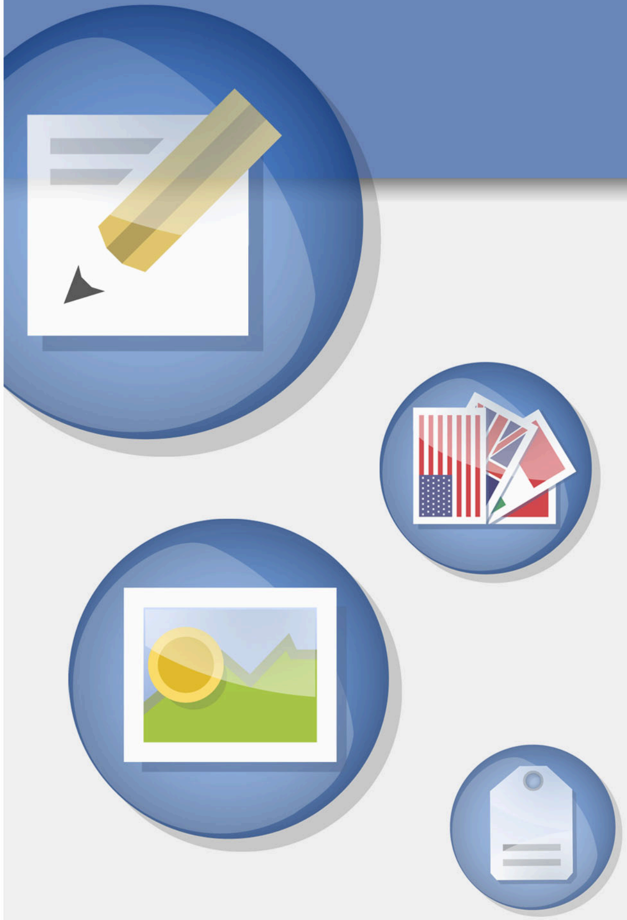
FatWire | Content Server 7

Version 7.0.3



Administrator's Guide

Document Revision Date: Nov. 5, 2007

FatWire®
SOFTWARE

You may not download or otherwise export or reexport this Program, its Documentation, or any underlying information or technology except in full compliance with all United States and other applicable laws and regulations, including without limitations the United States Export Administration Act, the Trading with the Enemy Act, the International Emergency Economic Powers Act and any regulations thereunder. Any transfer of technical data outside the United States by any means, including the Internet, is an export control requirement under U.S. law. In particular, but without limitation, none of the Program, its Documentation, or underlying information of technology may be downloaded or otherwise exported or reexported (i) into (or to a national or resident, wherever located, of) Cuba, Libya, North Korea, Iran, Iraq, Sudan, Syria, or any other country to which the U.S. prohibits exports of goods or technical data; or (ii) to anyone on the U.S. Treasury Department's Specially Designated Nationals List or the Table of Denial Orders issued by the Department of Commerce. By downloading or using the Program or its Documentation, you are agreeing to the foregoing and you are representing and warranting that you are not located in, under the control of, or a national or resident of any such country or on any such list or table. In addition, if the Program or Documentation is identified as Domestic Only or Not-for-Export (for example, on the box, media, in the installation process, during the download process, or in the Documentation), then except for export to Canada for use in Canada by Canadian citizens, the Program, Documentation, and any underlying information or technology may not be exported outside the United States or to any foreign entity or "foreign person" as defined by U.S. Government regulations, including without limitation, anyone who is not a citizen, national, or lawful permanent resident of the United States. By using this Program and Documentation, you are agreeing to the foregoing and you are representing and warranting that you are not a "foreign person" or under the control of a "foreign person."

FatWire Content Server Administrator's Guide
Document Revision Date: Nov. 5, 2007
Product Version: 7.0.3

FatWire Technical Support
www.fatwire.com/Support

FatWire Headquarters
FatWire Corporation
330 Old Country Road
Suite 207
Mineola, NY 11501
www.fatwire.com

Table of Contents

Part 1. Introduction

Part 2. CM Site Configuration Procedures

Part 3. Managing the Publishing System

Part 4. System Configuration Procedures

Appendices

About This Guide

This guide describes the Content Server administrator’s environment, accessible from the Advanced interface. It begins with an overview of Content Server, its add-on products, content management concepts, and the process you will follow to create your content management (CM) framework.

The rest of the guide explains your main tasks: how to configure Content Server so that writers, editors, and other content providers can electronically create and manage content, participate in workflows, and publish to the online site. Other sections describe Content Server’s database and utilities for tuning and maintaining Content Server installations.

Who Should Use This Guide

This reference is written especially for administrators. It is assumed that administrators have a clear knowledge of their company’s business needs, and a basic understanding of their roles in the site development process. This guide is also useful to developers, who collaborate with administrators by designing and coding the data model, presentation templates, security system, and caching options.

Administrators are not required to have programming experience, although a technical background is assumed. Developers, however, must know Java, JavaScript Pages, XML, and HTML.

Note

The examples and procedures throughout this guide are based on the Advanced interface. The Dashboard interface does not provide access to Content Server’s administrative functions.

How This Guide Is Organized

Information in this guide is organized by parts, where each part presents a set of chapters that are related to a particular task or function.

Part 1, “Introduction” presents an overview of Content Server, its add-on products, the interface you will be using, and the process you will follow to create your content management (CM) framework.

Part 2, “CM Site Configuration Procedures” describes Content Server’s administrative environment. It provides procedures for creating and managing users, assembling and organizing content management sites (the back end of the online site), managing users’ access to CM sites, creating workflow processes, and replicating sites.

Part 3, “Managing the Publishing System” describes Content Server’s publishing and approval systems. In particular, it provides an in-depth discussion of static publishing. This part also contains procedures for configuring dynamic, static, and Export to XML publishing processes.

Part 4, “System Configuration Procedures” presents information about configuring user interfaces, managing the publishing system, revision tracking, and implementing search engines.

The final part, “Appendices,” contains reference information about system defaults, the Content Server database, sample site configurations, LDAP authentication and deployment options.

At the end of this guide is a standard index and an index of procedures to help you quickly locate procedures for completing administrative tasks from the Advanced interface.

As you read this guide, keep a copy of the *Content Server Property Files Reference* handy. The *Property Files Reference* provides detailed descriptions of the properties that are mentioned in this guide.

Related Publications

The FatWire library includes publications written for Content Server users, administrators, and developers. The publications are provided as product manuals with your Content Server installation. They are also posted on the Web, by version number, at the following URL:

```
http://e-docs.fatwire.com/CS
```

Check the site regularly for updates.

Other publications, such as case studies and white papers, provide information about Content Server’s feature set and its business applications. To obtain these publications, contact `sales@fatwire.com`.

Part 1
Introduction

This part of the administrator's guide introduces you to Content Server, its system of managing content, the underlying concepts and constructs, and how they are implemented. The chapters in this part of the guide also provide you with guidelines for modeling your online site in the Content Server interface, as well as planning the configuration tasks that you will routinely perform in the Content Server environment.

This part contains the following chapters:

Chapter 1

Overview

Welcome to Content Server, FatWire’s content management system. In this chapter, you will learn about CM sites in the Content Server environment. The sites can be thought of as the back end of your online site, a place where business users work to create their electronic assets, manage them, and deploy them to their audiences.

As the administrator, you will be working with CM sites throughout your job, configuring the sites, replicating them, and managing their day-to-day use. Because they are a critical part of your work, you need to understand what CM sites are, and how they relate to the online site. This chapter provides an overview.

This chapter also outlines the administrator’s job in the Content Server environment and offers guidelines for planning configuration tasks.

This chapter contains the following sections:

Introduction

As the Content Server administrator, your job is to build the foundation of your online site, specifically, its back end. The back end itself is a site (or set of sites) in the Content Server installation—a site where business users work to create their electronic assets, manage them, and deploy them to their audiences. Building the back end involves configuring the business users, linking them to content-entry forms as well as other authoring tools, and providing the users with publishing and delivery systems for serving the online site to browsers.

When the back end is configured, its users log on and fill in content-entry forms. The content they enter is then saved to tables in the Content Server database. When ready for delivery, the content is drawn programmatically from the database tables, formatted, and served to browsers as the online site. Figures 1 and 2 illustrate the process flow, using a dynamic publishing scenario.

In our scenario, Figure 1 depicts the smallest unit that can function as a back end: a user who has permissions to write to the Content Server database and to invoke the publishing-delivery systems. This unit, Content Server defines as a “CM site.” Content Server imposes no limit on the number of CM sites that you can configure for the back end of the online site, or the number of ways in which the sites can be configured.

Throughout your job as the Content Server administrator, you will be working with CM sites—configuring them, replicating them, and managing their day-to-day use. Because they are a critical part of your work, you need to understand how CM sites are defined, how they must be configured, and how they relate to the online site. This chapter provides an overview.

Online Sites

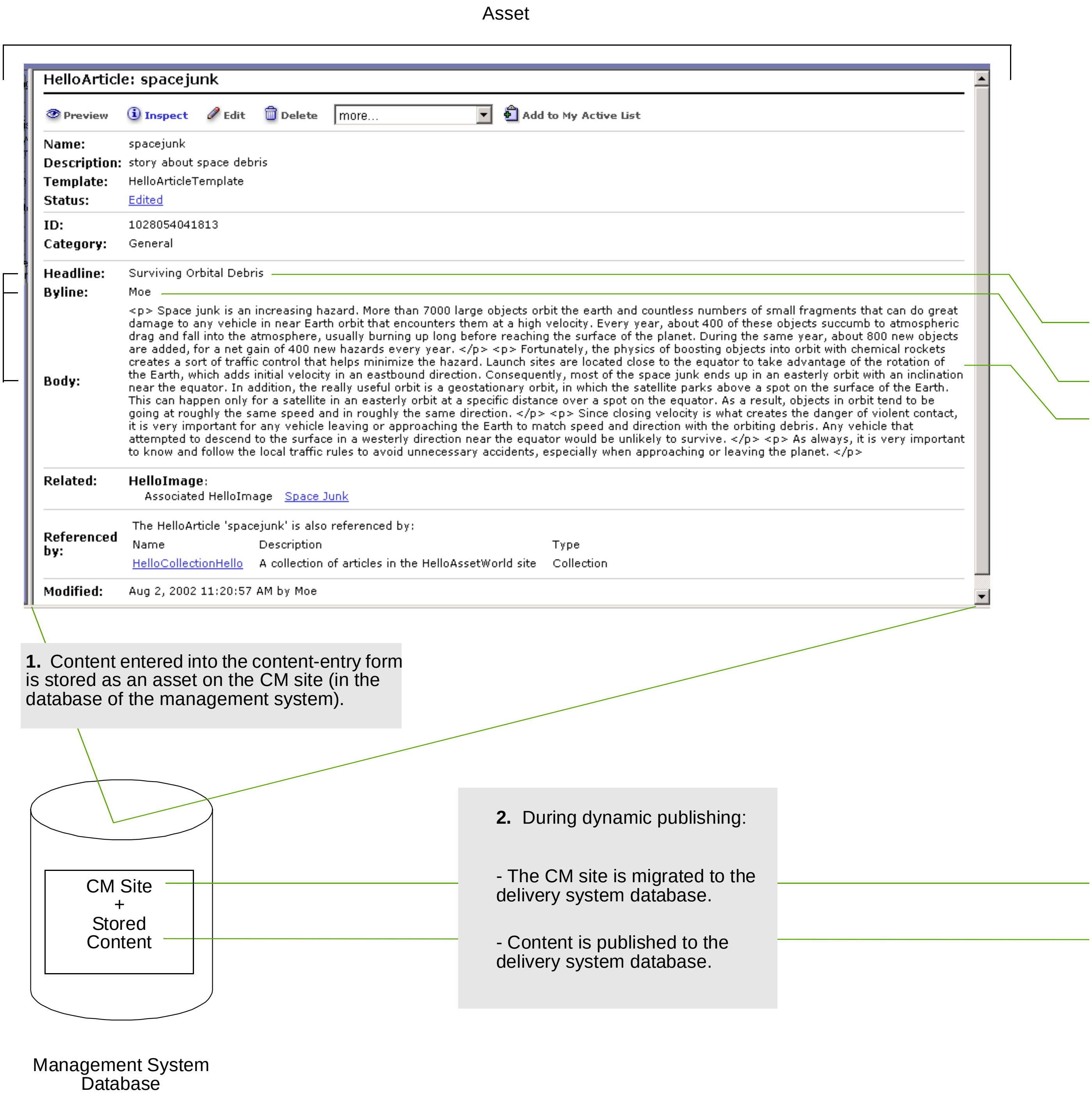
Although most Internet users don’t need a definition of “online site,” our objective in providing one is to first establish what we mean by an online site that is powered by Content Server. Later in this chapter, we explain how online sites can be modeled by CM sites.

A CS-powered online site is the set of pages that an organization displays to its target audience of customers, clients, and casual visitors. The online site can be an Internet site or a portal. It can be accessible to the general public or it can be a password-protected site. It can also be a completely exclusive site, such as a corporate intranet or departmental network, operating strictly within the private domain.

Regardless of its nature, an online site originates from either a single CM site, or many CM sites, depending on which model you choose. Throughout our product guides, we use the term “online site” generically to refer to websites and portals, both of which are supported in this release.

Figure 1: CM Site

This asset was created by a user populating the fields of the content-entry form below.



HelloArticle: spacejunk

Name:	spacejunk
Description:	story about space debris
Template:	HelloArticleTemplate
Status:	Edited
ID:	1028054041813
Category:	General
Headline:	Surviving Orbital Debris
Byline:	Moe
Body:	<p> Space junk is an increasing hazard. More than 7000 large objects orbit the earth and countless numbers of small fragments that can do great damage to any vehicle in near Earth orbit that encounters them at a high velocity. Every year, about 400 of these objects succumb to atmospheric drag and fall into the atmosphere, usually burning up long before reaching the surface of the planet. During the same year, about 800 new objects are added, for a net gain of 400 new hazards every year. </p> <p> Fortunately, the physics of boosting objects into orbit with chemical rockets creates a sort of traffic control that helps minimize the hazard. Launch sites are located close to the equator to take advantage of the rotation of the Earth, which adds initial velocity in an eastbound direction. Consequently, most of the space junk ends up in an easterly orbit with an inclination near the equator. In addition, the really useful orbit is a geostationary orbit, in which the satellite parks above a spot on the surface of the Earth. This can happen only for a satellite in an easterly orbit at a specific distance over a spot on the equator. As a result, objects in orbit tend to be going at roughly the same speed and in roughly the same direction. </p> <p> Since closing velocity is what creates the danger of violent contact, it is very important for any vehicle leaving or approaching the Earth to match speed and direction with the orbiting debris. Any vehicle that attempted to descend to the surface in a westerly direction near the equator would be unlikely to survive. </p> <p> As always, it is very important to know and follow the local traffic rules to avoid unnecessary accidents, especially when approaching or leaving the planet. </p>
Related:	**HelloImage**: Associated HelloImage Space Junk
Referenced by:	The HelloArticle 'spacejunk' is also referenced by: Name / Description / Type HelloCollectionHello / A collection of articles in the HelloAssetWorld site / Collection
Modified:	Aug 2, 2002 11:20:57 AM by Moe

1. Content entered into the content-entry form is stored as an asset on the CM site (in the database of the management system).

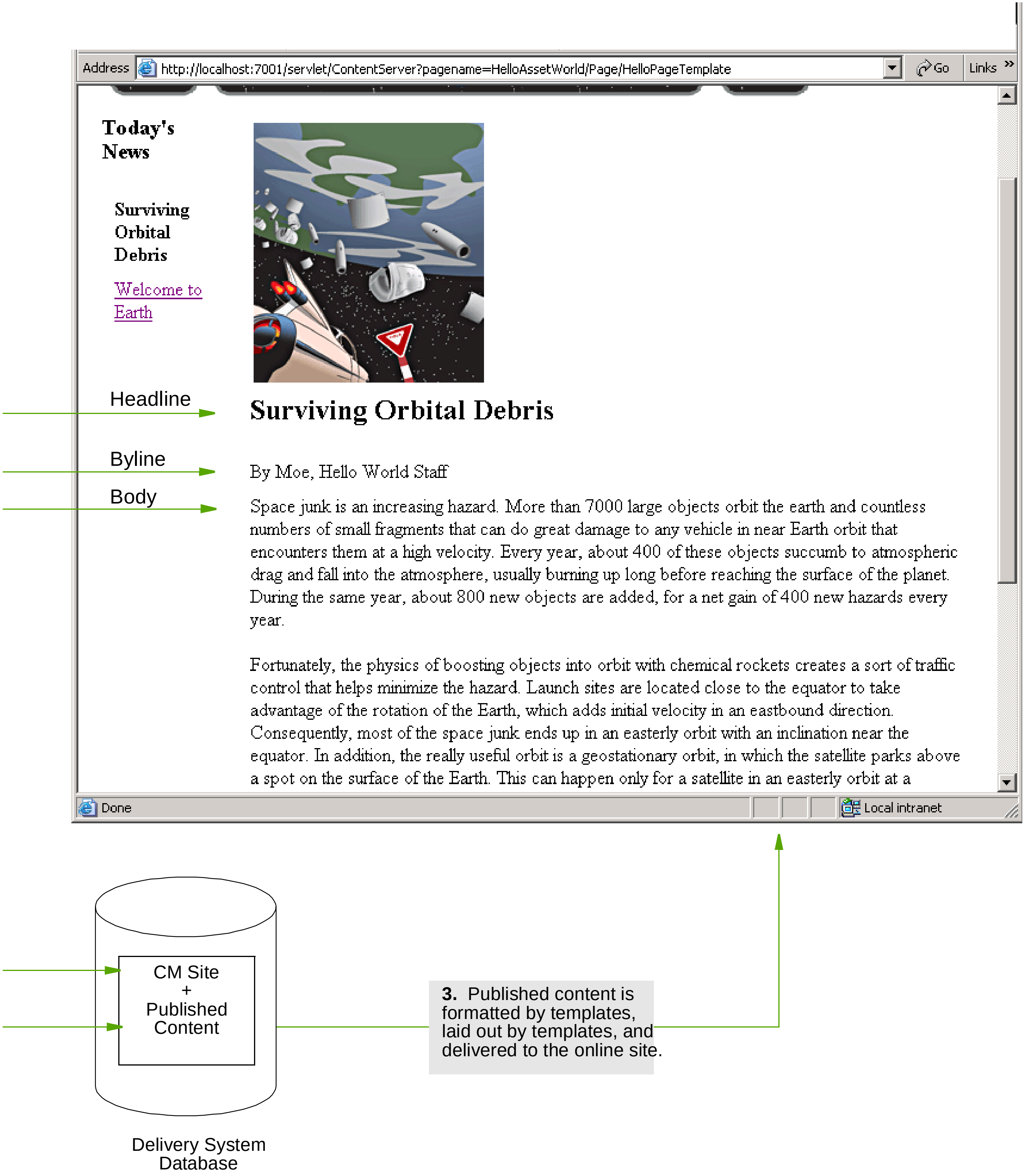
2. During dynamic publishing:

- The CM site is migrated to the delivery system database.

