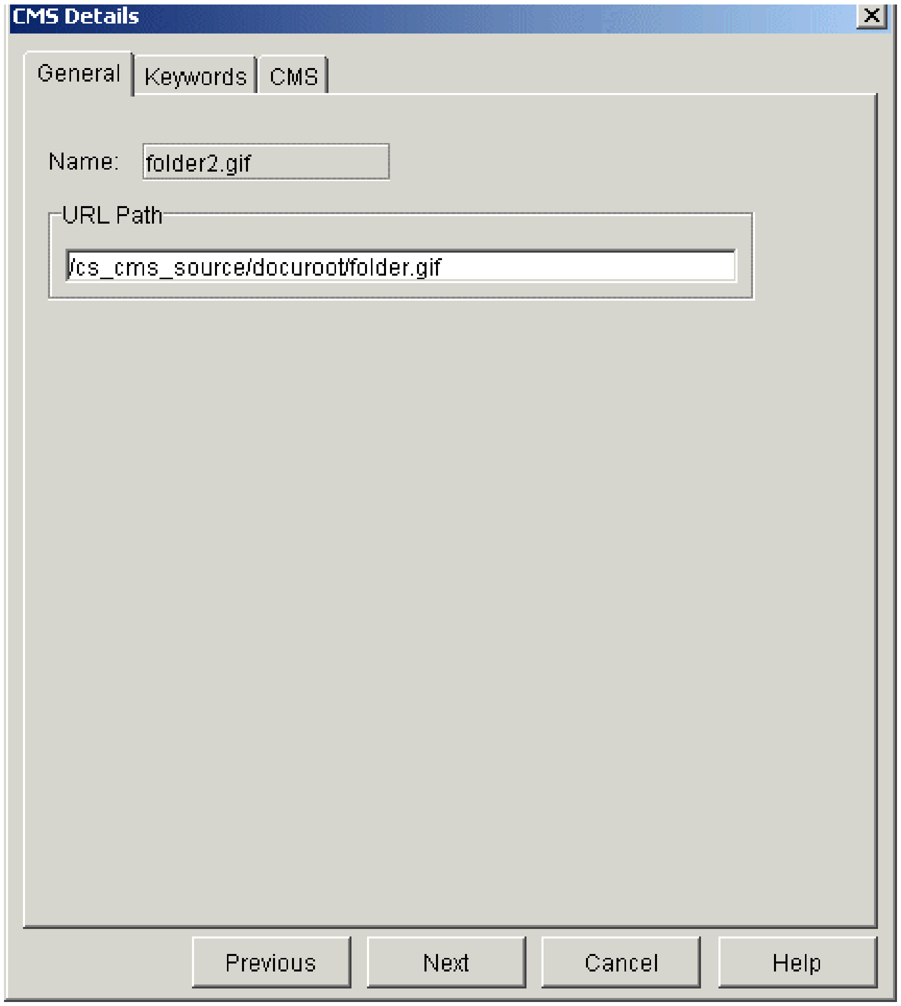
File [✓ Template]

Associating a File

As in Figure 39, using the **General** tab of the **CMS Details Wizard**, the name and URL path for the file are filled in.

NOTE: The leaf name and URL path must be unique within the CMS project, if they are not, you will receive an error message.