- Content is published to the delivery system database.

Figure 2: Delivery System and Online Site

This online site, a single page, was created from the asset shown in Figure 1.



Surviving Orbital Debris

By Moe, Hello World Staff

Space junk is an increasing hazard. More than 7000 large objects orbit the earth and countless numbers of small fragments that can do great damage to any vehicle in near Earth orbit that encounters them at a high velocity. Every year, about 400 of these objects succumb to atmospheric drag and fall into the atmosphere, usually burning up long before reaching the surface of the planet. During the same year, about 800 new objects are added, for a net gain of 400 new hazards every year.

Fortunately, the physics of boosting objects into orbit with chemical rockets creates a sort of traffic control that helps minimize the hazard. Launch sites are located close to the equator to take advantage of the rotation of the Earth, which adds initial velocity in an eastbound direction. Consequently, most of the space junk ends up in an easterly orbit with an inclination near the equator. In addition, the really useful orbit is a geostationary orbit, in which the satellite parks above a spot on the surface of the Earth. This can happen only for a satellite in an easterly orbit at a

3. Published content is formatted by templates, laid out by templates, and delivered to the online site.

CM Sites

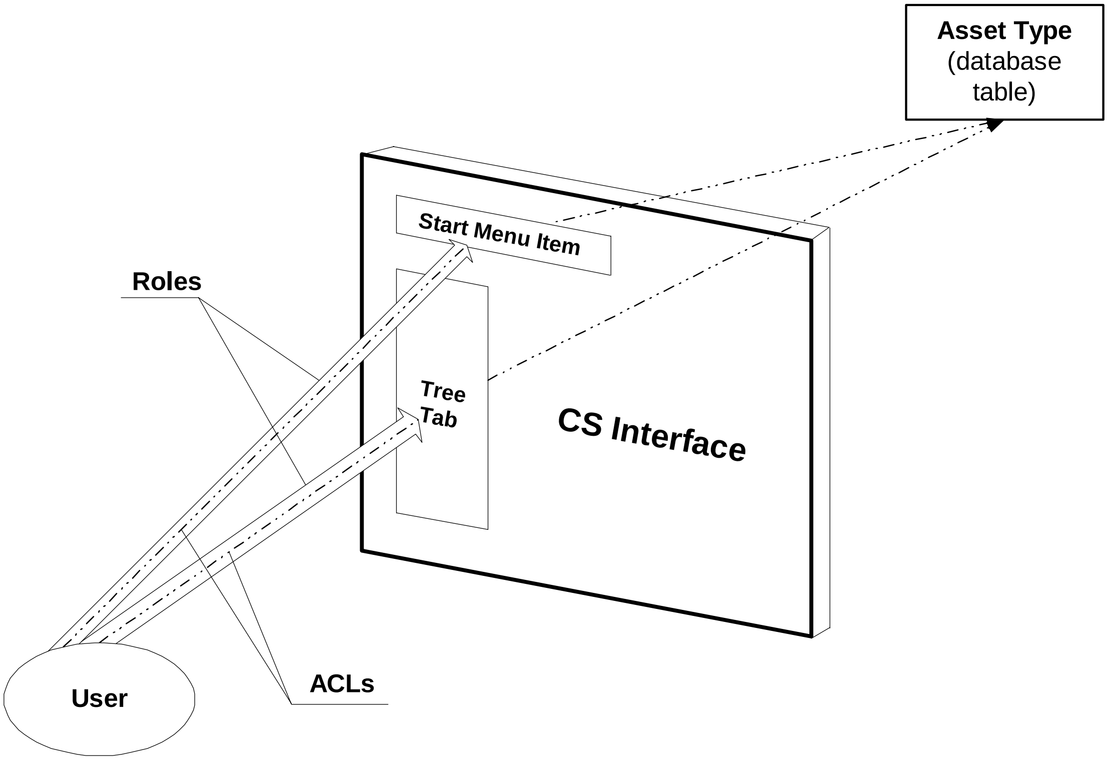
A CM site is a content management unit within Content Server. It is the source of content for the online site and can represent either an entire online site or one of its sections.

Note

From this point forward, when the term “site’ is used without a qualifier, it means “CM site.”

A site is an object that you must configure in order to (1) define the authors and managers of the online site, and (2) provide them with the permissions and content management tools they will need: content-entry forms, content-rendering templates, workflow processes, start menu items, publishing methods, and a delivery system. The process of configuring a site involves creating not only the site components, but also associating the components with each other. Making the associations defines a site, similar to the one shown in Figure 3.

Figure 3: CM Site



The site is identified by a name (also configured by the administrator), stored in Content Server’s database, and listed in authorized users’ interfaces. Users with the correct roles have access to the site. Within the site, roles manage the users’ access to specific functions (such as a “Start Menu Item”) in the Content Server interface. ACLs manage the user’s permissions to content (database tables). Through the interface functions (made accessible by roles), the user is able to actuate his permissions (as defined in the ACLs), and therefore operate on the database tables in order to author and manage specific types of content.

A Content Server system typically has many such sites, each one unique in its function and composition. When users log in to Content Server, they must also select a site where they will work. The site can be independent of other sites, or it can share assets with other sites, providing that the sites have a common set of users.

Note

In this release, Content Server offers a site-replication utility called "Site Launcher" that speeds up the site creation process. Instead of creating sites from scratch, you can replicate established sites as necessary, modify the replicates, and spin them off as new sites in the Content Server environment.

Configuration Components

Configuring a site involves using Content Server's administrator interface to access a system-wide configuration pool, select (or create) components that will make up the site, and associate the components with each other so they can function together to produce either the online site or one of its sections.

Table 1 lists site components, most of which are required. The optional components vary according to business needs and user preferences. Whereas developers are responsible for the code-based components, the administrator is responsible for all other components.

Site components are generally complex constructs, especially the data model. They are described in the sections that follow. Data modeling is described in detail in the *Content Server Developer's Guide*.

Table 1: Site Components

Component	Required	Created by	See page …
CM site definition	✓	CS administrator	24
Data model	✓	Developers	25
Users	✓	CS administrator	26
Roles	✓	CS administrator	27
Start menu items	✓	CS or its administrator	28
Workflow processes	–	CS administrator	28
Publishing system	✓	CS administrator	29
User interface options	–	CS administrator	29

CM Site Definition

A CM site definition is one of the components of a CM site. It consists of a site name and optionally a description, both specified by the administrator. The site name can represent an online site, a business topic, the work of a department, or yet another type of content. In any case, the site name establishes a business theme that users in the site are responsible for developing and maintaining.

When you specify a site name, Content Server creates a node to represent the site in its interface. It also appends several default sub-nodes for linking components to the site: an “Asset Types” sub-node for linking the data model to the site, a “Users” sub-node for linking users to the site, and two other nodes for enabling the optional user interfaces Desktop and DocLink. Descriptions of these components are given in the next sections.

For content providers, the site is a visibility control mechanism. The site provides authorized users access to certain content in the Content Server installation. When a site is properly configured, its name is displayed in the content providers’ interfaces, allowing the content providers to select the site and navigate within it according to their roles and permissions.

Data Model

The data model is a component of a site. It comprises a set of asset types (database tables) and asset type definitions, coded by developers for content providers’ use. To help developers equip content providers with the broadest possible set of content management options, Content Server supports three kinds of asset types.

- Content asset types, which are structured repositories for content and by design reflect the business theme established by the site definition. For example, if you defined a site named “Social Events,” suitable content asset types could be “Selected Moments in History,” “The 20th Century’s Greatest Events,” and so on, since they pertain to the theme suggested by the site name.

  From asset types, developers code asset type definitions. These are expressed in the CS interface as content-entry forms (such as the one in Figure 1), whose fields prompt users for information that will be delivered to the online site (or reserved for internal use, if necessary). The set of fields defines the asset type; users’ field entries define the asset (an instance of the asset type).
- Design asset types, which are used by developers to code template assets, which render the content assets. (We distinguish template assets from content assets in order to distinguish presentation code from content.)
- Management asset types, which are also used by developers to create tools such as simple searches and database queries that help content providers manage their content.

Ordinarily, developers test the asset types they create and pass them on to you so that you can link them to the site definition (through the “Asset Types” node) and complete another step in the site configuration process.

Note

Content Server provides a number of default asset types and allows developers to create their own. The asset types, their definitions, and the assets themselves are stored in the Content Server database as tables or table entries, and loosely referred to as “content” in this guide.

For information about data modeling, see the *Content Server Developer’s Guide*. We recommend that administrators read the guide to gain a basic understanding of asset types.

Users and Their Table-Level Permissions

Users are site components. In this guide, they are often referred to as “content providers”—people who use the developers’ data model to author and manage content.

Content providers are the subject matter experts. They can be:

- Authors of online content. Copywriters and designers would fall into this group.
- Reviewers, who examine and edit the content that other users submit to them in order to ensure its quality. Examples of content reviewers include editors and art directors who review and modify the copy and designs that are submitted to them.
- Content publishers, who ensure that content is ready to be delivered to the online site, and approve the content for delivery. An editor-in-chief could be a content publisher.
- Content managers, who oversee the authoring, review, and publishing processes.

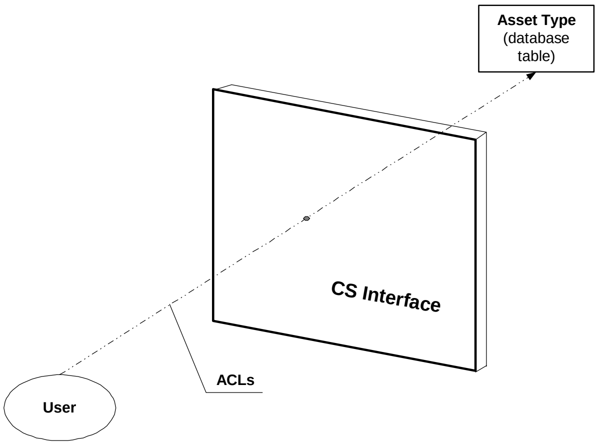
Each content provider must be identified to Content Server through a user account, which consists of a login name, a password, and Access Control Lists (ACLs), the foundation of Content Server’s security system.

An ACL is a set of permissions to database tables. The permissions (such as read and write) are granted when the same ACL is assigned to both the table and the user. If no ACLs are common to a table and a user, the user has no permissions to the table.

For example, the system table named `SystemUsers` contains user account information. The table is assigned three ACLs: SiteGod, UserReader, and UserEditor. If a user is assigned one of the ACLs—UserReader, in our example—he can read the table. If the same user is assigned a second ACL—UserEditor—he can also edit the table. If the user is not assigned any of the ACLs, he has no permissions to the table.

Be aware, that while ACLs give the user *permissions* to operate on tables, they do not give the user *the means* to operate on the tables. For example, in our preceding scenario, the user’s permissions to read and edit the table translate into permissions to use the “view” and “edit” functions in the Content Server interface. However, the functions are hidden from the user, unless the same roles are assigned to both the user and the functions. (For information about roles, see the next section, “Roles,” on page 27.)

ACLs give the user permission (but not the means) to operate on database tables



In general, Content Server uses ACLs at two levels:

- At the system security level, to provide authentication functionality and therefore prevent hackers from entering the Content Server environment.
- At the interface level, to control users’ permissions to database tables and, therefore, control the ability to use (but not view) interface functions through which the permissions are actuated.

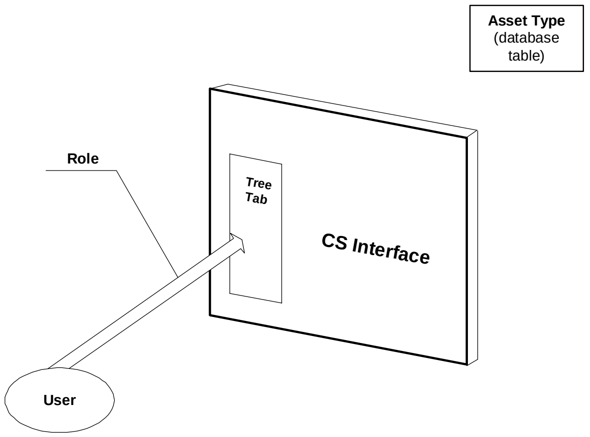
Content Server provides a number of default system ACLs and pre-assigns them to system tables. You can re-use the ACLs, or configure your own and assign them to custom tables, as necessary. For more information about ACLs, see Chapter 4, “Working with ACLs and Roles” and Appendix A, “System Defaults.”

Roles

Whereas ACLs give the user permission to operate on database tables, roles give the user the means to operate on database tables. Roles determine whether the user has access to a site, and whether interface functions, such as “edit,” “delete,” and “start workflow” are exposed in the user’s interface. If the functions are hidden, the user’s permissions to database tables (as specified in the ACLs) cannot be actuated, leaving the user unable to operate on the tables. Roles also define groups of users, such as “authors” and “editors.” They are used to describe the groups’ permissions (and therefore the users’ permissions) to sites, the sites’ content, to collateral (such as start menus for creating and locating content), and to workflow processes.

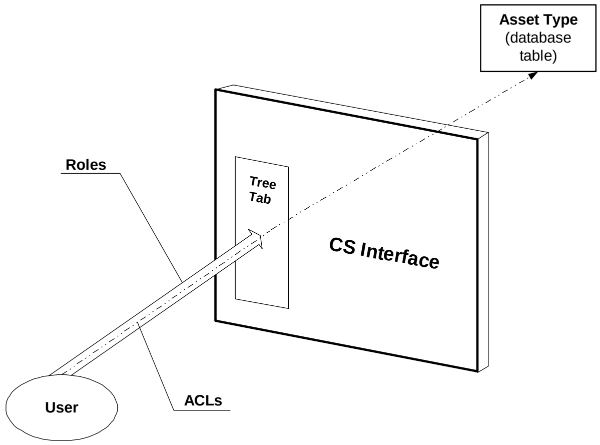
Roles are implemented in the same way as ACLs; that is, for a function to be displayed in the user’s interface, the function and the user must be assigned the same role. To illustrate, we continue our previous scenario (on page 26), where a user is given editorial permissions to the `SystemUser` table through the UserReader and UserEditor ACLs. To exercise those permissions, the user needs access to the “edit” function in the Content Server interface. The “edit” function is located in the “Admin” tab. To view the “Admin” tab, however, the user must be assigned the same role as the tab. By default, the tab is assigned the GeneralAdmin role; the same role must also be assigned to the user.

Roles, without ACLs, give the user the means (but not the permission) to operate on database tables



To summarize, the user gains full access to the database table only when he is assigned the ACLs of the table, *and* the role of the “Admin” tab. In practice, as in our scenario, roles and ACLs must be compatibly assigned—role assignments must support the permissions that are granted by ACL assignments.

ACLs + roles give the user the permission *and* the means to operate on database tables



In addition to displaying interface functions to site users, roles provide a way of grouping users according to their responsibilities on the site. Users with similar responsibilities can be assigned the same role(s). For example, administrative users need access to the “Admin” tab. All the administrators can be assigned the GeneralAdmin role and thereby be given access to the “Admin” tab.

Content Server defines several default system roles, all of which are pre-assigned to various functions in the CS interface. Content Server also allows you to configure and assign your own roles. When choosing role names, consider the responsibilities of the users on the site and select the role names accordingly. Note that unlike ACLs (which are mapped to database tables), roles are mapped to sites (which means that roles must be assigned on a per-user, per-site basis).

Note

Unlike ACLs, roles are exposed to content providers for enlisting other content providers into workflow processes.

Start Menu Items

A start menu item is a site component. Start menu items provide a way of coupling a user with the asset types he is to work with on the site. The coupling is accomplished by means of roles.

Content Server defines several start menu items: “New” which allows the user to create assets on the site; “Search” which allows the user to look for assets on the site; “CS-Desktop” which enables Microsoft Word as an alternative interface to CS content-entry forms; and “CS-DocLink” which enables the document management interface.

A start menu item specifies:

- The site(s) to which the start menu item applies
- The asset type that users on the sites can work with
- The roles that are allowed to create or search for assets of those types
- The workflow processes (if any) in which the roles can participate

In other words, start menu items determine which roles can create and search for assets of a specific type on a site, and which workflow processes the roles can participate in.

Content Server creates start menu items automatically and gives you the option to configure your own.

Workflow Processes

Workflow processes are optional constructs, mandated only by business needs, and used to regulate collaborations among content providers.

A workflow process enlists qualified users with complementary expertise to perform a sequence of operations that starts with the creation of a content asset, continues with review of the asset, and culminates in approval of the asset. Once approved, the asset is published to the delivery system by one of the users and finally, delivered as content to the online site.

Once workflows are configured, they can be specified in start menu items as processes that automatically begin the moment newly created assets are saved. Workflow processes can also be omitted from start menu items, to be invoked by users as necessary. For more information about workflow processes, see Chapter 9, “Creating and Managing Workflow Processes.”

Publishing System

The publishing system is a site component. The publishing system gives users the means to migrate sites and their content from one system to another. Two basic publishing methods can be configured: dynamic publishing (Mirror to Server) and static publishing (Export to Disk, and export to XML). The structure of the site (that is, the database schema) is migrated by means of the dynamic publishing method. The content itself can be published either statically or dynamically, at the administrator’s discretion.