Figure 39 CMS Details Wizard Screen, General Tab for Associating a File Asset



Name: folder2.gif

URL Path

/cs_cms_source/docuroot/folder.gif

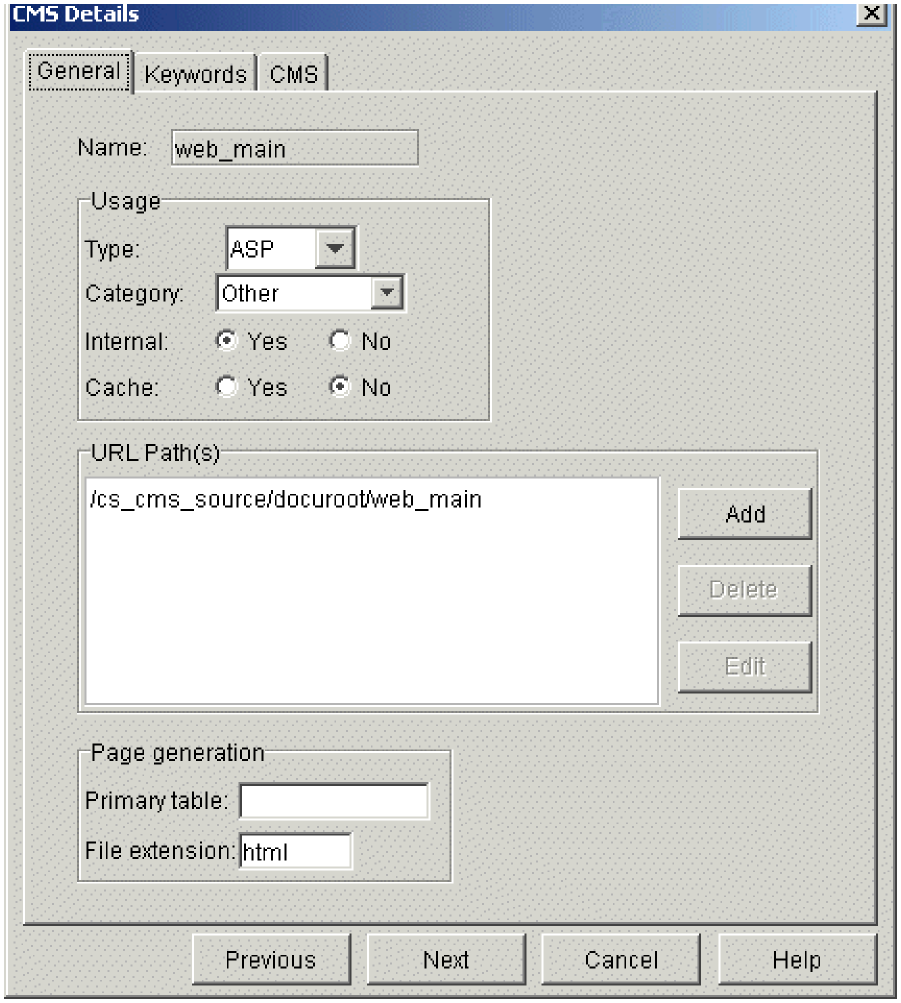
Associating a Template

As in Figure 40, using the **General** tab of the **CMS Details Wizard**, the name and a URL path for the template are filled in.

NOTE: The leaf name and URL paths must be unique within the CMS project, if they are not, when you click **Next**, you will receive an error message.

When adding a template to source and CMS, all fields must be completed on the **General** tab. For more information, click **Help** in the **CMS Details Wizard**.

Figure 40 CMS Details Wizard Screen, General Tab for Associating a Template Asset



Name: web_main

Usage

Type: ASP

Category: Other

Internal: [✓ Yes] No

Cache: Yes [✓ No]

URL Path(s)

/cs_cms_source/docuroot/web_main

Page generation

Primary table:

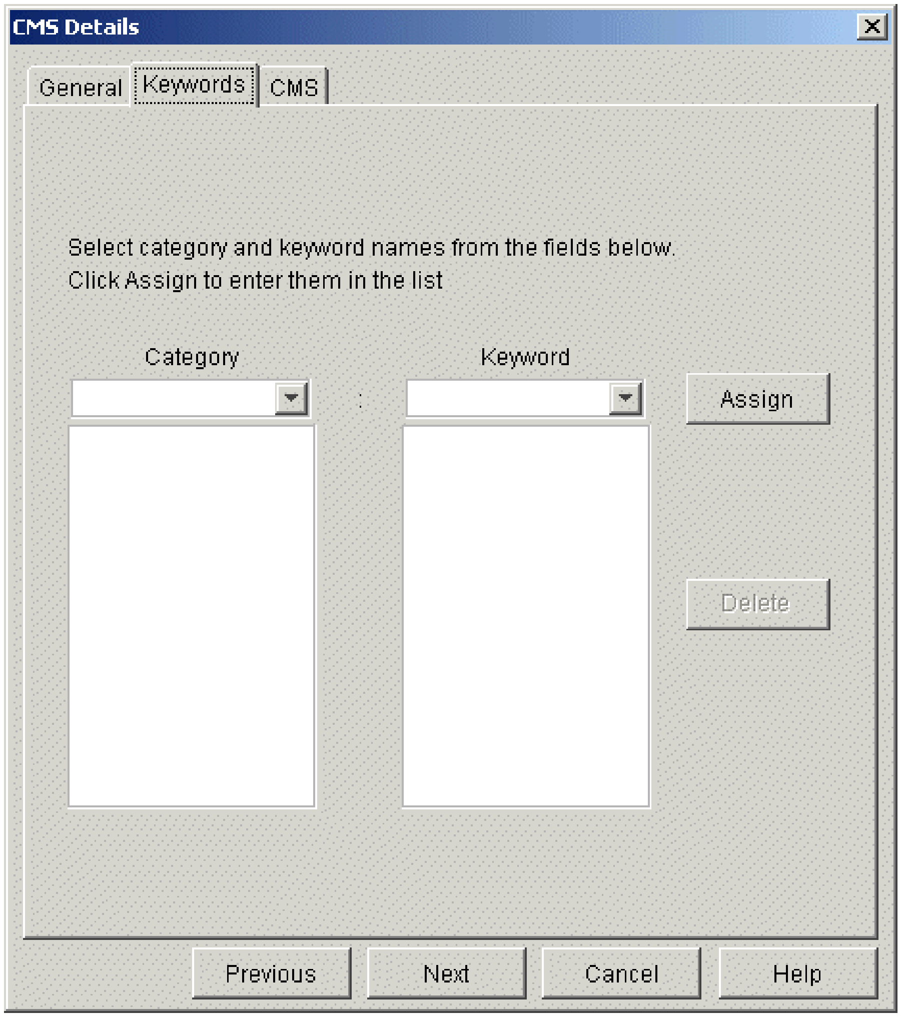
File extension: html

Defining Keywords for a Template or File

As in Figure 41, using the **Keywords** tab of the **CMS Details Wizard**, you can define categories and keywords for the template or file. Categories and keywords are the metadata the Vignette runtime system uses for indexing and searches. In ContentStudio, keywords and categories are optional.

NOTE: The **Keywords** tab of the **CMS Details Wizard** will be empty unless the CMS Administrator has created categories and keywords in the CMS. For more information on using keywords and categories, see the Vignette documentation.