User Interface Options

Content Server supports a number of optional interfaces through which users can interact with content:

- **Content Server Desktop (CS-Desktop)**, which provides users with the familiar Microsoft Word interface as an alternative to the content-entry forms that are native to Content Server. Users can create their content in MS-Word documents, import the Word documents as assets into the CS database, and from there, recall and edit the documents.

  Note that CS-Desktop requires the content in Word documents to be structured. For example, when using Content Server Desktop to author content, the user opens a Word document, enters content, and structures the content by tagging it with the same field names as defined in the equivalent content-entry form. The tagging utility is embedded in the Word interface, and the selection of fields is determined by the administrator. When the Word document is saved, the content in its fields is parsed to the appropriate database table(s).
- **Content Server DocLink (CS-DocLink)**, which supports unstructured content in the flex asset family.

  CS-DocLink provides a drag-and-drop interface for uploading and downloading unstructured content—documents, graphics, or other single binary files that are managed as flex assets. CS-DocLink also presents the hierarchical structure of any flex asset family in the Content Server database as folders and files in the Windows Explorer application.
- **InSite interface**, which supports the editing of content directly on the rendered page. Regular Content Server users can make quick edits in context, while infrequent users can accomplish their work without having to learn the Content Server interface.
- Portal interface, which re-interprets the Content Server interface as a set of portlets in the user’s workspace. One set of portlets is provided for the management of structured content, another set for the management of documents.

Summary

It is important to understand that *the components of a site are not site-specific; the associations among them are.*

Site components are constructs that either exist or must be created in a system-wide pool. From this pool, you select the components that must interact with each other in order to form a site; you then associate the components with each other, and assign them a site name. Because the components are system-wide, they are re-usable—components that are used to form one site can be used to form other sites.

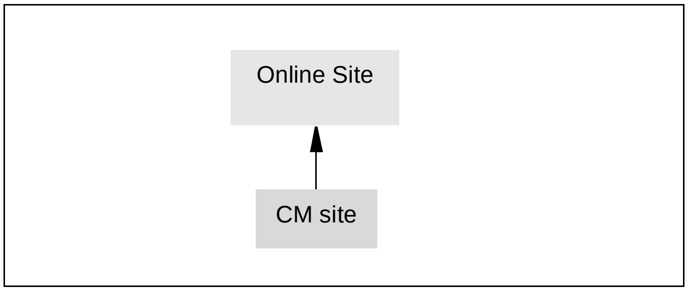
Note that to content providers, site components appear to be site-specific. However, any assets that content providers create on the site are indeed specific to that site. The assets can be copied to other sites, or shared among them.

The options of sharing and copying site components play an important role in the modeling and replication of online sites, as explained in the next section "Content Management Models."

Content Management Models

As the administrator, one of your biggest decisions, whether you make it alone or with collaborators, is how to model the online site in the Content Server interface: as a single site or a set of sites. Which content management model you choose depends largely on the size of your online site and the nature of its content.

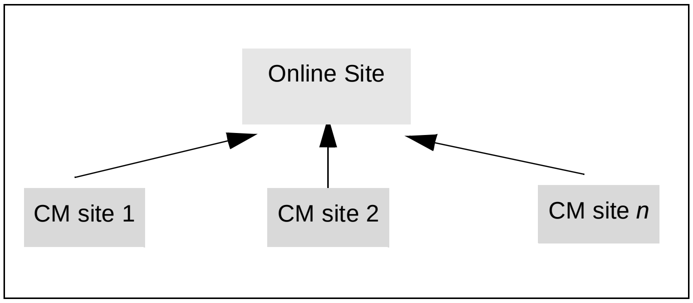
1:1 Model



The 1:1 model maps the online site directly to a single CM site. The CM site is the sole source of the online site.

The 1:1 model works efficiently for small online sites, where content tends to be limited, uniform, and managed by few users. For enterprise-level sites, however, pages often number in the millions and differ significantly in subject matter, content, presentation, and scope. Organizing large sites requires a model that can handle the complexities of the task.

1:*n* Model



The 1:*n* model maps the online site to multiple CM sites. Each section of the online site, whether logical or physical, maps to a certain CM site as its source. The CM sites can function independently of each other or overlap each other by sharing components and content.

For example, in a certain catalog site, content contributors who enter data about household goods never enter data about yard goods, so two separate sites are used to represent household goods and yard goods. Similarly, in a publication site where writers work independently of each other, the sports writers have a site that represents the sports news section, while the financial writers have a separate site that represents the financial news section. In each design, two sites contribute to one online site.

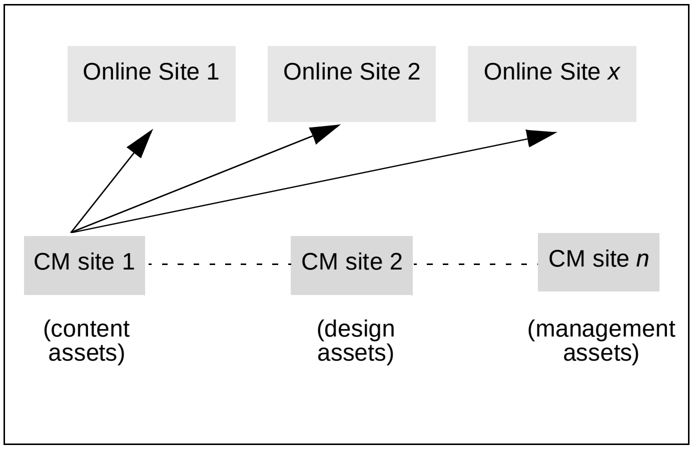
In a 1:*n* model, each site is unique; it has its own content types, templates, content providers, roles, workflow processes, and publishing mechanisms. Each site is the source of content for its corresponding online section.

When using a 1: *n* model, administrators have the option to configure sites to be independent of each other or to overlap each other by sharing components and content. An independent model works well when certain content needs to be segregated from other content. For example, the content is sensitive and must be handled only by certain content providers; or the content is so dissimilar that it needs to be handled by different specialists.

Overlapping sites, on the other hand, are used to support collaborations among remotely related content providers, such as those residing in different departments, or business units.

For example, consider the following scenario. A growing e-business specializing in mountain climbing disseminates mountain-climbing news and publishes a photograph album. It also sells gear and answers FAQs. The online site is clearly segmented into four sections: news, photos, gear, and FAQs. On many occasions, however, the news and photo sections share photographic content. In this scenario, one of your options is to create four sites: news, photos, gear, and FAQs such that news and photos share content. (Another one of your options is to create three independent sites: one for news and photos, one for gear, and one for FAQs.)

x:*n* Model



The *x*:*n* model maps multiple online sites to multiple CM sites. Of the three possibilities, this model is the most complex, but offers the greatest advantage to enterprise-level e-businesses.

In this model, content assets are localized on a given site, while the design and management assets are configured on separate sites from which they are shared to all other sites. This type of design allows the administrator to separate content from presentation and business logic.

Note

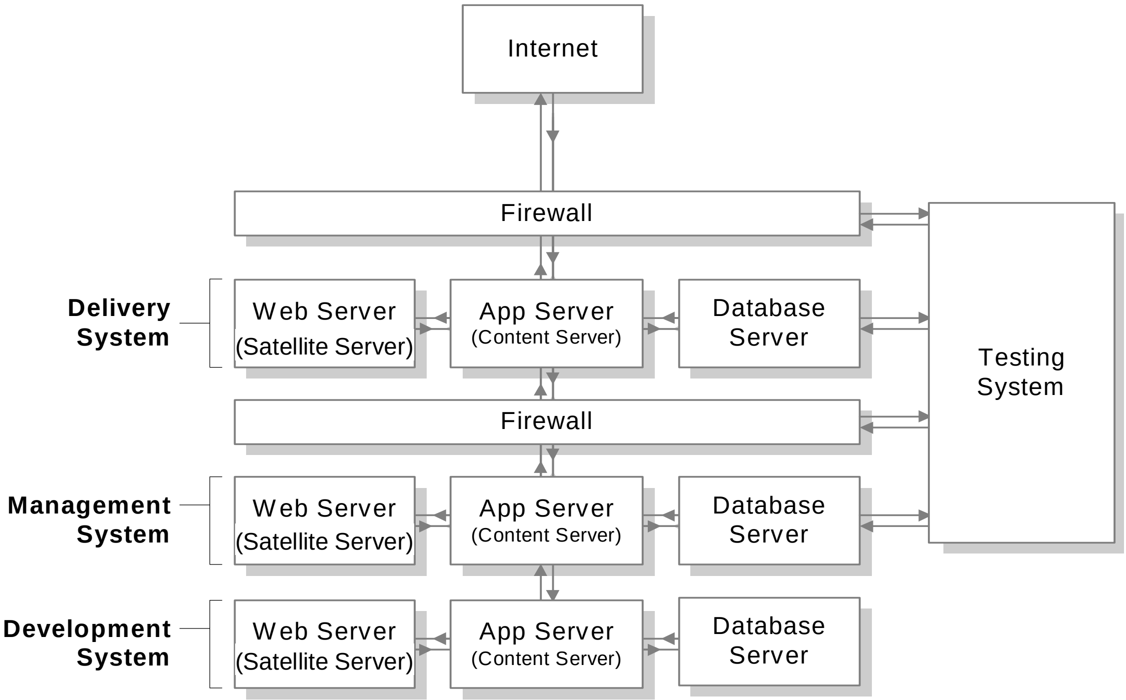
To help administrators and other users understand sites and their relation to the online site, Content Server provides sample sites. Each sample site maps 1:1 to its online site. For sample site specifications, see Appendix C, “Sample Site Configurations.” To log in to the sample sites, follow the login procedure in Chapter 2.

Content Server Environment

Site configuration is a collaborative process that takes place in a Content Server environment rather than an isolated Content Server system. An enterprise-level environment typically consists of four different CS-powered systems: development, management, delivery, and testing, as shown in Figure 4. Each system runs on its own database and at some point interacts with the other systems. One of the systems, testing, is an optional system that large organizations typically install for increased quality assurance. The remaining systems, however, are required.

As the CS administrator, you will work with all of the systems at one time or another. Your collaborators are developers, who work exclusively at the development system with you and with users whom you appoint as collaborators. This section summarizes the possible systems and your function in each of them.

Figure 4: Content Server Environment


Internet
Firewall
Delivery System
Web Server (Satellite Server)
App Server (Content Server)
Database Server
Firewall
Testing System
Management System
Web Server (Satellite Server)
App Server (Content Server)
Database Server
Development System
Web Server (Satellite Server)
App Server (Content Server)
Database Server

Note

The names of the systems in your CS environment might vary from the names used in this guide. Generally, the management system is also called “staging”; the delivery system is also called “production”; and the testing system is also called “QA” or “QA testing.”

- The development system is responsible for planning and creating the framework of the online site: the data model that will be used by the CM sites and the data presentation templates.
- The management system is the staging area, responsible for configuring site components (other than the data model), assembling the components (including the data model) into sites, making the sites available to users, and managing users’ day-to-day activities on the sites. The users’ responsibility is to develop and manage content for delivery to the online site.
- The delivery system is the production area, which receives content for the online site from the management system and serves it to the target audience.
- The testing system is where QA tests both the management and delivery systems, and the online site itself before its launch. When the testing system is absent, the development system doubles as the testing system.

The rest of this chapter describes your collaborators, your planning strategies, and your functions at each of the systems.

Administrator’s Job

Collaborating

Before development begins you typically collaborate with a team comprising many different specialists:

- Site designers
- XML and JSP developers
- Java application developers
- Database administrators
- System network administrators
- Marketers and advertising staff
- Business managers
- Product managers, if you are developing a commerce site
- Content providers

The job of the team is to establish functional requirements and design specifications for the management and delivery systems:

- Page design
- Caching strategy
- Security strategy
- Format vs. content
- Data model
- Content management model

Much of the preceding information is outlined in Chapter 1 of the *Content Server Developer’s Guide*. The information is presented in this *Administrator’s Guide* in the context of the administrator’s job.

Planning

Planning is critical, as the decisions that you and your collaborators make will determine how developers will code system security, the data model, and the delivery system. Your job, in particular, concerns the management system, as described below.

During the planning stage, your job is to establish with collaborators (mostly business managers) the requirements and design of the management system, and relay the information to developers. Items to consider are listed below. The list is not meant to be exhaustive, but to help you start gathering and organizing information that is most critical to a successful start of the development cycle.

- Content management model
    - Determine which of the models (1:1, *n*:1, and *x*:*n*) is most appropriate for your operation
    - Determine which content assets must be shared and which must copied. Do the same for template assets.
- Site replication

    Determine whether you will be replicating sites. If so, you must configure the source sites to conform to the requirements that are set by the replication process.

    Developers must know your decision in order to code correctly. For example, if sites are to be replicated, the names of templates that will be copied to the new sites must not be hard-coded.
- Data modeling
    - What types of assets should be created
    - Which asset types should use revision tracking
    - Which users must have access to which asset types? on which sites?
    - What types of templates need to be created by developers
- User management methods

    In addition to its native user manager, Content Server supports LDAP plug-ins, both hierarchical and flat-schema (for the portal environment). Determine which method is appropriate for your environment. (Procedures in this guide use Content Server’s native user manager system throughout.)
- User management models

    Determine the following:
    - How many users are needed
    - Who the users are
    - What permissions the users must be given to the database tables:
        - Which system ACLs the users must be assigned (without ACLs, no user accounts can be created).
        - Whether custom ACLs need to be created.

            (Note that you may also need to create ACLs to assign to visitors of the online site in order to restrict them from accessing certain database tables.)
    - How many roles are needed

    - What types of roles are needed (writers, editors, illustrators). Make sure that the roles are compatible with the users’ ACLs.
    - To which users the roles should be assigned.

- Workflow processes

  Workflow processes can be optionally attached to asset types to ensure that newly created assets of those types are automatically engaged in workflow.

  - Should workflow be implemented
  - If so, which asset types need workflow processes
  - How must the workflow processes be designed
  - Which users and roles need to participate in the workflow processes

- Publishing system options
  - Which type of publishing needs to be configured—static or dynamic
  - Whether publishing schedules need to be approved

- User interface options

  Determine which interfaces the users will need:

  - CS-Desktop
  - DocLink
  - InSite interface
  - eWebEditPro
  - Portal interface

Use both this book and the *Content Server Developer's Guide* to help you make these decisions.

Development

When the specifications of the online and sites are established, you are ready to begin your work. As the Content Server administrator, you will be working on all systems in the CS environment during the development stage. (Once the online site is running, however, you will spend most of your time at the management system, a smaller fraction at the delivery system, and perhaps none at all at the development system.)

At each system, you have a specific set of jobs. While developers are responsible for the data model and other code-based components, you are responsible for all other aspects.

At the Development System

The development system is where coding takes place.

- Your job is to provide developers with the specifications of the management system and to assist with content management operations, such as creating sites.
- The developers’ job is to:
  - Create sites with the same names as those that will be used on the management system
  - Code the data model (bearing in mind site-replication requirements:

  - Content asset types
  - Design asset types
  - Management asset types
- Create sample assets of each type.
- Test the data model.
- Code templates and, in general, the framework of the online site.

Once the data model is complete and tested to the satisfaction of all collaborators, developers migrate the code to the management system.

Note

In your CS environment, developers might choose to not migrate the data model, but to re-create the data model on the management system.

Note that once development is complete, the development system continues to operate. One of its ongoing functions is to revise the data model in response to the evolving needs of online site visitors, content providers, and administrators.

At the Management System

Configuration tasks that require no coding are typically completed at the management system. Here, your administrative work is not collaborative unless you choose to make it so. For example, because no data modeling or coding takes place at the management system, you have no need for assistance from developers. However, you can appoint developers (or other users) to be the site or workflow administrators, to manage specific sites or workflow processes for the system.

Note

On occasion, you will need help from developers; for example, you might want to add custom functionality to the tree tabs in Content Server’s interface.

At the management system, your job is the following:

- Create the sites

Note

In this release of Content Server, you can quickly spin off new sites by using the newly added Site Launcher feature to replicate source sites. You can then modify the replicates as necessary. For information about site replication and selecting or creating the appropriate source sites, see Chapter 10, “Replicating CM Sites.”

- Configure the users
  - Create their ACLs in Content Server, as necessary

  - Create user accounts, either through Content Server, or external user managers, such as LDAP plug-ins
  - Create roles in Content Server, as necessary
  - Assign users their roles for each site
- Assemble the sites by
  - Associating the users with the sites
  - Associating the correct asset types and publishing system with the sites
- Enable the users by
  - Assigning users their roles for each site
  - Using roles to associate users and asset types with start menu items
  - Using roles to associate users with interface functions such as tree tabs
  - Creating tree tabs, as necessary
- Create and manage workflows
- Enable revision tracking
- Configure site and workflow administrators, as necessary
- Configure the publishing process
- Configure the users’ interfaces, as necessary

At the Delivery System

The delivery system is configured by developers and typically does not involve an administrator until the system is ready for use. For dynamic delivery, the delivery system is the entire Content Server installation. For static delivery, the delivery system is simply the web server component of the Content Server installation.

At the Testing System

The testing system is where QA analysts test the performance of both the management system and the delivery system, as well as the online site itself before its launch. If the testing system is absent, the development system doubles as the testing system. Ordinarily, you are not involved in testing.

Implementation

When the management and delivery systems are in use and the online site is running, you work almost exclusively at the management system, where you:

- Manage the users
  - Adjust their permissions to database tables
  - Add, delete, and re-organize users
  - Create workflows
  - Configure the users’ interfaces
- Manage the publishing system
- And otherwise respond to the needs of online site visitors, content providers, and the e-business

The Focus of This Guide

This guide focuses on the configuration tasks that you execute at the management system during the development stage, as described in "Development," on page 35. It is assumed that the data model resides on the management system by the time you begin your configuration tasks.

Administrator's Prerequisites

To function in the jobs that were described in the previous section, you must be equipped with a certain amount of information. This section summarizes the information you need to know or be familiar with.

- Technical aspects of content management

  Although you do not have specific technical requirements, a basic knowledge of the following is desirable:

  - Java
  - JSPs
  - Static and dynamic publishing
  - Databases and database management
  - Browsers

- Your collaborators

  For detailed information, see "Collaborating," on page 33 and the *Content Server Developer's Guide*.