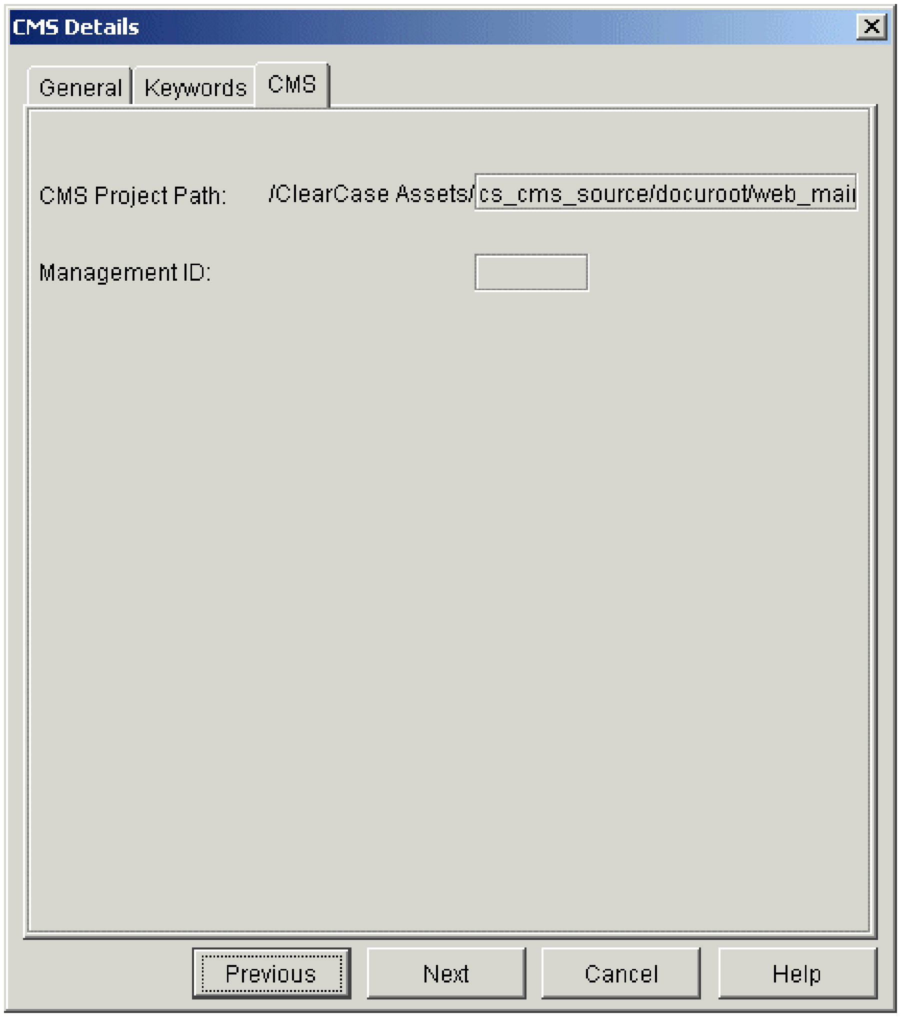
Figure 41 CMS Details Dialog, Keywords Tab When Associating a File or Template Asset


Select category and keyword names from the fields below.
Click Assign to enter them in the list

Viewing CMS Project Location for a Template or File

As in Figure 42, using the **CMS** tab of the **CMS Details Wizard**, you can view the CMS project path for the template or file that will be used when imported to the CMS project. The **Management ID** box is empty, a value will be assigned when the template or file is imported into the CMS. The information on the **CMS** tab is read-only.

Figure 42 CMS Details Wizard Screen, CMS Tab When Associating a File or Template Asset



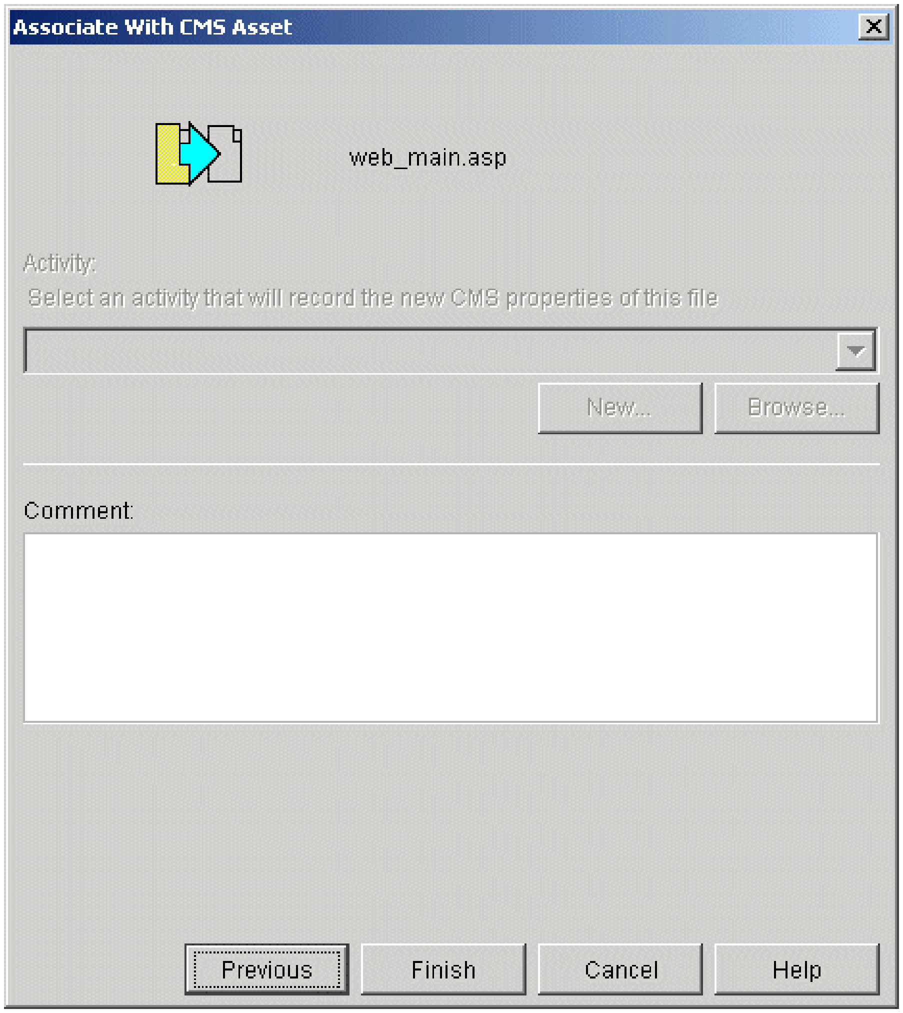
CMS Project Path: /ClearCase Assets/cs_cms_source/docuroot/web_mai

Management ID:

Entering a Comment and Assigning an Activity for a Template or File

As in Figure 16, using the **Associate with CMS Asset Wizard**, you can enter a comment and associate a UCM activity (if you are working in a UCM-enabled view). The **Activity** box will only be enabled if the template or file is being added to a UCM project.

Figure 43 Associate With CMS Asset Wizard Screen for File and Template Assets



Checkin for a Template or File

When checking in a template or file, the ClearCase / CMS integration will only function if you have used either the **Add to CMS** or **Associate with CMS** shortcut menu options.

If you check in a template or file that has not been associated with the CMS, the checkin operation will not perform an import operation to the CMS project.

NOTE: If you check in a template or file from a stream or project branch that is different than the stream or project branch defined in the ClearCase / CMS **Integration Map**, the checkin will proceed, but there will be no import operation for the ClearCase / CMS integration.

If all checkin operation conditions are met, when the element is checked in, the ClearCase / CMS integration will perform an import operation to copy the checked in version of the element to the CMS project. You can use the **Logs** tab for the **ClearCase/CMS Integration** function in the ContentStudio Console to audit the success status of the import operation.

Modify CMS Details

After a template or file has been added or associated with the CMS project, you can modify the CMS Details if you are an authorized ClearCase user for the ClearCase registry region where the code VOBs resides. The CMS Details contain the template or file details that are stored in V / 5 when the artifact is imported to the CMS. After a file or template is added to or associated with the CMS, you must change the CMS Details from a ClearCase view. When the CMS Details are changed for a file or template, assuming that the ClearCase / CMS integration rules are met, the new version of CMS Details is imported to the CMS project.

If you are using UCM with ClearCase, to modify CMS Details:

1. From a view, select a file or template that has been previously added or associated with the CMS.
2. Using ClearCase Explorer or Rational ClearCase Explorer, right-click **CMS Details** from the shortcut menu to open the **CMS Details** dialog. Using **CMS Details** dialog, you can change information in the **General** and **Keywords** tabs.
3. After making changes to the **CMS Details** dialog, click **Apply** or **OK**.

If you are working in the integration stream that is specified in the ClearCase / CMS **Integration Map**, the template or file properties are immediately updated in the CMS project.

If you are working in a development stream (where the development stream specifies the integration stream specified in the ClearCase / CMS **Integration Map**), the update of the template or file properties will not occur until the associated template or file is delivered to the integration stream.

If you are using a non-UCM ClearCase, to modify CMS Details:

1. From a view, select a file or template that has been previously added or associated with the CMS.

2. Using ClearCase Explorer or Rational ClearCase Explorer, right-click **CMS Details** from the shortcut menu to open the **CMS Details** dialog. Using **CMS Details** dialog, you can change information in the **General** and **Keywords** tabs.

3. After making changes to the CMS Details dialog, click **Apply** or **OK** from the **CMS Details** dialog.

   If you are working in the main project branch that is specified in the ClearCase / CMS **Integration Map**, the template or file properties are immediately updated in the CMS project.

   If you are working in private branch (where your view's config spec defines a main project branch that matches the branch name specified by the ClearCase / CMS **Integration Map**), the update of the template or file properties will not occur until the associated template or file is merged into the main project branch and, subsequently checked in.

If all of the conditions for modifying CMS Details are met, the ClearCase / CMS integration will perform an import operation to copy the latest version of the element's CMS Details to the CMS project. You can use the **Logs** tab for the **ClearCase/CMS Integration** function in the ContentStudio Console to audit the success status of the import operation.

cleartool mv

After a template or file has been added or associated with the CMS project, you can use **cleartool mv** to move or change the name of a file or template. When a **cleartool mv** is executed, the ClearCase / CMS integration checks the CMS project for leaf name uniqueness. If the effect of **cleartool mv** operation still preserves a unique leaf file name with respect to the CMS project, when the containing directory is checked in, the ClearCase / CMS integration automatically changes the associated CMS Details for the new file leaf name and URL path information and changes the CMS project path and file name for the new location.