- Online site specifications

  Online site specifications are typically prepared by site designers or the corporate communications department to define the site's audience, the site's size, its message and goals, the nature of the content, and the presentation style. Having an understanding of the online site will help you model the online site, plan sites, and collaborate efficiently with developers.

- The Content Server environment

  As the Content Server administrator, you can work with up to four different systems: development, management, delivery, and testing. Make sure you have a good understanding of the systems and your function at each of them. For detailed information, refer to the sections "Content Server Environment" and "Administrator's Job" (both in this chapter), and the *Content Server Developer's Guide*.

- Content Server's administrative interface and system defaults

  The administrative interface provides you with site configuration tools, and access to system defaults such as ACLs, roles, and asset types. For information about the administrator interface, see Chapter 2, "Administrator's Interface." For information about system defaults, see Appendix A, "System Defaults" and Appendix B, "System Data: Content Server Database."

Chapter 2

Administrator's Interface

Content Server defines three kinds of administrators, each with a different range of administrative rights. This chapter summarizes the administrators and shows you how to log in to Content Server as the general administrator. It describes the administrator's interface, its tabs, the nodes within the tabs that allow you to manage sites, and important system defaults that are accessible either from the interface or the back end.

This chapter contains the following sections:

Types of CS Administrators

Content Server defines three types of administrators:

- The general administrator, for whom this guide is written.

  The general administrator typically administers all systems in the CS environment and therefore has unrestricted access to each system’s interfaces. The jobs of the general administrator at each system are outlined in Chapter 1, “Overview.”
- Site administrators, who administer only certain sites on the management system.
- Workflow administrators, who create workflow processes on the management system

Each administrator is given permissions to relevant parts of the CS interface.

Procedures in this guide are specific to the general administrator. Whenever steps can be executed by the site or workflow administrators, this guide indicates which steps.

Finally, this guide is not restricted to any particular system in the CS environment. The concepts and procedures it contains are valid at all systems in the CS environment.

Note

As this guide is written for the general administrator, the word “you,” whenever it is used, refers strictly to the general administrator.

Logging in to the Administrator's Interface

This section shows you how to log in to Content Server as the general administrator.

> **Note**
>
> If you are *not* the general administrator and need to log in as a site administrator or workflow administrator, obtain your user name and password from the general administrator before following the steps below. Also bear in mind that because your permissions to the Content Server interfaces are limited by your role, only certain parts of this guide are relevant to you.

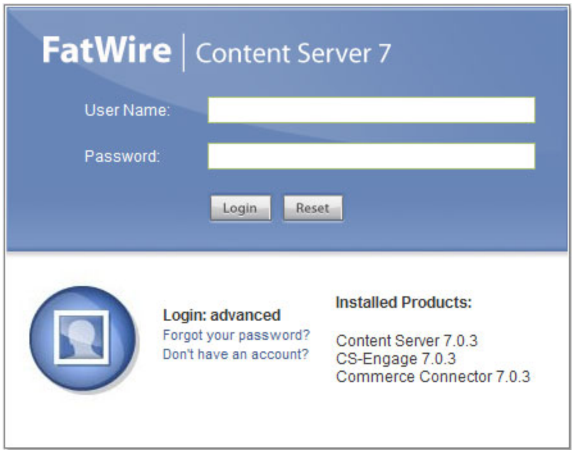
To log in to Content Server as the general administrator

1. Open your browser.
2. Set your browser so that it checks for newer versions of stored pages with every visit to the page.
3. Navigate to the following URL:

   `http://<server>:<port>/<context>/Xcelerate/LoginPage.html`

   where `<server>` is the host name or IP address of the machine running Content Server, and `<context>` is the name of the CS web application which was deployed on that machine. Depending on how the system was set up, you might also need to include the port number. The system can be development, management, delivery, testing, or another system in your CS environment.

   Content Server displays the login form, where the lower left-hand corner lists the products that are installed on the system you are logging in to.



Login: advanced
Forgot your password?
Don't have an account?

Installed Products:

Content Server 7.0.3
CS-Engage 7.0.3
Commerce Connector 7.0.3

4. Enter your user name and password.

> **Note**
>
> If you are the administrator of a new installation that had no previous administrator, log in as the default system user, with user name **fwadmin** and password **xceladmin** and then immediately change the password to a value other than “admin.” For instructions on changing your password, follow the procedure “To create a user,” on page 78.
>
> Because you have logged on as an administrator, your user name must have the xceladmin ACL (Access Control List) assigned to it. See Chapter 5, “Configuring Users, Profiles, and Attributes” for information about user accounts and their ACLs.

5. Click **Login**.
6. Content Server responds according to the number of sites that have been created in its interface. To continue, complete one of the following steps:

   - If no sites have been created, you are automatically logged in to the default management site, which enables you to create sites. (Note that the situation of no sites is rare. Most new installations run the sample sites that are provided with Content Server.) Continue with step 7.
   - If one site has been created, you are automatically logged in to that site. Continue with step 7.
   - If two or more sites have been created, they are listed in your CS interface, along with the sample sites that your installation engineer has elected to install.

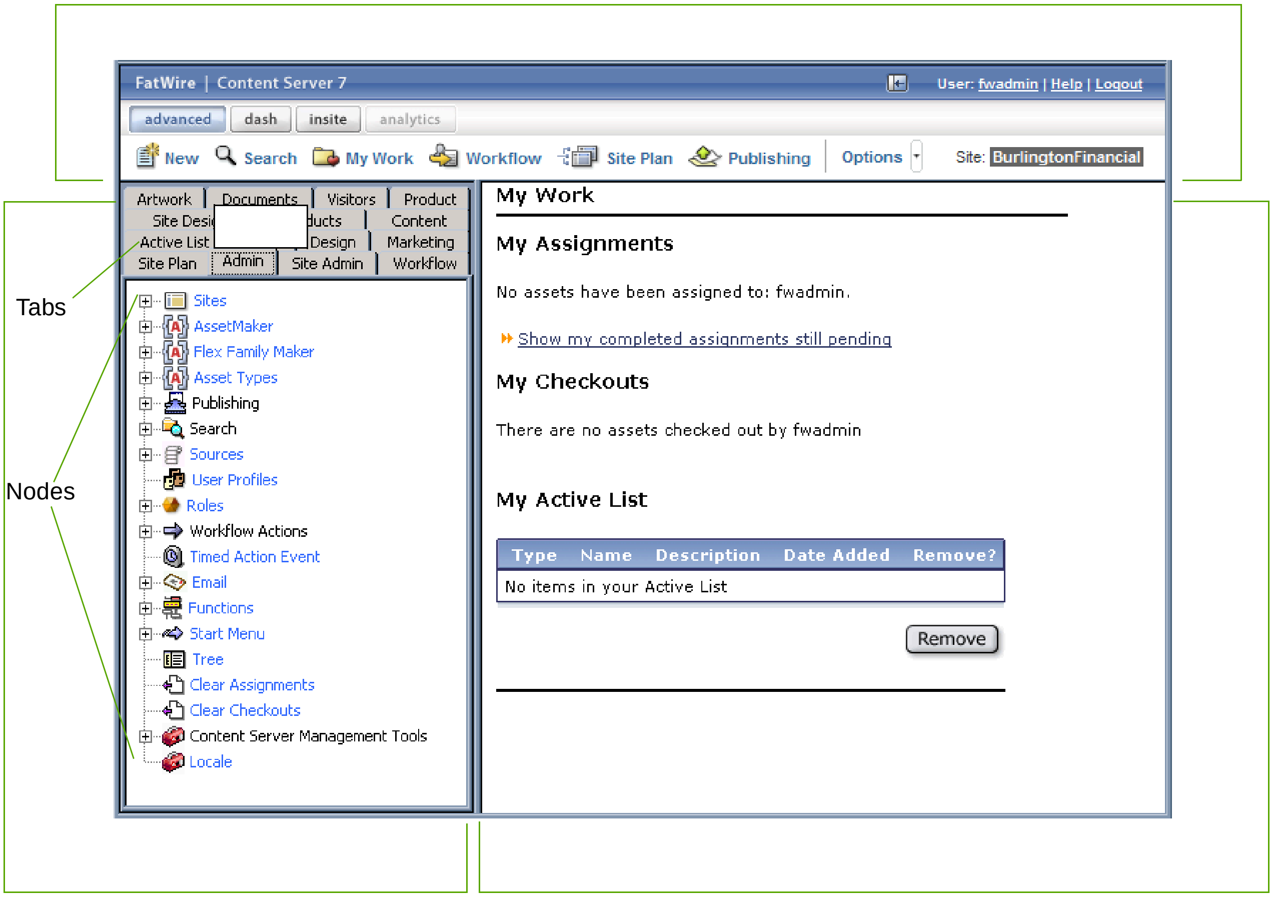
   a. Select the site that you want to work on. (For the list of sample sites and their specifications, see Appendix C, “Sample Site Configurations.”)

   b. Continue with step 7.
7. At this point, you are logged in to either the site of your choice, or the default site. All the tasks that you complete pertain to the site, until you switch to a different site. The one exception is the set of administrative tasks that you execute from the “Admin” tab, which is not site specific. The “Admin” tab contains data for all sites in the system.

   Within your selected site, Content Server displays the administrator’s interface, consisting of the three frames shown in the next figure:

Top Frame: Button bar

Displays icons and buttons for navigating Content Server.



My Work

My Assignments

No assets have been assigned to: fwadmin.

Show my completed assignments still pending

My Checkouts

There are no assets checked out by fwadmin

My Active List

Type	Name	Description	Date Added	Remove?
No items in your Active List				

Left Frame: Content Server Tree

The Content Server tree is a collection of tabs with nodes. Each node is a building block of the site.

Right Frame: Site-Building Form

This frame displays the forms that Content Server calls when you select nodes from the left frame. Each form is used to enter data into the selected node and manage the data.

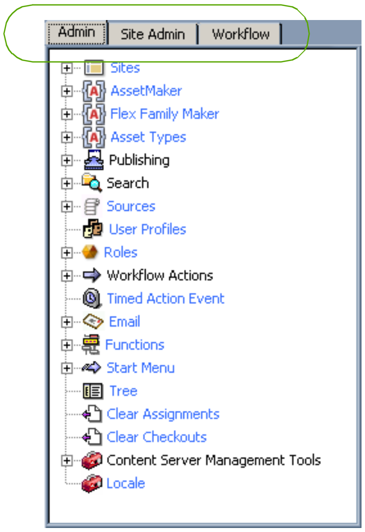
If you are a first time CS user, we recommend that you familiarize yourself with the following features:

- The administrator’s interface—the “Admin” tab in particular. You will be using the “Admin” tab almost exclusively throughout this guide to create, configure, and manage sites. For more information, see “Administrator Tabs,” on page 44.
- System defaults, such as ACLs, which you will need in order to configure users, their access rights, and so on. For more information about system defaults, see Appendix A, “System Defaults.”

Administrator Tabs

Up to three administrator tabs can be displayed in Content Server's tree:

- Admin
- Site Admin
- Workflow



The tabs are permissions-based. To be exposed to the user, they require the user to have the ACLs and roles shown below:

Tab	ACL	Role
Admin	xceladmin	GeneralAdmin
Site	xceladmin	SiteAdmin
Workflow	xceladmin	WorkflowAdmin

The tabs are also conditional on the type of site in which you are working:

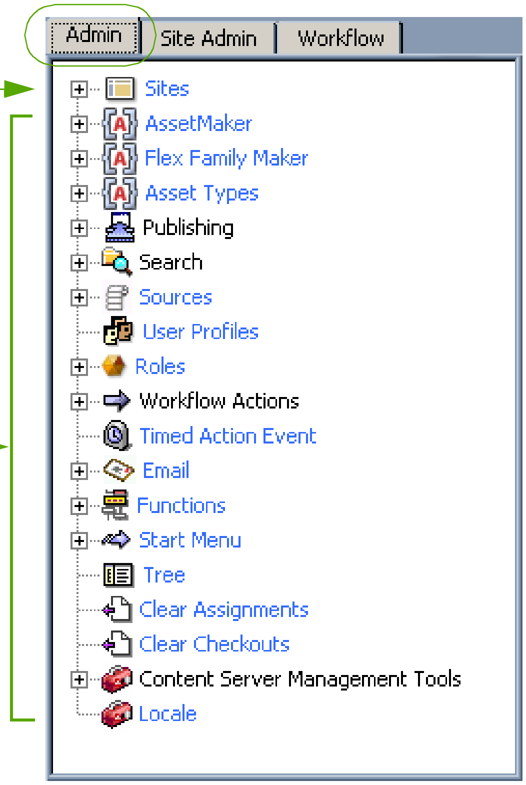
- If you log in to a Content Server installation where no sites have been created, you are automatically logged in to the default management site. Here, you see only the "Admin" tab, a system-wide tab for managing the Content Server installation.
- If you are logged in to a custom site, both the "Admin" and the "Site Admin" tabs are displayed in your interface. The "Admin" tab remains system-wide. The "SiteAdmin" tab is specific to the site you are logged in to. It is therefore a sub-set of the "Admin" tab.
- If workflow processes have been created for any site (except the default site), the "Workflow" tab is also displayed in your interface. The "Workflow" tab accounts for all workflow processes that have been configured for all sites in the system.

Whether administrator tabs are exposed to other users depends on whether you assign the users the associated ACLs and roles. For more information about ACLs and roles, see Chapter 4, "Working with ACLs and Roles."

'Admin' Tab

The "Admin" tab is the main administrative tab, used to create, configure, and manage sites. The "Admin" tab accounts for all content management components in the Content Server installation.

The "Admin" tab displays the following nodes:



- **Sites**, that is, CM sites (described in Chapter 1, "Overview").

  The **Sites** node has the following functions:

  - Lists sites that have been created in the system.
  - Enables you to define sites and associate to them the components from the system-wide configuration pool, below the "Sites" node.

    Content Server provides several sample sites as a learning tool for you. For a description of how they are configured, see Appendix C, "Sample Site Configurations."

  The configuration pool consists of the following nodes:

- **AssetMaker**, for site developers to create new asset types that use the basic asset data model. See the *Content Server Developer's Guide* for information.
- **Flex Family Maker**, for site developers to create new flex families and flex asset types. See the *Content Server Developer's Guide* for information.
- **Asset Types**, for configuring asset types:
  - To add subtypes or asset associations for basic asset types.
  - To create start menu items for asset types that appear on the **New** and **Search** lists.
  - To enable or disable revision tracking for specific asset types. See Chapter 20, "Revision Tracking" for information.
  - To enable assets for Content Server Desktop. For information, see Chapter 18, "Configuring the User Interfaces."
  - To enable assets for CS-DocLink. For information, see Chapter 18, "Configuring the User Interfaces."
- **Publishing**, for setting up the publishing system on the development and management systems. This release provides the "mirror to destination" feature that is managed from the Advanced interface. See Part 3, "Managing the Publishing System" for information.

- **Searching**, for configuring asset types so that they are indexed correctly when you are using one of the supported third-party search engine modules. See Chapter 19, "Configuring the Lucene Search Engine" for information.

> **Note**
>
> The **Searching** node is displayed only if a supported third-party search engine is installed.

- **Sources**, which you or your site developers use to add new sources if your asset types are designed to use them. See the *Content Server Developer's Guide* for information.
- **User Profiles**, which you use to specify e-mail addresses for your users. See Chapter 5, "Configuring Users, Profiles, and Attributes" for information.
- **Roles**, which you use both for determining the recipients of workflow assignments and to control users' access to functions in the Content Server interfaces.
- **Workflow Actions**, which you use to create the workflow building blocks called actions and conditions. These items can be used by any workflow process. See Chapter 9, "Creating and Managing Workflow Processes" for information.
- **Timed Action Event,** which you use to configure the frequency with which due dates are calculated for assets that are in workflow. See Chapter 9, "Creating and Managing Workflow Processes" for information.
- **Email**, which you use to create the workflow e-mail messages that can be used by any workflow process. See Chapter 9, "Creating and Managing Workflow Processes" for information.
- **Functions**, which your developers or FatWire professional services staff use to add customized functions to the Content Server interfaces.
- **Start Menu**, which you use to provide links from the form that is displayed when you click the **New** button in the Advanced interface and the links in the form that is displayed when you click the **Search** button. You use them to create new assets or search for existing ones. Start Menu items can also be created for managing access to the CS-Desktop and CS-DocLink interfaces. See Chapter 8, "Managing Access to CM Site Components" for information.

> **Note**
>
> This release of Content Server supports automatic creation of the **New** and **Search** Start Menu items.

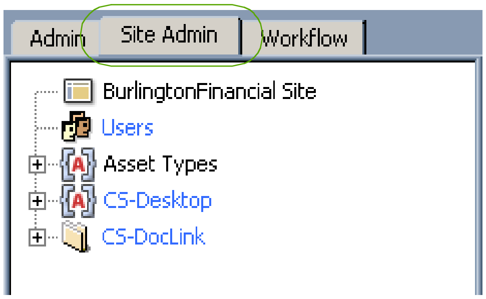
- **Tree**, which you use to control access to the existing tree tabs or to create new ones. See Chapter 8, "Managing Access to CM Site Components" for information.
- **Clear Assignments**, which you use to clear workflow assignments. See Chapter 9, "Creating and Managing Workflow Processes" for information.
- **Clear Checkouts**, which you use to unlock and check back in to the database the assets that are checked out by specific users. See Chapter 20, "Revision Tracking" for information.

- **Content Server Management Tools**, which you use the create ACLs and users, and to enable or disable revision tracking for non-asset tables. For information, see Chapter 4, "Working with ACLs and Roles," Chapter 5, "Configuring Users, Profiles, and Attributes," and Chapter 20, "Revision Tracking."
- **Locale**, which you use to set the default locale for the CS system. See Chapter 18, "Configuring the User Interfaces" for information.

'Site Admin' Tab

The "Site Admin" tab is site-specific. It is used to manage the site that you are logged in to.

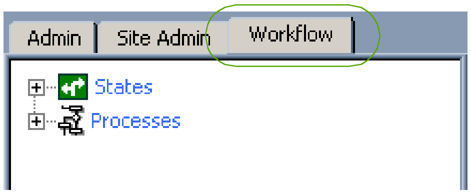
The "Site Admin" tab holds a subset of the administrative functions in the "Admin" tab. It allows you to manage existing users of the site; enable or disable asset types; enable or disable Content Server Desktop and CS-DocLink (if it is installed). It does not allow you to create users, asset types, or sites.