If you are using ClearCase UCM and execute the **cleartool mv** in an integration view, the integration stream defined in your integration view must be the exact stream name that has been defined in the ClearCase / CMS **Integration Map**. Assuming that the ClearCase / CMS integration rules are met, the import to CMS operation happens immediately. If you execute **cleartool mv** in the development view, the import operation will not occur until the development stream is delivered.

If you are using a non-UCM ClearCase and execute the **cleartool mv** in a view where the config spec is specified for a main project branch, the main branch defined in your view must be the exact branch name that has been defined in the ClearCase / CMS **Integration Map**. Assuming that the ClearCase / CMS integration rules are met, the import to CMS operation happens immediately. If you execute **cleartool mv** in a view whose config spec selects a private branch, the import operation will not occur until the private branch is merged to the main project branch.

If all of **cleartool mv** operation conditions are met, the ClearCase / CMS integration will perform an import operation to copy the renamed version of the element to the CMS project. You can use the **Logs** tab for the **ClearCase/CMS Integration** function in the ContentStudio Console to audit the success status of the import and remove operations.

cleartool rmname

After a template or file has been added or associated with the CMS project, you can use **cleartool rmname** to remove the name of the element from a directory version. When a **cleartool rmname** is executed, the ClearCase / CMS integration will remove the file name from its existing CMS project location when the containing directory is checked in.

NOTE: The associated CMS project asset is moved to **\ClearCase Assets\**<*VOB-tag*>**\lost+found\**.

If you are using ClearCase UCM and execute the **cleartool rmname** in an integration view, the integration stream defined in your integration view must be the exact stream name that has been defined in the ClearCase / CMS **Integration Map**. Assuming that the ClearCase / CMS integration rules are met, the CMS remove operation happens immediately. If you execute **cleartool rmname** in the development view, the CMS remove operation will not occur until the development stream is delivered.

If you are using non-UCM ClearCase and execute the **cleartool rmname** in a view where the config spec is specified for a main project branch, the main branch defined in your view must be the exact branch name that has been defined in the ClearCase **Integration Map**. Assuming that the ClearCase / CMS integration rules are met, the CMS remove operation happens immediately. If you execute **cleartool rmname** in a view whose config spec selects a private branch, the CMS remove operation will not occur until the private branch is merged to the main project branch.

If all of **cleartool rmname** operation conditions are met, the ClearCase / CMS integration removes the associated artifact from the CMS project. You can use the **Logs** tab for the **ClearCase/CMS Integration** function in the ContentStudio Console to audit the success status of the remove operation.

cleartool rmver

After a template or file has been added or associated with the CMS project, you can use **cleartool rmver** to remove a version of the element from the version tree of the element. When a **cleartool rmver** is executed, the ClearCase / CMS integration checks to see if the version of the associated artifact in the CMS project is the same version that is specified in the **cleartool rmver** operation. If the version in the CMS project does match the version specified in the **cleartool rmver** operation, the version in the CMS project is removed, and whichever version that the CMS project view (specified in the ContentStudio Console) selects is imported to the CMS project. (If the version specified in the **cleartool rmver** operation does not match the version in the CMS project, there is no further processing by the ClearCase / CMS integration.)

If you are using ClearCase UCM and execute the **cleartool rmver** in an integration view, the integration stream defined in your integration view must be the exact stream name that has been defined in the ClearCase / CMS **Integration Map**. Assuming that the ClearCase / CMS integration rules are met, the import to CMS operation happens immediately. If you execute **cleartool rmver** in the development view, the import operation will not occur until the development stream is delivered.

If you are a non-UCM ClearCase, for the change to take effect immediately, you must execute the **cleartool rmver** in a view where the config spec is specified for a main project branch. The main branch defined in your view must be the exact branch name that has been defined in the ClearCase / CMS **Integration Map**. Assuming that the ClearCase / CMS integration rules are met, the import to CMS operation happens immediately. If you execute **cleartool rmver** in a view whose config spec selects a private branch, the import operation will not occur until the private branch is merged to the main project branch.

If all **cleartool rmver** operation conditions are met, the ClearCase / CMS integration removes the associated version of the artifact from the CMS project and imports the next latest version of the element into the CMS. You can use the **Logs** tab for the **ClearCase/CMS Integration** function in the ContentStudio Console to audit the success status of the remove and import operations.

cleartool rmelem

After a template or file has been added or associated with the CMS project, you can use **cleartool rmelem** to remove the element from the source code VOB. When a **cleartool rmelem** is executed, the ClearCase / CMS integration removes the associated artifact from the CMS project.

Unlike the other supported ClearCase / CMS integration operations, since **cleartool rmelem** affects all version of a template or file asset, there are no integration restrictions on performing the operation from a specific stream or branch that is defined by the **Integration Map**. When a **cleartool rmelem** is executed on an element that has been previously added or associated with the CMS project, the operation occurs immediately.

CAUTION: Use care when executing a **cleartool rmelem** operation on an element that has been previously associated or added to the CMS project; the affect of this operation will be to remove the element from the source code VOB and the corresponding asset in the CMS project.

You can use the **Logs** tab for the **ClearCase/CMS Integration** function in the ContentStudio Console to audit the success status of the remove operation.

5.5 ClearCase/CMS Integration Log

The ClearCase / CMS integration logs the processing status for the supported ClearCase operations.

To view the ClearCase / CMS log:

1. Log on to ContentStudio Console.

2. Click **ClearCase/CMS Integration** and then click the **Log** tab.

The following are examples of the ClearCase / CMS integration log entries for the add (or associate) and the check-in operation. These examples are representative of integration logging.

All supported ClearCase operations have corresponding log entries. For more information on using the ClearCase / CMS integration log, see ContentStudio Console online help.

Example Log for an Add to CMS or Associate with CMS Operation

```
01/06/08 13:06:50 Inform AssociateWithCMS "jsmith" kermit
"C:/cc_cms_view/cs_cms_source/docuroot/web.jsp"
Checking uniquenes of "C:/cc_cms_view/cs_cms_source/docuroot/web.jsp" as
template. Leaf name was unique.

01/06/08 13:06:55 Inform AssociateWithCMS "jsmith" kermit
"C:/cc_cms_view/cs_cms_source/docuroot/web.jsp"
Check uniqueness of URL Path(s) for template:
/cs_cms_source/docuroot/web
URL path(s) are unique.

01/06/08 13:06:59 Inform AssociateWithCMS "jsmith" kermit
Check if branch path "\main\cs_cms_jsp\1" contains branch under integration
control. Project branch cs_cms_jsp is managed by integration.

01/06/08 13:07:03 Inform AssociateWithCMS "jsmith" keremit
"C:/cc_cms_view/cs_cms_source/docuroot/web.jsp"  \main\cs
Associated "C:/cc_cms_view/cs_cms_source/docuroot/web.jsp" with CMS template.

01/06/08 13:07:07 Inform AssociateWithCMS "jsmith" keremit
Setting the version OID for metadata succeeded.
```

In general, an add or associate operation for the integration logs the following:

1. The verification for leaf name uniqueness.
2. The verification for URL path uniqueness.
3. The verification for the stream or branch specified in the **Integration Map**.
4. The confirmation of the import to the CMS project.
5. The confirmation for metadata creation.

Example Log for a Check-in Operation

```
01/06/08 13:16:37 Inform CheckIn "jsmith" kermit
"cs_cms_sourcedocuroot@@maincs_cms_jspCHECKEDOUT.379web.jspmaincs_cms_jspCHECK
EDOUT.473"  \main\cs

\main\cs_cms_jsp\CHECKEDOUT is on project, brtype:cs_cms_jsp, branch.

Will import after checkin completes.
Succeeded in importing source from
\main\cs_cms_jsp\2 for CMS asset,
"/ClearCase Assets/cs_cms_source/docuroot/web.jsp"
```

In general, a checkin operation for the integration logs the following:

1. The verification that the check-in operation is occurring from the stream or branch specified by the **Integration Map**.
2. The confirmation of the successful import to the CMS project.

5.6 Additional Operations When Working with Templates and Files Using the ClearCase/CMS Integration

In addition to the ClearCase operations supporting the ClearCase / CMS integration, in the process of template development, you will also need to use the following operations:

- ➤ Preview a template
- ➤ Define asset list items for page generation
- ➤ Manage workflow for template development

Preview a Template in a Browser

After checking in a new version of a file or template (with a successful import), you can use these methods to preview and unit test your work residing in the Development CDS:

Method 1: Use V/5 Platform Tools to Preview

1. Log on to V/5 Platform Tools.
2. Select the Template View from the Production Center.
3. From the Template View window, select a template to preview and click **Template** > **Preview** to open the template in a browser window

Method 2: Use a Web Browser to Preview

1. Start a browser on your desktop.
2. Enter the URL for the template in the Development CDS.

Method 3: Use a Programming IDE to Preview

1. Start a programming IDE on your desktop.
2. From a view, use your programming IDE's capability to preview your template.

   NOTE: If your template references content records residing in the Vignette CMS, template previewing using this method probably will not be able to display Vignette content records.

If you encounter problems with your template code, you will need to return to your ClearCase view to continue template development, then check in your latest changes, and preview again.

When your template passes unit testing, you can use V/5 tools to launch the template for page generation by ContentStudio.

Define Asset Lists for Page Generation

After unit testing your templates, you must use ContentStudio Console to define asset list items for your templates that have passed unit testing and are in a live state.

For more information on defining asset lists, see Chapter 2 or logon to ContentStudio and click **Page Generator**, click the **Asset List Mgt** tab, and then click **Help**>**Asset List Mgt**.

When a page generation task is run, the generated HTML pages reside in the content VOB.

Manage Workflow for Templates and Files

After generating HTML pages in the content VOB, Quality Assurance can system test the HTML pages. If problems in the HTML pages are encountered in system testing, Quality Assurance can use a ClearQuest client to submit code defects. If there are associated content problems for an HTML page, ContentStudio supports a child option for a ContentChangeRequest record. For more information on using ClearQuest to manage workflow for defects and ContentChangeRequest records, see Chapter 3.

Glossary

ADMINISTRATOR. The member of a Web team who installs and configures the server software for ContentStudio. The administrator also configures the Page Generator, the ClearQuest integration, the ClearCase / CMS integration, maintains the user accounts and group assignments in V / 5, and the deployment schedules in ContentStudio.

ASSET LISTS. A list of persistent objects for all content items in a CMS project that are in the CDS. Asset lists are used to designate candidates for page generation. They are stored in the ContentStudio database and maintained using the ContentStudio Console.