For any other user to access this tab, the user must be assigned the xceladmin ACL and the **SiteAdmin** role for the site(s) that the user logs in to.

'Workflow' Tab

The "Workflow" tab is used to configure workflow processes from the workflow building blocks that were created in the "Admin" tab.



Note

While the “Workflow” tab itself is site-specific, its content is not.

The tab. When a user logs on to a site, the user can access the “Workflow” tab if he is assigned the xceladmin ACL and WorkflowAdmin role *for that site*. In this respect, the “Workflow” tab is site-specific.

The tab’s content. The “Workflow” tab displays system-wide information, that is, all workflow processes for all sites in the installation. Therefore, the content of the “Workflow” tab is not site-specific.

Non-Administrative Tabs

Your interface displays default non-administrative tabs, and custom tabs if they have been configured by a previous administrator. You can create additional tabs as necessary.

For information about the default tabs, see “Default Tree Tabs,” on page 364. For information about creating and configuring tabs, see “Managing Access to the Tree (Advanced interface only),” on page 120.

System Defaults

Content Server has a number of system defaults:

- Non-administrative tabs, described in the previous section
- ACLs
- Roles
- Users
- Asset types
- System Tables

  You must be familiar with the defaults in order to configure users and sites. Information about system defaults is given in Appendix A, “System Defaults” and Appendix B, “System Data: Content Server Database.”

Sample Sites

To help you better understand Content Server, we have packaged it with sample sites. You can use the sites to learn about and experiment with Content Server without affecting your own sites. The sample sites vary in complexity and purpose. In addition to providing code samples for developers, they exemplify how sites are configured.

If your installation engineer has elected to install the sample sites, they will be displayed in your interface.

- To identify the sample sites and obtain their specifications, see Appendix C, “Sample Site Configurations.”

- To work on the sample sites, follow the steps in “Logging in to the Administrator’s Interface,” on page 41. You can access the corresponding online sites by pointing your browser to each site’s URL, available from your installation engineer.

Note

As mentioned earlier, Content Server now supports the portal environment and provides its business users with two set of portlets: one for managing structured content, the other for managing document-based content. Content Server also provides the “Spark” sample site, which can be optionally installed to demonstrate the portal environment (described in the *Content Server Portal Interface User’s Guide*). Note that the CS administrator continues to use the standard interface to manage the portal environment and its users.

Also note that while some administrative portlets are packaged with Content Server, they are specific to the Spark sample site and are there to help CS administrators manage the Spark sample site, if it is installed. Readers who are interested in learning more about the Spark product can contact their sales representatives for information.

Chapter 3

Site Configuration Guidelines

Site configuration in Content Server is based on a relational system, where components in a system-wide configuration pool must be associated with each other and with a site name in order to function as a site. Making the correct associations creates a functional site—one that hosts the correct users, provides them with the correct asset types for authoring and managing content, supports workflow as necessary, publishes correctly, and delivers approved content to the intended audiences.

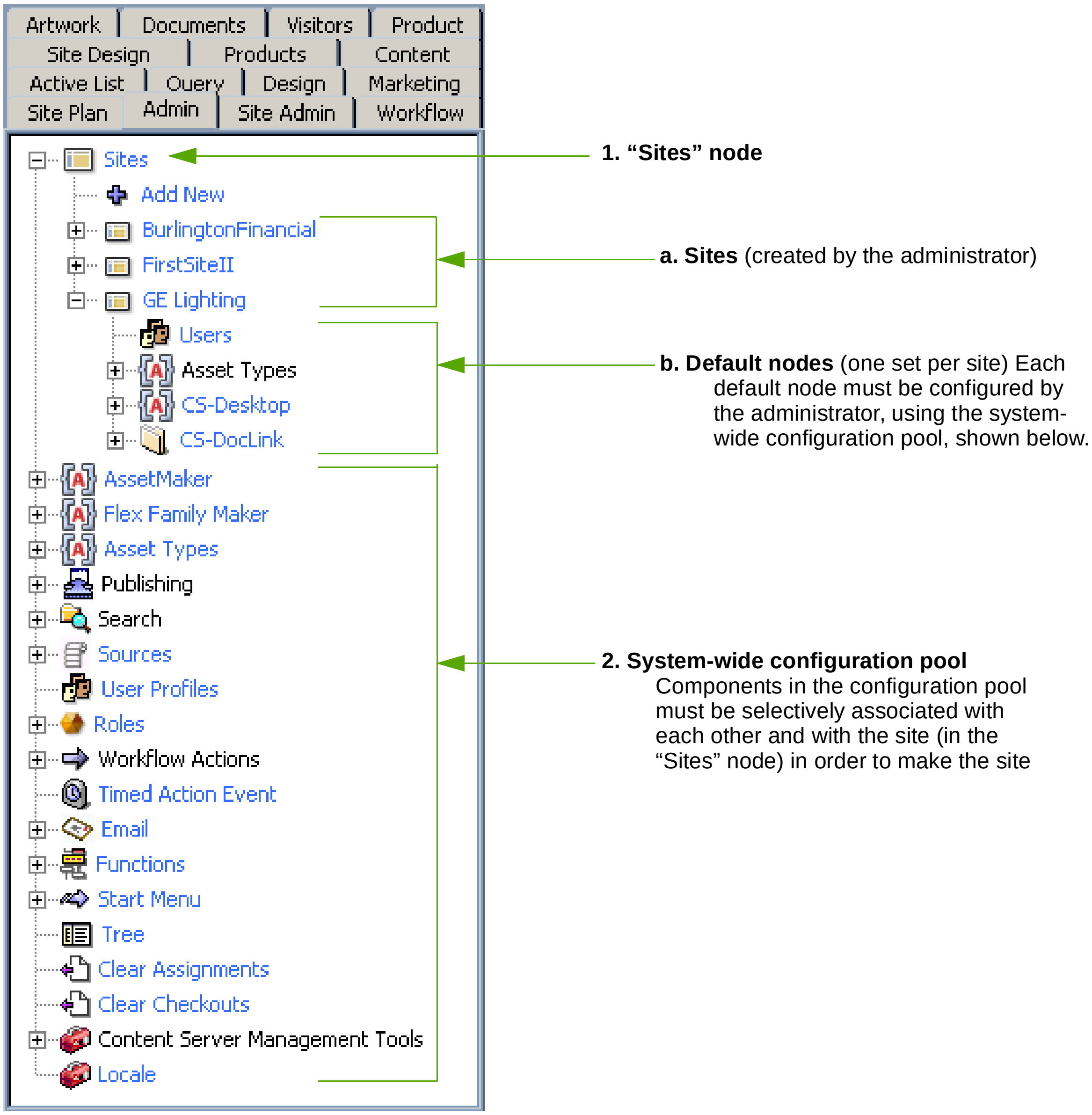
This chapter contains procedures that show first-time administrators how to configure sites. It also contains recommendations for the more experienced administrators to help them manage and maintain established sites.

This chapter contains the following sections:

Overview

When creating and configuring sites, you will be using the “Admin” tab, shown in the figure below. The tab is logically separated into two sections: (1) the “Sites” node, where sites are first established, and (2) the system-wide configuration pool, where site components are created, associated to each other, and managed. The configuration pool begins with the “AssetMaker” node and ends with “Content Server Management Tools.”

Initially, when a new site is defined and added to the “Sites” node, it is empty and non-functional, because it is not associated in any way with the configuration pool. You make the site functional by associating with it the correct components in the configuration pool. The components are sharable among sites. Any component that you associate with one site, you can associate with as many other sites as necessary. More information about the “Sites” node and the configuration pool is given on the next page.



1. “Sites” node

a. Sites (created by the administrator)

b. Default nodes (one set per site) Each default node must be configured by the administrator, using the system-wide configuration pool, shown below.

2. System-wide configuration pool
Components in the configuration pool must be selectively associated with each other and with the site (in the “Sites” node) in order to make the site

Sites are configured in four *basic* steps from the “Admin” tab, as shown below. We emphasize *basic* to indicate that the steps are intended to provide only an overview of the site configuration process. Detailed procedures are given in the rest of this guide.

Note

In the procedure below, steps 1 and 2 are interchangeable.

Step	Description
1. Add sites to Content Server: **a.** In the **Admin** tab, expand **Sites** and double-click **Add New**. **b.** Enter the site name and description. **c.** Select a preview method by which you will preview the site’s content (when the content is created).	The “Sites” node (rather than the configuration pool) can be the starting point for creating sites. When you execute step 1, Content Server adds the site to the “Sites” node. Under the newly added site, Content Server creates a set of sub-nodes that prompt you for site components: • The “Users” sub-node prompts you for users. • The “Asset Types” sub-node prompts you for asset types (created by developers). • The “CS-Desktop” sub-node prompts you to enable CS-Desktop. • The “CS DocLink” sub-node prompts you to enable CS-DocLink. Each sub-node is equipped with a linking mechanism that allows you to associate the sub-node, and therefore the site, with data in the configuration pool. Because of this linking mechanism, the sub-nodes cannot be deleted or supplemented with custom sub-nodes. All data required by the sub-nodes must be entered in to the configuration pool, described next.
2. Populate the configuration pool with components, such as users, asset types, and roles.	The configuration pool begins with the **AssetMaker** node and ends with **Content Server Management Tools**, as shown in the figure on page 52. The configuration pool enables you to select and create components for the site and to associate the components with each other. Data that you establish in the configuration pool remains independent of any sites (and therefore inaccessible to users) until you associate the data with the sites.

Step	Description
3. Associate and enable components in the configuration pool with the site: **a.** In the **Admin** tab, expand **Sites** and double-click the desired site. **b.** Add users and asset types to the site. **c.** Enable the users and asset types by means of start menu items, access permissions, and permissions to tree tabs.	The “Sites” node is the endpoint of the site building process. In this step, you once again use the “Sites” node, this time to associate components in the configuration pool to the site that you added in step 1. Note that components in the configuration pool are reusable; they can be associated with as many sites as necessary.

Site Replication Option

If you plan to create many sites that are similar in content and structure, you can shorten the creation process. You can configure the appropriate source sites, then replicate the sites as often as necessary, and finally modify the replicates as necessary. The source sites must conform to certain replication requirements. For more information, see Chapter 10, "Replicating CM Sites."

Site Configuration Steps

In the site configuration process, certain components in the configuration pool are required by other components and must be created first if re-tracing of steps is to be minimized. For example, before users can be created, their ACLs must first exist.

In general, the order of steps in the site configuration process is flexible. One possible order is given in the procedure below, where the critical site components are created first, then assembled into a site, and finally enabled; the optional components (with the exception of workflows) are considered last, after the publishing process is configured.

If you are a first-time administrator, we suggest that you familiarize yourself with the procedure below in order to obtain a basic understanding of the configuration process and gather all the information that you will need. Once you understand how sites are configured, you can modify the procedure to suit your needs.

We also suggest that you make a copy of the procedure below and keep it handy when configuring sites. Each step refers you to sections of this guide (or the *Content Server Developer's Guide*) where you can find detailed information about the particular step.

Assumptions

We assume that you and developers will code the data model at the development system, and after successful testing, migrate the data model to the management system. At the management system, you will complete the site configuration tasks, test the management system, and finally deploy the system. We also assume that before beginning the configuration tasks, you have read Chapter 1, "Overview" to obtain a general understanding of Content Server concepts, components, and content management models.

Pre-Configuration Decisions

Step 1. Determine the user management method

Your options are Content Server's native user manager, an LDAP plug-in, or an external manager.

While this guide is based on the native Content Server user manager, it does describe system behavior for LDAP-integrated systems. For information, see Appendix D, "Managing Users, Sites, and Roles in LDAP-Integrated CS Systems." For information about configuring users in LDAP, refer to the LDAP product documentation.

Step 2. Determine users' table-level permissions (ACLs)

Regardless of which user management method you will be implementing, you must have ACLs defined in Content Server. Examine the list of system ACLs (and custom ACLs, if any have been created) to determine which ACLs to assign to users, and whether you need to create additional ACLs. Typically, the system ACLs are sufficient. For general information about ACLs, see Chapter 4, "Working with ACLs and Roles." For specifications, see Appendix A, "System Defaults."

Step 3. Determine the users' roles

Content Server defines several system roles and additional roles for the sample sites. You can reuse the roles for your sites, or create your own. Typically, you will need to create roles to cover the full range of users' responsibilities on any given site.

For information about roles, see the following:

- Chapter 1, "Overview" and the section "Roles," on page 71 for general information.
- Appendix A, "System Defaults," for the list of system roles and their specifications.
- Appendix C, "Sample Site Configurations" for a list of sample roles, their descriptions, and their usage on the sample sites.

Step 4. Determine whether you will be replicating the sites you are configuring

If you plan to replicate the sites you are configuring, make sure that template-coding and other requirements are satisfied for the source sites (the sites you are currently configuring). For an overview of site replication as well as replication options and requirements, see Chapter 10, "Replicating CM Sites."

Step 5. Determine system security

Before developers begin designing the online site, or contemplating changes to the user interface on the management system, you must determine and implement your security protocols. Decisions that you make about security affect the way that developers code and implement the online site. For information about security, see Chapter 6, "Setting Up External Security."

Configuration Steps at the Development System

Configuration steps at the development system are also given in detail in Chapter 2 "CS Development Process," in the *Content Server Developer's Guide*. When coding at the development system is complete, the data model is migrated to the management system, where site configuration is completed.

Step 1. Create the sites that will be used on the management system

In this step, you will create sites with the names that will be used on the management system. For information and procedures on creating sites, see Chapter 7, "Assembling and Organizing CM Sites" (in particular "Creating a Site," on page 94).

Step 2. Create the data model (asset types and their definitions)

1. Create content asset types.
2. Create design asset types (templates), bearing in mind site-replication requirements, if any.
3. Create management asset types.

4. Create sample assets of each type.

For information about creating asset types, see the *Content Server Developer's Guide*.

Step 3. Test the data model

Step 4. Migrate the data model to the management system

To migrate the data model, mirror publish it to the management system. For instructions, see Chapter 15, "The Dynamic Publishing Process."

Configuration Steps at the Management System

Step 1. Create ACLs, if necessary

For procedures on creating ACLs, see Chapter 4, "Working with ACLs and Roles" (in particular "Creating a New ACL," on page 67).

Notes

- Under no circumstances should you modify or delete a system ACL.
- **Database tables.** When the data model is migrated to the management system, assign ACLs to the custom database tables (asset types), as necessary. For procedures on assigning ACLs to database tables, see Chapter 4, "Working with ACLs and Roles" (in particular "Assigning ACLs to Custom Tables," on page 69).
- **Content Server page entries.** Typically there is no need to assign ACLs to Content Server page entries on the management system because you use roles to control access to assets and interface functions. However, if you need instructions for assigning ACLs to Content Server page entries, see "Assigning ACLs to Content Server Pages (SiteCatalog Page Entries)," on page 70.

Step 2. Configure the external user manager, if you are using one

For procedures on configuring an LDAP server, see our guide, *Configuring Third-Party Software*.

Step 3. Create roles

For procedures on creating roles, see Chapter 4, "Working with ACLs and Roles" (in particular "Creating a Role," on page 72).

Step 4. Create the users (the content providers)

For instructions on creating user accounts in Content Server, see "Creating a New User," on page 78. If you are creating user accounts in external managers, refer to the product documentation.

Step 5. Create user profiles, if necessary

User profiles are required for users who will be working with:

- CS content applications (Engage, Analytics)
- Language packs and setting a default language
- Workflow processes, in which e-mail messages will be sent to notify workflow participants of their assignments. The user profile supports workflow actions by mapping a user name to an e-mail address.

User profiles must be created in Content Server. For instructions, see "Creating and Editing a User Profile," on page 82.

Step 6. Set user attributes, if necessary

If users require attributes beyond the locale and e-mail attributes that are specified in the user profile, you can create the attributes. For instructions, see "Modifying, Adding, and Deleting User Attributes," on page 83.

Step 7. Create workflow processes for asset types, as necessary

Implementing workflow processes is a business decision, and not a Content Server requirement. If you need to create workflow processes, do the following:

1. Create the "Workflow" tab for the site, if one does not already exist. For instructions, see "Creating Tree Tabs," on page 121.
2. Plan your workflow process by sketching it. See Chapter 9, "Creating and Managing Workflow Processes" for help with this step. Then refer to your sketch and notes throughout this section.
3. If you have not already done so, create the roles that are needed for the workflow processes. in you need instructions, see "Creating a Role," on page 72.
4. Ensure that users are set up to participate effectively in the workflow:

   a. Ensure that users who will participate in workflow have user profiles created for them. Otherwise, they will not receive e-mail messages from the workflow process.

   a. For shared assets, if you want the pool of workflow candidates to include people from all the sites that an asset is shared to, enable the cross-site assignments feature. See "Cross-Site Assignments and Participants," on page 138 for details.
5. Create the e-mail objects that you need for your actions and enable the `xcelerate.emailnotification` property in the `futuretense_xcel.ini` file. For instructions, see Chapter 1, "Overview."
6. Create the step actions, timed actions, deadlock actions, group deadlock actions, and delegate actions that you need. For instructions, see "Setting Up the Workflow Actions and Conditions," on page 161.
7. If your states have deadlines, be sure to configure the Timed Action Event so that the deadlines of assets are calculated regularly and the appropriate timed actions (if any) are invoked in a timely way. For instructions, see "Setting Up the Timed Action Event," on page 164.
8. Create your states. For instructions, see "Setting Up the States," on page 165.

9. Create your process. While creating your workflow process, you create the steps for that process. The steps link together the states so they occur in the proper order. Additionally, while creating your process, you configure any function privileges that you need. For instructions, see "Setting Up the Workflow Processes," on page 166.
10. Test your workflow processes. For instructions, see "Testing Your Workflow Process," on page 175.

Set up your start menu shortcuts so that workflow processes are assigned automatically to assets when they are created. For help with start menu items, see "Creating Start Menu Items," on page 108.

Step 8. Re-create the sites that were created on the development system

Again, for information and procedures on creating sites, see "Assembling and Organizing CM Sites," on page 93 (in particular "Creating a Site," on page 94).