ASSET TYPE. The type of object defined in an asset list. An asset type can be a file, a template, a content record, or a project.

BUSINESS ANALYST. The member of a Web team who defines requirements for the Web application.

CDA. See content delivery application.

CDS. See content delivery server.

CMA. See content management application.

CMS. See content management server.

CMS OPTIONS. The ClearCase / CMS integration enables the template developer to create and manage V / 5 template or staic file details directly from a ClearCase view using the CMS Options dialog.

CHANGE REQUEST. A record in ClearQuest that is used during each phase of the development cycle to track software defects or define requests for enhancements for an application or component stored in a code VOB. A change request may be one of three types: defect, enhancement request, ContentChangeRequest. See also *content change request*.

CLEARCASE/CMS INTEGRATION. A feature of ContentStudio that supports an integration between a ClearCase code VOB for template development and a Vignette CMS project.

CLEARCASE/CMS INTEGRATION MAP. The stream or branch and ClearCase view that is specified for the ClearCase / CMS integration using the ContentStudio Console.

ClearCase/CMS integration view. The ContentStudio-specific view that is specified using the ContentStudio Administration Utility for exclusive use by the ClearCase / CMS integration.

ClearQuest integration. A feature of ContentStudio that mirrors content change requests from ClearQuest in V / 5 as a project task. The progress of V / 5 tasks can be tracked in ClearQuest for unified reporting of both V / 5 task states and traditional development activity.

CMS database. A database that contains the template and content records used to create Web pages.

CMS project view. The view specified in the ClearCase / CMS Integration Map using the ContentStudio Console. This view is used by the ClearCase / CMS integration for administrative purposes.

code VOB. A ClearCase VOB, which can be enabled for UCM, that contains code for the Web application (for example: JavaBeans, COM objects, and ASP / JSP pages that are not contained in ASP or JSP templates) and, when using the ClearCase / CMS integration, ASP, JSP templates and files.

content change request. A ClearQuest record type for ContentStudio. When you use the ClearQuest integration, the ClearQuest record is mirrored in V / 5 as a project task when a content change request is assigned an owner. A ContentChangeRequest record may either be a standalone record or a child of a user-defined parent record type.

content contributor. Any member of a Web team who creates content for the Web application. The content contributor logs into a CMA to submit, modify, or delete content from the Vignette content database. A content contributor can also use a CMA to route changes for review or approval.

content delivery application (CDA). A set of V / 5 templates that process data and format Web pages.

content delivery server (CDS). A V / 5 E-business Platform component that, in conjunction with a Web server, assembles pages from V / 5 templates and content items.

content item. A file, record, or template that resides in the CMS database.

content management application (CMA). A set of V / 5 templates that manage and process data in the CMS database. Some templates produce Web pages that are used to edit or enter content. Others are used to process data, for example, to add new records, to modify existing records, or to delete records.

content management server (CMS). A server host that processes the production management information that travels throughout a platform configuration. The CMS acts as the central coordinator of the platform configuration.

content VOB. A ClearCase VOB that contains files and the static Web pages produced by the Page Generator.

ContentStudio Console. A graphical user interface that is used to manage page generation, deployment, Rational NetDeply triggers, and the ClearCase / CMS integration; it runs on client desktops.

CURL (CUSTOM URL). A custom form of URL that the V/5 E-business Platform software uses to identify Web pages generated from a platform template. A CURL serves the same purpose as a URL, but provides additional information:

- ➤ Whether to cache generated pages
- ➤ The path to the directory where pages are cached
- ➤ Template identification
- ➤ Content identification
- ➤ Browser feature identification (variations)

DEPLOYMENT. The process of moving Web artifacts from a ClearCase VOB to one or more target Web servers.

DEPLOYMENT SOURCE. One or more ClearCase VOBs.

DEPLOYMENT TARGET. Any Web server on which the Rational NetDeploy agent is installed and configured.

DEPLOYMENT TASK. A subset of the topology and a deployment schedule that specifies which versions to deploy.

DEPLOYMENT VIEW. A ClearCase view of the files to be deployed that is configured with the specification from the deployment task.

DIRECTORY MAPPING. The mapping of the source location of artifacts stored in a ClearCase VOB to the directory on a target Web server. This mapping specifies where the artifacts are to be deployed.

EDITOR. The member of a Web team who approves content and templates. (When these items are approved, new or revised Web pages can be generated.) The editor is also the V/5 project owner and manages the V/5 workflow.

FULL DEPLOYMENT. A full deployment is one where all files are sent and all directory maps for a server are involved, otherwise the deployment is partial.

IMPORT. When a template or file is checked in from the code VOB, the ClearCase/CMS integration imports a copy of the checked in version of the template or file to the CMS project.

IINTEGRATION MAP. A configuration setting for the ClearCase/CMS integration defined by the Administrator using the ContentStudio Console.

LAUNCH. The action of making records, files, and templates visible on the Live CDS with a status of live. Only a project owner can launch content in a project.

LOG. The file that records all page generation and deployment requests.

PAGE GENERATION. The process of creating a new version of a page when a template or content record is launched to the Live CDS. This feature is controlled from the ContentStudio Console.

PAGE-GENERATION TASK. An instance of running the Page Generator based on an associated asset list.

PAGE-GENERATION VIEW. The ClearCase view used by the Page Generator.

PAGE GENERATOR. The ContentStudio component that generates HTML pages and files , which correspond to one or more V/5 assets (files, templates, or content records), and puts these HTML pages and files into a ClearCase VOB. The Page Generator uses a ClearCase view to select the latest versions on the **main** branch in the content VOB.

PARTIAL DEPLOYMENT. A partial deployment is either a deployment for which only a subset of a server's directory maps are deployed or for which the copy policy is **Changes only**.

PREVIEW. In the V/5 E-business Platform software , to view a template, combined with its content, through a web browser.

PROJECT. An organizing container or grouping of content items and workflow in the V/5 Production Center. A project includes the following:

- ➤ Content—The database records, files, and templates.
- ➤ Workflow—The tasks done on each record.
- ➤ People—The users who manage the project or work on the records, files, and templates.

PROJECT MANAGER. The member of the Web team who is responsible for reporting on task status and maintaining the project schedule.

PROJECT OWNER. In the Production Center, the user who has responsibility for a project. A project may have one or more owners. Only project owners can launch and expire records, files, templates, and projects. A project owner can perform all actions on all objects in that project.

PROJECT TASK. A task that applies to the project as a whole, rather than to individual records, files, and templates; for example, posting completed content to the Live CDS at daily at 7:00 A.M. and 6:00 P.M. The ClearQuest integration synchronizes project tasks in Vignette with content change requests in ClearQuest.

RATIONAL SUITE CONTENTSTUDIO BASE FEATURES. The Rational products that are included with ContentStudio: ClearCase LT, ClearQuest, RequisitePro, SoDA, TestManager, and Rational Unifed Process.

RATIONAL NETDEPLOY. The ContentStudio component that transfers Web application artifacts from ClearCase VOBs to the target Web servers.

RATIONAL NETDEPLOY AGENT. The ContentStudio component that runs on target Web servers to manage communications between the server and the deployment controller.

RATIONAL NETDEPLOY TRIGGER. NetDeploy triggers are user-defined routines for processing before or after a specified operation; a Rational NetDeploy trigger can run when executing deployment, redeploy, or rollback tasks.

RECORD. In the platform software, a row of data in a database table and the data in the platform system tables that defines any production management information related to the content (for example, content provider's name).

ROLE. A means of grouping sets of privileges that helps the administrator assign multiple privileges to Vignette users. For example, the site manager can create the Writers role and assign Create Article, Edit Article, and Add Image privileges to that role. Roles are not module specific; the administrator can assign privileges from different modules to a single role.

SECURE SOCKETS LAYER (SSL). A security protocol for Internet communications. SSL employs a combination of symmetric encryption and public key / private key encryption to ensure secure communications across a network. SSL handles authentication and encryption.

SHADOW DIRECTORY. A nonproduction directory on a target Web server to which the deployment event is copied. When the deployment event completes, this directory is moved to a live directory on the target Web server.

SITE MANAGER. The member of a Web team who is responsible for configuring and using page generation , deployment, and Rational NetDeploy triggers.

SOFTWARE DEVELOPER. The member of a Web team who writes code for JavaBeans, COM objects, and ASP / JSP pages that are not contained in ASP or JSP templates.

SYNCHRONIZATION TASK. A scheduled process that synchronizes the content VOB with the V / 5 CMS database. When this process runs, it removes any files from the ClearCase VOB whose corresponding V / 5 CMS database record has been deleted. The administrator specifies how often this process runs.

TASK. In the V / 5 Production Center, a work item to be completed. Tasks may be project tasks or workflow tasks. ContentStudio supports an integration of project tasks with ClearQuest content change requests to allow unified project management of code and content change requests.

TEMPLATE. The mechanism in the V / 5 E-business Platform software that defines the structure, format, and functionality of Web pages. Templates combine ASP or JSP template scripting with common Web languages such as HTML. This combination lets them act as formatting guides or outlines for Web pages and as small content-processing applications. There are four types of templates:

- ➤ A component template is a mechanism that provides content to be included in multiple pages, that is updated independently of those pages, and that avoids regeneration of those pages if the component content changes.
- ➤ An index template generates pages that contain links to individual pages. For example, you can create a front page for your business news site that includes links to individual news stories.
- ➤ An item template defines the structural presentation and format for individual stories, reviews, catalog items, and so on. It is typically applied to individual pieces of content.
- ➤ A library template defines frequently used structural processes and format commands that you can include in individual item and index templates.

TEMPLATE DEVELOPER. The member of a Web team who designs and implements Vignette ASP or JSP templates for the CDA and CMA. The template developer can use the ClearCase / CMS integration to develop templates residing in the code VOB.

TESTER. The member of a Web team who performs quality testing on a staging server before final deployment to the target Web servers. Typically, the tester uses ClearQuest to create content change requests for content problems and change requests for software defects.

TOPOLOGY. The comprehensive definition of which artifacts are deployable and where they are to be deployed.

URL (UNIFORM RESOURCE LOCATOR). The address that defines the route to a file on the Web or another Internet facility. URLs are typed into the browser to access Web pages and also are embedded in these pages to provide the hypertext links to other pages.

VOB. See code VOB and content VOB.

WEB APPLICATION. The code and content that is deployed from ClearCase VOBs to one or more target Web servers.

WORKFLOW. A sequence of tasks in the V / 5 Production Center. For example, a workflow can consist of writing, editing, and launching a content record.