Step 9. Migrate the data model, if you have not already done so

At this point, you will need the data model in order to complete most of the remaining steps. Mirror publish the data model from the development system to the management system by using procedures in Chapter 15, "The Dynamic Publishing Process."

Step 10. Assign custom ACLs (if any) to the custom tables, if you have not already done so

For procedures on assigning ACLs to database tables, see Chapter 4, "Working with ACLs and Roles" (in particular "Assigning ACLs to Custom Tables," on page 69).

Step 11. Assemble the sites

1. Grant users access to the sites. For instructions, see "Granting Users Access to a Site (Assigning Roles to Users)," on page 98.
2. Enable asset types for the sites by associating the asset types with the sites. For instructions, see "Enabling Asset Types for a Site," on page 101.

> **Note**
>
> This step provides you with the option of allowing Content Server to automatically create "Start Menu" items for the asset types you are enabling. The "Start Menu" items **New** and **Search** items allow site users to create and search for assets of those types. You can also create your own "Start Menu" items, as summarized in the next step.

Step 12. Make site components available to users

1. If you have chosen to create your own "Start Menu" items for various asset types, go to "Managing Access to Asset Types," on page 108 for instructions.
2. Set access permissions on assets. For instructions, see "Setting Access Permissions for Assets," on page 114.
3. Give users access to the tabs in Content Server's tree. For instructions, see "Managing Access to the Tree (Advanced interface only)," on page 120.

Step 13. Configure the CS publishing process

1. Determine which type of publishing your management system is to execute:
    - Static (Export to Disk)
    - Dynamic (Mirror to Server)
    - Data transformation (**Export Assets to XML**)

2. Configure the publishing process so the information that content providers create can be made available to the delivery system.

For general information and instructions on configuring the publishing process, see Part 3, "Managing the Publishing System."

Step 14. Configure user interfaces, as necessary

1. The user interface options are CS-Desktop, CS-DocLink, InSite interface, and the portal interface. For instructions on making these interfaces available to content providers, see Chapter 18, "Configuring the User Interfaces."
2. If you have more than one language pack installed on your system, configure the default locale for the system. For instructions, see Chapter 18, "Configuring the User Interfaces."

Step 15. Set up revision tracking, as necessary

Revision tracking prevents assets from being edited by more than one user at a time. When you enable revision tracking for a database table, Content Server maintains multiple versions of the assets in that table. For instructions on setting up revision tracking, see Chapter 20, "Revision Tracking."

Step 16. Enable asset types for search engines, as necessary

If you purchased a supported third-party search engine modules and you are using basic asset types, configure the asset types to be searched, so that they can be indexed correctly and found by your search engine. For instructions on configuring the asset types, see Chapter 19, "Configuring the Lucene Search Engine."

Step 17. Set up the delivery system

For more information, see Chapter 2 "CS Development Process," in the *Content Server Developer's Guide*.

Step 18. Test all systems

Step 19. Give users the URL to the CS installation and their login information

Part 2
CM Site Configuration Procedures

This part contains detailed procedures for configuring and replicating sites on your Content Server system.

This part contains the following chapters:

Chapter 4

Working with ACLs and Roles

Several user management components are used to control access to a Content Server system: ACLs, user accounts, user profiles, roles, and sites. Site configuration, itself, begins with ACLs, with the creation of user accounts, and with the creation of roles.

ACLs can be used on both your management and your delivery systems. However, the main focus of this chapter is user management on the management system. For information about user management on your delivery system, see the “Site Development” section in the *Content Server Developer’s Guide*.

This chapter contains the following sections:

- Overview
- ACLs
- Working with ACLs
- Roles
- Working with Roles

Overview

ACLs and roles are of paramount importance in Content Server.

- ACLs are used to regulate entry in to the Content Server system. The assignment of ACLs to users, database tables, and Content Server Pages determines users' permissions to operate on Content Server's database tables. If the user's ACLs match the database ACLs, the user has certain permissions (defined by the ACLs) to operate on the database tables. ACLs therefore serve as the foundation of the security and user management model. Without ACLs user accounts cannot be created.
- Roles are used to manage access to sites and their components. The assignment of roles to users and interface functions on a given site determines whether the interface functions are enabled for the users or hidden from them. If the user's roles match the roles that are assigned to the interface functions, the functions are enabled for the user. Otherwise, the functions are hidden.

Before you can establish users you must determine which ACLs and roles the users need to be assigned. If the ACLs and roles do not already exist, you must create them in Content Server, even if you intend to use LDAP plug-ins to create the user accounts.

This section provides you with a basic explanation of ACLs and roles, and shows you how to create, modify, and delete ACLs and roles.

ACLs

Access Control Lists, called **ACLs** for short, are named sets of database operation permissions such as read, write, create, and retrieve. Because just about everything in Content Server (and the Content Server content applications) is represented as one or more rows in one or more database tables, user management on any of your CS systems starts with ACLs.

With ACLs, you can limit access to the following items:

- Individual database tables
- Individual Content Server pages

Content Server and the CS content applications use ACLs to enforce access restrictions to the various functions of those applications by controlling the user's access rights to the database tables that represent those functions. How? By verifying that the users attempting the function have the same ACL assigned to their **user accounts** as are assigned to the database table.

For example, user account information is contained in the system tables named `SystemUsers` and `SystemUserAttrs`. Because certain ACLs are assigned to those system tables, only a user with the same ACLs assigned to his or her user account is able to create new users or edit existing user information.

ACLs serve as the foundation of the security and user management model in your CS system by providing authorization functionality. Even if you are using an external user manager like LDAP to store user information, you must use Content Server ACLs.

A user must always have at least the Browser ACL in order to view CS pages. However, additional ACL restrictions are enforced only when the `cc.security` property in the `futuretense.ini` file is set to `true.` For information about the `cc.security` property, see futuretense.ini in the *Property Files Reference*. For information about configuring security on your CS system, see Chapter 6, "Setting Up External Security."

ACLs are assigned to three things: user accounts, database tables, and page entries in the SiteCatalog table (that is, Content Server pages).

> **Note**
>
> **User Management on the Delivery System.** User management on your delivery system is also based on ACLs. If your online site is designed to require visitors to register or log in before they can access areas of the site, you create the ACLs that are needed on the delivery system and then assign them to the appropriate database tables.
>
> Typically your site designers take care of assigning ACLs to Content Server pages. The *Content Server Developer's Guide* discusses how to design a user management process on the delivery system and provides code samples for pages that log visitors in to the site and verify their identities.

User Accounts

Every user must be assigned at least one ACL and the ACLs assigned to a user define that user's access to the CS system.

While users have one user account and one set of ACLs, no matter how many sites they have access to, they can have one set of roles for one site and a different set of roles for another site. Therefore, users must be assigned all the ACLs necessary to provide them the permissions that they need to fulfill all of their site-specific roles.

For example, if you create a role that allows a user with the role the ability to create template assets, the user assigned that role must also be assigned the ElementEditor ACL, because creating templates writes data to the ElementCatalog table.

Database Tables

To restrict access to the data in a table, assign an ACL to it through the **Content Server Database** forms available through the **Admin** tab. Then, only those users with the same ACL have access to the data in that table.

If you assign more than one ACL to a table, a user needs only one of those ACLs to access the table. The user's access rights to that table (read, write, create, and so on) are the ones defined by the ACL.

All of the Content Server system tables (and several of the CS content applications tables) have ACL restrictions. The SystemInfo table lists all the tables in the Content Server database and the ACLs assigned to them.

> **Note**
>
> Do not add ACLs to database tables through the Content Server Explorer application. Instead, use the **Content Server Database** form in the **Admin** tab.

With one exception—to register a foreign table in the Content Server database—never attempt to change the information in the SystemInfo table even if your user account has the ACL that allows you to do so.

For information about:

- Assigning ACLs to database tables, see "Assigning ACLs to Custom Tables," on page 69
- Registering foreign tables, see the "Database" chapter in the *Content Server Developer's Guide*

Page Entries in the SiteCatalog Table

The `SiteCatalog` table holds page entries for all the pages displayed for the CS content applications as well as the pages displayed for your online site (that is, the site that you are delivering from the delivery system.) If you want to restrict access to a page, assign an ACL to it.

Typically, site developers determine how page restrictions should be configured on the delivery system. If you are customizing the user interface on the management system, however, you might need to use ACLs to restrict access to your custom pages.

System ACLs

Content Server and the CS content applications use a number of system ACLs to control user access to their features and functions. Different combinations of these ACLs must be assigned to users. Information about system ACLs and their permissions are given in Appendix A, "System Defaults."

Sample ACLs

A newly installed Content Server system contains only system ACLs. No additional ACLs have been created for any of the sample sites that are packaged with Content Server.

Custom ACLs

Because Content Server provides a comprehensive of ACLs, it is unlikely that you will need to create your own. However, situations can arise where user management needs on the management or delivery systems require you to create ACLs. For example,

- If your online site requires user registration, you may need to create a set of ACLs for site visitors.
- If your developers create new functions and put them on new tabs to customize the management system, you might need to create additional ACLs (or roles) to support the new functions and tabs.

Although creating and applying ACLs is an administrative task, you must first work with your site designers and developers to determine which ACLs you need and how to apply them. Once you have determined the ACLs, create them by following procedures in the rest of this chapter.

Working with ACLs

This section shows you how to create ACLs, edit, and delete custom ACLs; apply ACLs to the database tables and Content Server pages; and customize “access restricted” messages.

Note

When using an LDAP integration option, be aware of system response to user and site management operations. For information about system response, see Appendix D, “Managing Users, Sites, and Roles in LDAP-Integrated CS Systems.”

Creating a New ACL

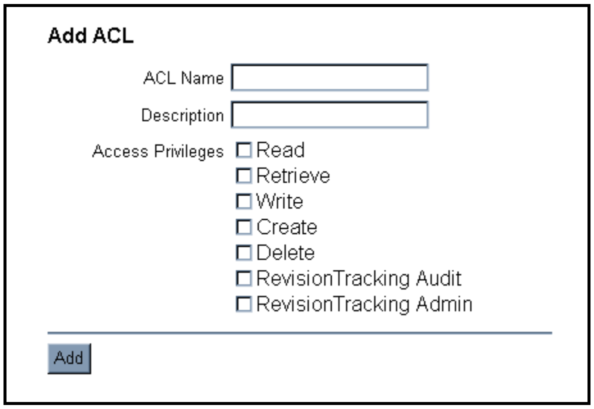
Note

When creating ACLs, consider the roles you will be using in order to ensure that the ACLs are commensurate with the roles. For example, if you will be creating a role that allows a user to create template assets, the user who is assigned that role must also be assigned the ElementEditor ACL, because creating templates writes data to the `ElementCatalog` table.

To create a new ACL

1. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **ACLs**.
2. In the form that appears, select **Add ACL** and click **OK**.

   The “Add ACL” form appears.



Add ACL

ACL Name

Description

Access Privileges
□ Read
□ Retrieve
□ Write
□ Create
□ Delete
□ RevisionTracking Audit
□ RevisionTracking Admin

3. In the “ACL Name” field, enter a unique name.
4. Select the access privileges you want to assign to this ACL. For information on each privilege, see “Permissions,” on page 354).
5. Click **Add**.

   Content Server creates the ACL and writes it to the `SystemACL` table. The new ACL appears in the drop-down list in the form described in step 2 of this procedure.

6. If you are using LDAP, create a group (on your LDAP server) that exactly matches the ACL you just created. After you create the group, assign it to the appropriate users.

Editing a Custom ACL

> **Caution**
>
> Never modify any of the system ACLs. For a list of these ACLs, see "System ACLs," on page 356.

To edit a custom ACL

1. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **ACLs**.
2. In the form that appears, do the following:
   a. In the drop-down list, select the ACL you want to edit.
   b. Select **Modify ACL** and click **OK**.
3. In the "Modify ACL" form, make the desired changes. For information on the displayed options, see "Permissions," on page 354.
4. Click **Modify**.

   Content Server writes your changes to the `SystemACL` table.

Deleting a Custom ACL

> **Caution**
>
> Never delete a system ACL. For a list of these ACLs, see "System ACLs," on page 356.

To delete a custom ACL

1. If you are using LDAP, delete (from your LDAP server) the group corresponding to the ACL you will be deleting.
1. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **ACLs**.
2. In the form that appears, select the desired ACL from the drop-down list and click **OK**.

   Content Server displays a warning message.
3. Click **OK**.

   The ACL has been deleted.

Assigning ACLs to Custom Tables

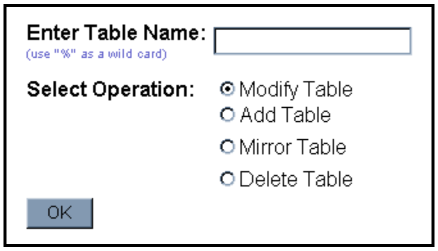
If you or the site designers create new tables, you might need to restrict access to those tables by assigning ACLs to them. Typically, you assign ACLs to new tables when you create those tables. (For more information, see the *Content Server Developer's Guide*.)

> **Note**
>
> Do not assign additional ACLs (beyond the ones assigned by default) to system or core product tables.

To assign ACLs to an existing table

1. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **Content Server Database**.

   Content Server displays the following form.



Enter Table Name:
(use "%" as a wild card)

Select Operation:
● Modify Table
○ Add Table
○ Mirror Table
○ Delete Table

2. Enter the name of the table to which you want to assign ACLs. If you do not know the name of the table you want to work with, do one of the following:
   - Leave the field blank. Content Server will return a list of all tables in the database.
   - Enter a partial name, ending with the wildcard character (**%**). Content Server will return a list of tables named similarly to your criteria.
3. Select **Modify Table** and click **OK**.
4. In the list of tables, select the desired table.

   Content Server displays the "Modify Catalog" form.
5. In the **ACL** field, select the ACL(s) you want to assign to the selected table. To select multiple ACLs, **Ctrl-click** each desired ACL. You can also select a range of ACLs by **Shift-clicking** the first and last ACL in the range.

   > **Note**
   >
   > Do **not** change the value of the **File Storage Directory** field. For information about this field, look up the `defdir` property in the *Content Server Developer's Guide*.
6. Click **Modify**.

Assigning ACLs to Content Server Pages (SiteCatalog Page Entries)

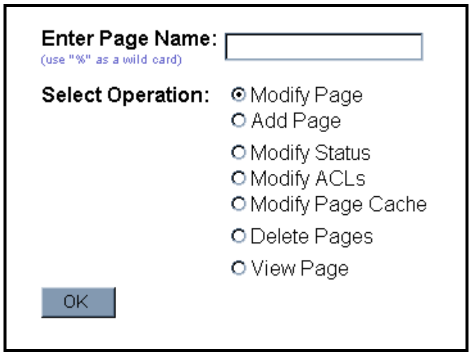
There are at least two ways to assign ACLs to `SiteCatalog` page entries.

- When developers create “SiteEntry” or template assets, they can assign ACLs to the page entry created for that asset through a field in the “Create” or “Edit” form.
- For page entries that are not associated with a “SiteEntry or template asset, developers can use the Content Server Management Tools.

To assign ACLs to a page entry that is not associated with a SiteEntry or template asset

1. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **Site**.

   Content Server displays the following form:



Enter Page Name: ▭
(use "%" as a wild card)

Select Operation:
● Modify Page
○ Add Page
○ Modify Status
○ Modify ACLs
○ Modify Page Cache
○ Delete Pages
○ View Page

2. Enter the full path and name of the page to which you want assign ACLs. If you do not know the name of the page you want to work with, do one of the following:
   - Leave the field blank. Content Server will return a list of all page entries in the `SiteCatalog` table.
   - Enter a partial name, ending with the wildcard character (**%**). Content Server will return a list of pages named similarly to your criteria.
3. Select **Modify ACLs** and click **OK**.
4. In the scrolling list at the top of the form, select the ACLs you want to assign to one or more pages. To select multiple ACLs, **Ctrl-click** each desired ACL. You can also select a range of ACLs by **Shift-clicking** the first and last ACLs in the range.
5. In the list of pages, select the **Apply** check box next to each page to which you want to assign the ACLs you selected in step 4. ACLs currently assigned to each page are shown in the **ACL** column. (To select all of the pages in the list, click **All**. To deselect all of the pages in the list, click **None**.)
6. Click **Apply**.

Setting the ACL Restriction Error Message

When users attempt to access a page for which they do not have the appropriate permissions, Content Server displays an error message. This message is stored in the following file:

```
<cs_install_dir>/futuretense_cs/formpriv.html
```

You can customize this and other error message pages in the `futuretense_cs` directory, with the following restrictions:

- Do not rename any of the files.
- Do not alter any occurrences of the `{0}` string in any of the files. Content Server uses the `{0}`string to automatically generate these error messages.

Roles

User management includes the concept of roles. Roles complement users in the following ways:

- Whereas the user definition (account) describes an individual's access to the underlying CS functionality (database tables) through ACLs, roles are used to manage site-specific access to Content Server's interface functions.
- A role represents a job description or the title of individuals with similar functions: for example, content provider, editor, site designer, and administrator.
- Each CS user has one user definition (account) no matter how many sites have been created. However, the user's roles can vary by site.
- When you enable a user for a site, you enable that user within the context of the roles that user is to fulfill for that site.

The roles assigned to a user for a site determine the following:

- Which assets the user can create on that site.
- Which assets the user can search for on that site.
- Which tabs are displayed in the tree when the user logs in to the site.
- Whether the user is eligible for participation in any workflow processes, and, if so, for which steps in those workflow processes.
- Which functions a user can or cannot perform on an asset while it is moving through a workflow process.
- Whether the user can administer a workflow process or create or modify a workflow group on that site.

System Roles

Several system roles are installed by Content Server. One role is required for the CS content applications to function, and three are required for the Content Server administrators to function. For more information, see "System Roles," on page 361.

Sample Roles

If you installed one or more sample sites, you will have access to a number of sample roles that are included with the sites. The roles permit the sample site users to access different tree tabs. You can use the sample roles as examples of how you can configure access control on your sites. For descriptions of sample roles, see “Sample Roles,” on page 377.

Custom Roles

Unlike ACLs, roles are objects that you will most likely need to create in order to account for the full range of users’ responsibilities on your sites. To create roles, follow instructions in the next section, “Working with Roles.”

Working with Roles

This section shows you how to create, edit, and delete roles.

Note

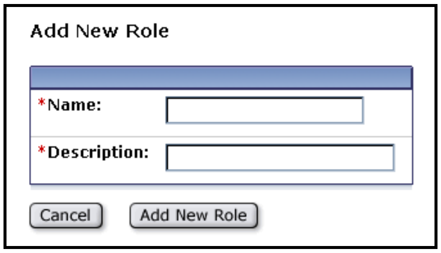
If you are using LDAP, be aware of system responses to user and site management operations. For information about system responses, see Appendix D, “Managing Users, Sites, and Roles in LDAP-Integrated CS Systems.”

Creating a Role

To create a new role

1. In the **Admin** tab, expand **Roles**, then double-click **Add New**.

   Content Server displays the “Add New Role” form.



Add New Role

*Name:

*Description:

2. In the **Name** field, enter a unique name of up to 32 characters.
3. In the **Description** field, enter a short and informative description of no more than 255 characters.
4. Click **Add New Role**.
5. Add the role to the default tree tabs (**Workflow**, **Site Admin**, **Admin**, **Site Plan**, and **Active List**), as appropriate. For instructions, see “Editing Tree Tabs,” on page 123.

Editing a Role

Although you cannot change the name of a role after you have created it, you can edit the description of the role.

To edit the description of a role

1. In the **Admin** tab, double-click **Roles**.
2. In the list of roles, navigate to the role you want to edit and click its **Edit** (pencil) icon.
3. In the “Edit Role” form, make your changes, then click **Save**.

Deleting a Role

Caution

Do not delete any of the system default roles: GeneralAdmin, SiteAdmin, WorkflowAdmin.

To delete a role

1. In the **Admin** tab, double-click **Roles**.
2. In the list of roles, navigate to the role you want to delete and click its **Delete** (trash can) icon.

   Content Server displays a confirmation message.
3. Click **Delete Role**.

   The role has been deleted.

Chapter 5

Configuring Users, Profiles, and Attributes

Users can be created through one of the following options: Content Server's native user manager (in the Advanced interface), or via LDAP. However, to configure user profiles and attributes, you must use the Advanced interface.

This chapter outlines your options for creating users. This chapter also shows you how to create users in Content Server, and configure their profiles and user attributes. For information about creating users in external sources, refer to the product documentation.

This chapter contains the following sections:

Overview

CS users can be created through Content Server's native user manager, or through external user managers such as LDAP.

Every CS user is defined by the following set of data:

- User account, which gives the user access to the Content Server system and its database tables
- User profile, which is required for users who will be working with:
    - CS applications
    - Language packs and setting a default language
    - Workflow processes, in which e-mail messages will be sent to notify workflow participants of their assignments. The user profile supports workflow actions by mapping a user name to an e-mail address.
- User attributes (in addition to the e-mail and locale attributes in the user profile), if actions and events in addition to workflow must be supported.

Once users are created and configured, they must be associated, by means of roles, with the sites they are to work in. This chapter shows you how to create and configure users. Procedures for associating users with sites are given in Chapter 7, "Assembling and Organizing CM Sites."

User Management Options

Content Server's Directory Services API enables your CS system to connect to external directory servers or user managers that contain authentication information, user information, and so on. The following connection options are available:

- Native system—The Content Server native user manager, which uses the native Content Server user management tables `SystemUsers` and `SystemUserAttrs`.
- The LDAP plug-in—With this option, user names and attributes are stored in the directory server rather than in the Content Server database.

Because Content Server security is based on ACLs, any external user management system (such as LDAP) must be configured to match the Content Server ACLs.

Information about configuring the plug-ins is given in our guide, *Configuring Third-Party Software*. Properties that configure the plug-ins are located in the files `futuretense.ini` (the **Authentication** tab), `ldap.ini`, and `dir.ini` files. The files are described in the *Property Files Reference*.

Note

This guide uses the native Content Server user manager throughout.

Native Content Server User Manager

If you are using the native Content Server user manager, follow the guidelines in Chapter 3, "Site Configuration Guidelines" to create and configure users, and then grant them access to the management system.

LDAP Plug-In

If you are using LDAP to manage your users on either the management or the delivery system, you create user accounts with LDAP rather than with the Advanced interface. However, you must still use the Advanced interface to create ACLs and roles in the Content Server database. Instructions for granting users access to a CS management system integrated with LDAP are given in our guide, *Configuring Third-Party Software.*

Configuring Users in Content Server

Each Content Server user is completely defined by a user account, user profile, and, if necessary, user attributes.

- A user account is required for anyone who is to work with Content Server.
- A user profile is required for users who will be working with CS modules and products, setting a default language, and participating in workflow processes in which e-mail messages will be sent.
- User attributes, in addition to the locale and e-mail attributes in the user profile, may also be required for your operation. If so, the additional attributes can be created.

Once you have created the user, you must enable that user for the appropriate sites by assigning roles to the user name for each site the user will work in. For information about enabling users after you have created them, see "Granting Users Access to a Site (Assigning Roles to Users)," on page 98.

After you have created and enabled a new user, be sure to give that user the following information:

- The user name/password combination of the user account.
- The URL to the CS applications on the management system:

  ```
  http://<server>:<port>/<context>/Xcelerate/LoginPage.html
  ```

  where

  `<server>` is the host name or IP address of the machine running Content Server. Depending on how the system was set up, you might also need to include the port number— `server:8080` for example; and

  `<context>` is the name of the web application on the same server.

The rest of this chapter shows you how to create user accounts, profiles, and attributes, as well as modify and delete them.

Working with User Accounts

This section shows you how to create, edit, and delete user accounts in Content Server.

Note

This section provides procedures for creating, editing, and deleting users in the Advanced interface. If you are using LDAP, refer to the LDAP product documentation. Also, be sure to substitute the word "group" for the word "ACL" and create users who belong to the groups with these names.

Creating a New User

Note

This section shows you how to create users in the Advanced interface. If you are using LDAP, refer to the LDAP product documentation.

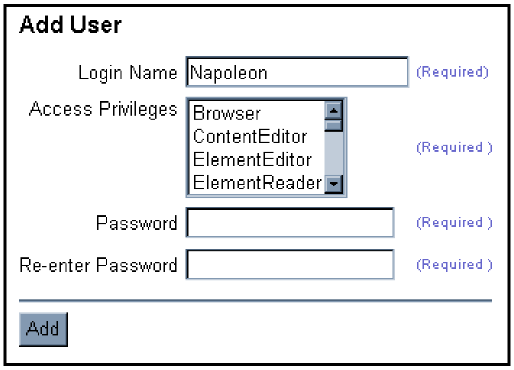
Before creating a user

1. Before creating a user, determine the user's:
    - Login name
    - Password
    - ACLs, which regulate the user's access to Content Server's database tables.
        - To determine the user's required ACLs, see "Required ACLs for Custom Users," on page 360.
        - To determine the user's additional system ACLs, see "System ACLs," on page 356.
        - To determine which ACLs are assigned to the database tables the user must access, follow the steps in "Assigning ACLs to Custom Tables," on page 69.

To create a user

2. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **User**.
3. In the form that appears, select **Add User** and click **OK**.

4. Fill in the “Add User” form as follows:



Add User

Login Name	Napoleon	(Required)
Access Privileges	Browser ContentEditor ElementEditor ElementReader	(Required)
Password		(Required)
Re-enter Password		(Required)

   a. In the “Login Name” field, enter a unique name. Do not include spaces or special characters, such as punctuation. The underscore character (_) is allowed.

   b. In the “Access Privileges” list, select ACLs for the user. To select multiple ACLs, **Ctrl-click** each desired ACL; you can also select a range of ACLs by **Shift-clicking** the first and last ACL in the range.

   c. Enter the same password into the **Password** and **Re-Enter Password** fields.

   d. Click **Add**.

      The user has been created.

To follow up with post-creation procedures

5. If the user will be implementing any of the following options:

   - Content Server products such as Engage
   - Language packs and different languages
   - Workflow processes that send e-mail messages

   create a profile for the user. For instructions, see “Creating and Editing a User Profile,” on page 82.

6. If the user requires attributes in addition to or in place of locale and e-mail (specified in the user profile), create the attributes. For instructions, see “Modifying, Adding, and Deleting User Attributes,” on page 83.

7. Once the user has been completely defined, you must associate the user with a site by means of roles.

   a. If you have not already done so, create roles for the user, following instructions in “Creating a Role,” on page 72.

   b. To associate the user to the site, create the site and add the user to the site. For instructions, see “Creating a Site,” on page 94, and “Granting Users Access to a Site (Assigning Roles to Users),” on page 98.

Editing a User

Note

This section shows you how to edit users in the Advanced interface. If you are using a LDAP, refer to the LDAP product documentation.

To edit a user

Caution

Do **not** change the names or ACLs of CS system users (`DefaultReader`, `Content Server`, `xceladmin`).

1. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **User**.
2. In the form that appears, enter the name of the user you want to work with. If you do not know the user name, leave the field blank; Content Server will return a list of all users in the system.
3. Select **Modify User** and click **OK**.
4. In the list of users, select the user you want to work with.
5. In the “Modify User” form, make the desired changes, then click **Modify**.

Deleting a User from the System

Note

This section shows you how to delete CS users using the Advanced interface. If you are using LDAP, refer to the LDAP product documentation.

To delete a user from the system

Caution

Do **not** delete any of the CS system users (`fwadmin`, `Content Server`, or `DefaultReader`).

1. Delete the user profile, as shown in “Deleting a User Profile,” on page 83, then continue with the next step.
2. Delete the user:
   a. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **User**.

b. In the form that appears, enter the name of the user you want to delete. If you do not know the user name, leave the field blank; Content Server will return a list of all users in the system.

c. Select **Delete User** and click **OK**.

d. In the list of users, select the **Delete** radio button next to the user you want to delete.

e. Click **Delete**.

   Content Server displays a warning message.

f. Click **OK**.

   The user has been deleted.

Working with User Profiles and User Attributes

A user profile is required for any user who will be working with the following:

- CS modules and products
- Language packs (and languages must be enabled)
- Workflow processes in which e-mail messages will be sent to notify workflow participants of their assignments. The user profile supports workflow actions by mapping a user name to an e-mail address.

A **user profile** holds a set of **user attributes**. By default, the only user attributes a user profile holds are:

- The **email attribute**, which is used to support workflow actions and takes the user's e-mail address as a value. (You can create workflow actions that send workflow participants e-mail about the assets that are assigned to them.)
- The **locale attribute**, which is used to determine which language to use when your Content Server system has a language pack installed. This attribute takes the user's preferred location as a value.

You can add more user attributes and store values for them in the Content Server user management tables if you want to. However, to use these values in the Content Server interfaces requires you to customize the elements that display the user profile forms. For information about customizing elements for the Content Server user interface, see the *Content Server's Developer's Guide*.

Creating and Editing a User Profile

Note

If you are using LDAP, be aware of system responses to user and site management operations. For information about system responses, see Appendix D, "Managing Users, Sites, and Roles in LDAP-Integrated CS Systems."

To create or edit a user profile

1. In the **Admin** tab, double-click **User Profiles**.

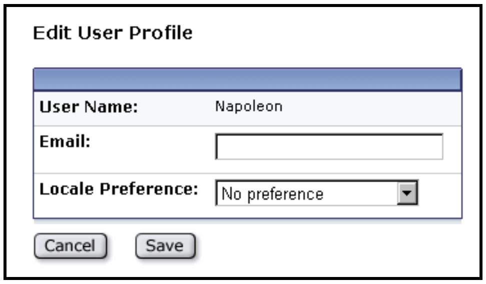
   Content Server displays the "User Profile Management" form.

2. In the form, enter the desired user name and click **Select**.

   Content Server displays the profile of the selected user.

3. Click **Edit**.

   Content Server displays the "Edit User Profile" form.



Edit User Profile

User Name: Napoleon

Email:

Locale Preference: No preference

4. In the **Email** field, enter the user's e-mail address.
5. (Optional) If one or more language packs are installed on your CS system, select a locale preference for this user from the "Locale Preference" drop-down list.
6. Click **Save**.
7. Enable this user for the sites the user needs to work with. See "Granting Users Access to a Site (Assigning Roles to Users)," on page 98.

Deleting a User Profile

To delete a user profile

1. In the **Admin** tab, double-click **User Profiles**.
2. In the "User Profile Management" form, click **Delete**.

   Content Server displays a warning message.
3. Click **Delete User Profile**.

   Content Server displays a message confirming the deletion.

Modifying, Adding, and Deleting User Attributes

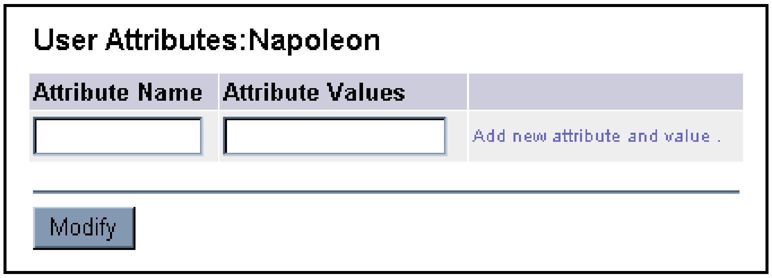
By default, the only user attributes that the CS content applications need are an e-mail address and locale preference. You use the user profile feature to assign these attributes to a user, as shown in "Creating and Editing a User Profile," on page 82. If you need to, you can store and use additional user attributes for your users in this table, even if you are using LDAP.

The **Modify User Attributes** option allows you to modify the attributes that are used in the user profile. It also allows you to add and delete attributes.

To modify a user's attributes

1. In the **Admin** tab, expand **Content Server Management Tools**, then double-click **User**.
2. In the form that appears, enter the name of the user you want to work with. If you do not know the user name, leave the field blank; Content Server will return a list of all users in the system.
3. Select **Modify User Attributes** and click **OK**.
4. In the list of users, select the user whose attributes you want to modify.

5. Fill in the “User Attributes” form, as explained below:



User Attributes:Napoleon

Attribute Name	Attribute Values	
		Add new attribute and value .

Do one of the following, as required:

- Change the current value (or values) assigned to an attribute by editing the contents of the **Attribute Values** field.
- Delete any unwanted attributes by deleting the associated value (in the **Attribute Values** field).
- Add a new attribute by entering its name and at least one value in the fields at the bottom of the form.

6. Click **Modify**.

Content Server commits your changes to the database.

Chapter 6

Setting Up External Security

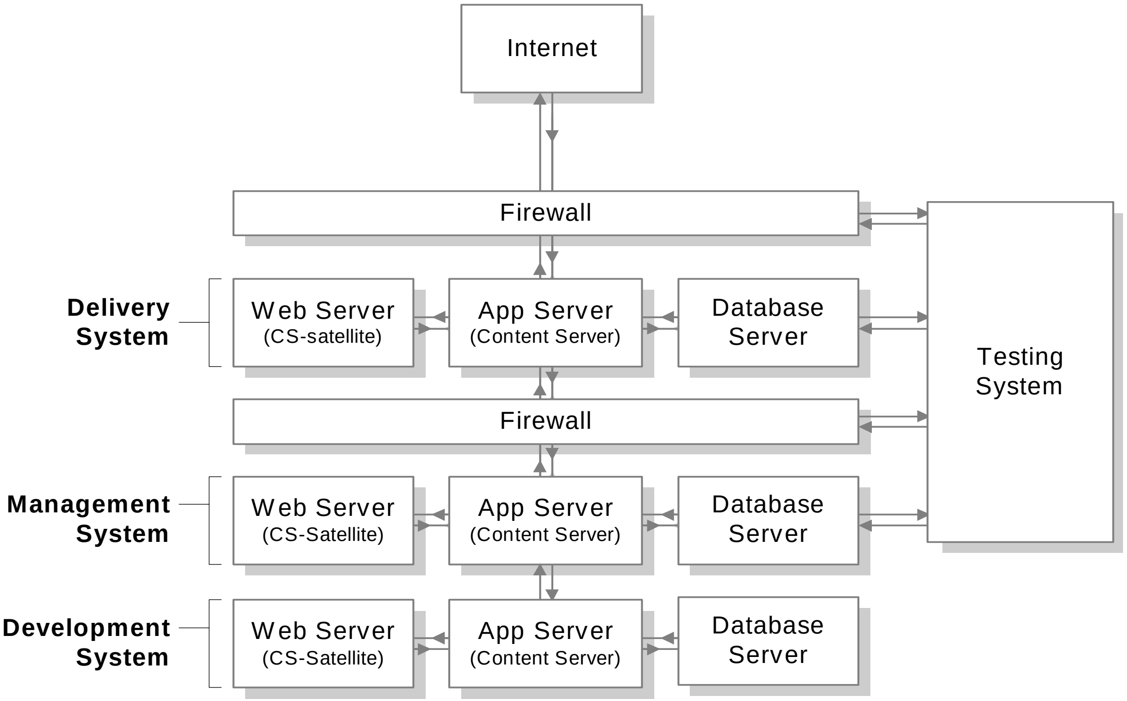
Determining what your security protocols should be and then implementing them is an important part of a CS administrator’s duties. You and the site developers must discuss and determine how security will be configured on both the management and the delivery systems before they start coding templates and designing asset types.

This chapter contains the following sections:

Overview

Before developers begin designing the online site or contemplating changes to the user interface on the management system, you must determine and implement your security protocols. The decisions you make about security configuration affects the way that you code and implement your online site.

At regular intervals, you should review your systems to determine whether they are working as they should. The following figure shows an example of a secure CS system:


Internet
Firewall
Delivery System
Web Server (CS-satellite)
App Server (Content Server)
Database Server
Firewall
Management System
Web Server (CS-Satellite)
App Server (Content Server)
Database Server
Development System
Web Server (CS-Satellite)
App Server (Content Server)
Database Server
Testing System

ACLs and Security

As mentioned in Chapter 4, “Working with ACLs and Roles,” ACLs (access control lists) serve as the foundation of the security and user management model in your CS system. ACLs are sets of permissions that restrict access to both database tables and Content Server pages.

ACL restrictions are enforced only when the `cc.security` and the `security.checkpagelets` properties in the `futuretense.ini` file are set to `true`. These properties are set to `true` by default. If you need to disable security for any reason, open the Property Editor and set the `cc.security` property to `false`. However, FatWire recommends that you keep this property set to `true` on all systems to ensure that site development is designed to work with security enabled.

When planning security measures for your system, examine the list of default ACLs (they are described in Appendix A, “System Defaults”), and determine whether you need additional ACLs. You might need to create an ACL with a different combination of permissions than any of the system ACLs provide if your developers plan to create new tables or to make additions to the user interface.

Note

Under no circumstances should you modify a system default ACL or modify the ACLs assigned to a Content Server system table or a CS content application table.

DefaultReader, secure.CatalogManager, and secure.TreeManager

Content Server and the CS content applications are delivered with several default user accounts, one of which is named DefaultReader. Because anyone who visits a site hosted by Content Server must have a Content Server user identity, the DefaultReader user is assigned to any unidentified (unauthorized) visitor.

The DefaultReader user account has one ACL: Browser. Because many of the Content Server database tables have the Browser ACL assigned to them, this means that someone could log in to a Content Server database as DefaultReader using Content Server Explorer and examine information about your system (although they cannot write to any tables as this user).

To prevent someone from using the DefaultReader user account to see tables in your Content Server database, set the following properties in the `futuretense.ini` file to `true`:

- `secure.CatalogManager`
- `secure.TreeManager`

When these properties are set to `true`, the DefaultReader user cannot access the CatalogManager and TreeManager servlets—not even for read-only data.

BlobServer and Security

When you want to implement security for your blobs you must enable the security feature for the BlobServer servlet. You do this by setting the `bs.security` parameter in the `futuretense.ini` file to `true`.

When `bs.security=true`, BlobServer refuses to serve a blob unless the URL for the blob contains evidence that the person requesting it has been authenticated as a valid user. What evidence? A value in the URL from the `csblobid` parameter whose value matches a session variable named `csblobid`. Therefore, when BlobServer security is enabled, your developers must code links to blobs differently than usual.

For information about how those links should be coded, see the *Content Server Developer's Guide*.

Security Goals

Typically, you have a different set of security goals for each of your CS systems: the development, management, and delivery systems.

Security Goals for the Development System

Even though development systems are typically behind firewalls, you should implement the same security configuration on the development system that will be in place on the system the developers are designing for (management or delivery). Why? To be sure that the code will work properly on the target system, because the same conditions exist on both systems:

- When you plan to use BlobServer security on the delivery system, templates that create URLs for blobs must be coded differently. Therefore, you must enable BlobServer security on the development system as well.
- If your online site will require that visitors identify themselves, you must have the same security configuration in place on the development system that you will use on the delivery system.

Security Goals for the Management System

Security on the management system encompasses two main concepts:

- Ensuring that only valid CS users can access the system (described in this chapter).
- Ensuring that those valid users can access only the functions that are appropriate for them. For information, see Chapter 4, "Working with ACLs and Roles."

Security Goals for the Delivery System

The precautions that you take on the delivery system are more stringent by nature because when you deliver a site to the public, you must ensure that while visitors can access the content on your site, hackers cannot access areas that you do not want made public.

When you configure your delivery system, you disable the Content Server user interfaces to prevent anyone from adding assets or code through the interfaces or with any of the developer tools. Additionally, you restrict access to some of the Content Server servlets.

Implementing Security

This section describes the steps you must take to configure the security measures you have decided to implement: setting properties, changing passwords for default user accounts, mapping URLs for specific Content Server servlets, and disabling certain parts of the applications on the delivery system.

Each section states which steps should be performed on which system or systems.

Properties That Configure Security Settings

This section describes how to configure the properties in the `futuretense.ini` file that implement various kinds of security.

For All Systems

Use the Property Editor to open the `futuretense.ini` file and verify that the following properties are set to `true`:

- `cc.security`
- `security.checkpagelets`

- `secure.CatalogManager`
- `secure.TreeManager`

If you plan to use BlobServer security, set the following property to `true`:

`bs.security`

To ensure that the session timeout value is appropriate for each system, set the following property, as well:

`cs.timeout`

On the development and management systems, set the timeout value for as long as you think that you safely can. On the delivery system, set the timeout value for as short a time as you can without frustrating your visitors.

For the Delivery System

In addition to the properties described in the preceding section, there is one more property to specify for the delivery system: `cs.wrapper`.

"Wrappers" are the static HTML files located in the `futuretense_cs` directory on the web server. Also included in that directory is a subdirectory named Dev, which you should remove from your delivery system.

However, if you decide to remove the entire `futuretense_cs` directory from the web server on the delivery system, you must set the `cs.wrapper` property to `false`.

For more information, see "CS Forms and Pages (Delivery System Only)," on page 90.

Users and Passwords

Be sure that the default user accounts were made secure after CS was installed on all systems.

For All Systems

Complete the following steps on all systems—development, management, and delivery:

- Change the default password for the `fwadmin` user account (**User** node under **Content Server Management Tools** in the **Admin** tab in the Advanced interface).
- If the Content Server user that was created during the installation is named `ContentServer`, verify that its password is not the default password (`password`). If the password used is the default password, change it (**User** node under **Content Server Management Tools** in the **Admin** tab in the Advanced interface).
- Change the default administrator username/password for your application server.
- Change the default administrator username/password for your web server.
- If you have the sample sites installed, a mirror user was created for the Mirror to Server publishing method (name: `mirroruser`; password: `mirroruser`). Change the password for this user.
- For any user accounts that have the SiteGod ACL, change the password frequently (**User** node under **Content Server Management Tools** in the **Admin** tab in the Advanced interface). Handle any user account that has the SiteGod ACL as you would the UNIX `root` user.

Note

Do **not** change the default password for the DefaultReader user.

For the Management and Delivery Systems

It is best if the sample sites (HelloAssetWorld, Burlington Financial, and GE Lighting) are not installed on either the management or delivery systems.

If any of the sample sites were installed on such a system, **delete all the sample users including editor**, but do **not** delete the xceleditor ACL and do **not** delete the fwadmin user. Additionally, change the password for the mirroruser user.

For the Delivery System

Do **not** assign the xceleditor ACL to any user on the delivery system other than the system administrator of that system. This ACL allows access to the CS content applications installed on that system.

URLs and the Web Server (Delivery System Only)

On the web server of the delivery system, be sure to give access to the following Content Server servlets only. How? By mapping only their URLs to the application server:

- ContentServer
- BlobServer
- Satellite
- CookieServer

Do **not** map URLs to the application server for any of the other Content Server servlets. Instead, map the URLs for the following servlets to an error page such as the “404 Not Found” page:

- HelloCS
- CatalogManager
- TreeManager
- DebugServer
- CacheServer
- Inventory

CS Forms and Pages (Delivery System Only)

On the delivery system, be sure to disable or completely remove the following pieces of the CS applications:

- Remove the `futuretense_cs/Dev` directory from the **web server**. This directory holds forms that are useful for developers, which means that you do not want them on your delivery system.

Note

Do not remove this directory from the application server. Remove it from the **web server only**.

You can optionally remove the entire `futuretense_cs` directory from the web server. (Do **not** remove it from the application server.) If you do, be sure to set the `cs.wrapper` property in the `futuretense.ini` file to `false` (the wrappers are the static HTML files from that folder). If you forget, visitors on the site will see "404 Page Not Found" rather than some of the system error messages.

- Go to `SiteCatalog/OpenMarket/Xcelerate/UIFramework` and rename all of the pages. At the very least, rename `LoginPage` and `LoginPost`.

Testing Security

After you have implemented your security measures, test your systems.

Security Tests for All Systems

Complete the following steps on your development, management, and delivery systems:

1. Try to log in to the database with Content Server Explorer using the default user accounts:
   - `DefaultReader`

     If you can log in using `SomeReader` as the password, the `secure.CatalogManager` and `secure.TreeManager` properties are set to `false`. Change them to `true`.
   - `ContentServer`

     If you can log in using `password` as the password, change the password immediately.
   - `editor`

     If you can log in using `xceleditor` as the password, change the password immediately.
   - `fwadmin`

     If you can log in using `xceladmin` as the password, change the password immediately.
2. Verify that the sample site users do not exist on the management or delivery systems.
3. Verify that you cannot log in with `ContentServer/password` using a CatalogManager URL:

   ```
   http://<server>:<port>/<context>/
      CatalogManager?ftcmd=login&username=ContentServer&password=
      password
   ```
4. Verify that you cannot flush the entire cache as `ContentServer/password` using a CacheServer URL:

```
http://<server>:<port>/<context>/
   CacheServer?all=true&authusername=ContentServer&authpassword
   =password
```

5. Verify that you cannot log in to the application server as the default administrator user.
6. Verify that you cannot log in to the database as the default administrator user.
7. Verify that you cannot log in to the web server as the default administrator user.

Chapter 7

Assembling and Organizing CM Sites

CM sites can be assembled and organized once the components are created, The components are the data model, ACLs, roles, and users. This chapter shows you how to work with sites (create, edit, and delete sites); how to manage site users (add, edit, and delete users), and how to manage asset types for the sites (enable and remove asset types).

This chapter contains the following sections:

Overview

Users' access to sites is regulated by roles. For a user to gain access to a site, the user must be associated to the site by one or more roles. The roles allow the user to log on to the site and use certain interface functions. Roles also describe the user's place in workflow processes that might have been created for that site.

This section presents the basic procedures for creating, editing, and deleting sites, for associating users to sites, and for deleting users from the sites.

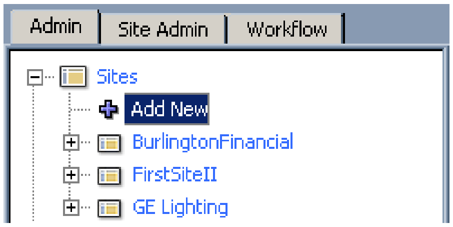
Working with Sites

Creating a Site

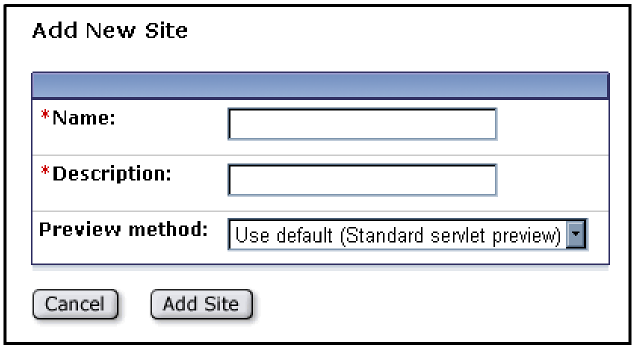
Creating a site means specifying a site name, description, and a method for previewing the content that the site will display.

To create a site

1. In the **Admin** tab, expand **Sites** and double-click **Add New**.



The "Add New Site" form appears.



Add New Site

*Name:	
*Description:	
Preview method:	Use default (Standard servlet preview)

2. In the **Name** field, enter a unique name of up to 255 characters.

Note

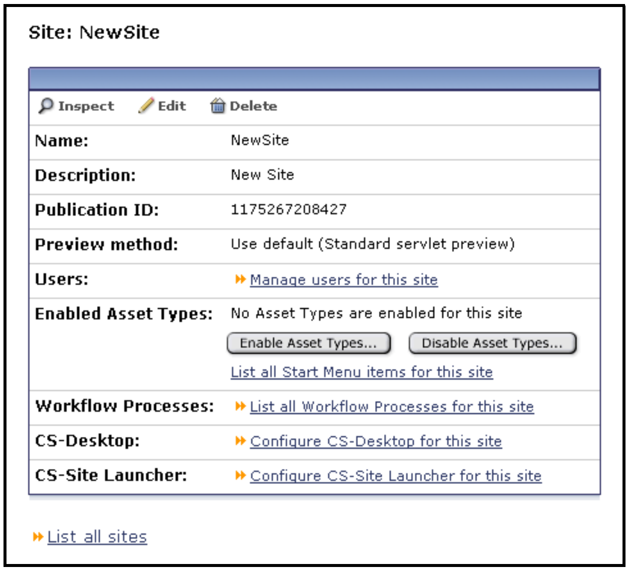
Be certain that you have entered a correct name for the site. Although you can edit the description of a site, you cannot change the name of a site after you have created it.

3. In the **Description** field, enter the name of the site as you want it to appear in the Content Server interfaces (you can use up to 64 characters). This is the name that will be used in site lists throughout the user interface.

4. In the **Preview method** field, select the method by which you will preview the site's content.

5. Click **Add**.

   The "Site" form appears, where you can execute a number of operations; for example, inspect, edit, or delete the site, manage users, or configure Site Launcher to replicate the site.



Site: NewSite

Name:	NewSite
Description:	New Site
Publication ID:	1175267208427
Preview method:	Use default (Standard servlet preview)
Users:	Manage users for this site
Enabled Asset Types:	No Asset Types are enabled for this site Enable Asset Types... Disable Asset Types... List all Start Menu items for this site
Workflow Processes:	List all Workflow Processes for this site
CS-Desktop:	Configure CS-Desktop for this site
CS-Site Launcher:	Configure CS-Site Launcher for this site

List all sites

When the new site is added to the **Sites** node, Content Server creates, below the site name, a set of sub-nodes that prompt you for site components:

- **Users** prompts you for users.
- **Asset Types** prompts you for asset types (created by developers).
- **CS-Desktop** prompts you to enable CS-Desktop.
- **CS DocLink** prompts you to enable CS-DocLink.

Each sub-node is equipped with a linking mechanism that allows you to associate the sub-node, and therefore the site, with data in the configuration pool. Because of this linking mechanism, the sub-nodes cannot be deleted or supplemented with custom sub-nodes. Selecting a sub-node allows you to create All data required by the sub-nodes must be entered in to the configuration pool, described next.

Note that the procedure above only adds new sites.

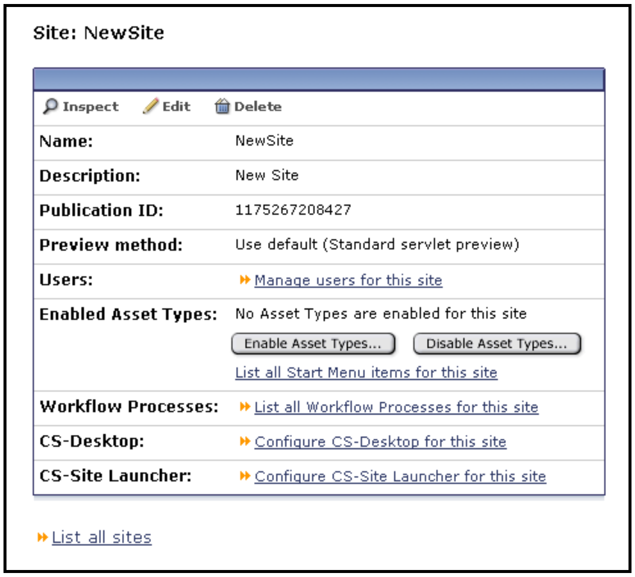
- To make a site usable, you must associate users and asset types (created by developers) with the site. If you plan to use workflow processes, you should create those processes as well.
- To copy or migrate the site to another Content Server system, use the Mirror to Server publishing method. See "The Dynamic Publishing Process," on page 259.

Obtaining Site Configuration Information

In certain situations, you will need to know how a site has been configured—for example, which asset types and workflow processes are enabled, and who are the site users.

To obtain site configuration information

1. In the **Admin** tab, expand **Sites** and double-click the desired site.
2. In the “Site” form, select the option that provides the information you are looking for.



Site: NewSite

Name:	NewSite
Description:	New Site
Publication ID:	1175267208427
Preview method:	Use default (Standard servlet preview)
Users:	Manage users for this site
Enabled Asset Types:	No Asset Types are enabled for this site Enable Asset Types... Disable Asset Types... List all Start Menu items for this site
Workflow Processes:	List all Workflow Processes for this site
CS-Desktop:	Configure CS-Desktop for this site
CS-Site Launcher:	Configure CS-Site Launcher for this site

List all sites

Editing a Site

Editing a site means changing its description and/or preview method. If you need to change the name of a site, you must delete the site and create a new one with the correct name.

To edit a site description

1. In the **Admin** tab, expand **Sites** and double-click the desired site.
2. In the “Site” form, click **Edit**.
3. In the “Edit” form, do the following:
    a. In the **Description** field, make the appropriate changes.
    b. In the **Preview Method** field, select a different preview method, if necessary.
4. Click **Save**.

Deleting a Site

When you delete a site, all configuration information for that site is either removed or unshared (if the site shared data with other sites). The following configuration information is affected by the deletion of a site:

- Start Menu items that were enabled for the site are deleted or unshared

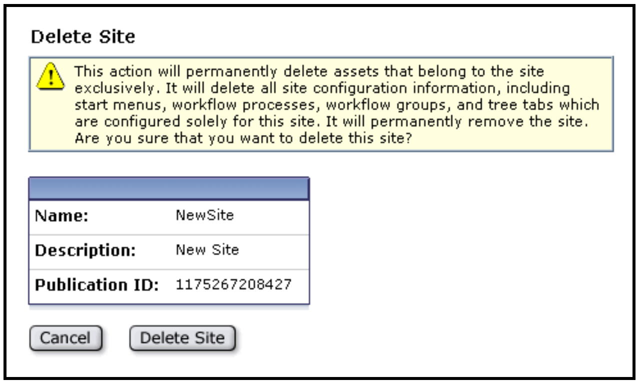
- Saved searches are deleted or unshared
- Publishing destinations are unshared
- Users are removed from the site
- Workflows that were shared are unshared or deleted
- Template assets that were shared have their SiteEntry removed
- Asset subtypes that were enabled for that site as well as tree tabs are removed or unshared

Therefore, before deleting a site, be sure that you have shared the assets that you need to keep.

To delete a site

1. In the **Admin** tab, double-click **Sites**.
2. In the list of sites, navigate to the site you want to delete and click its **Delete** icon.

   A warning is displayed to inform you that site components will be deleted. The warning identifies the components and prompts you to confirm your intention to delete the site.



Delete Site

This action will permanently delete assets that belong to the site exclusively. It will delete all site configuration information, including start menus, workflow processes, workflow groups, and tree tabs which are configured solely for this site. It will permanently remove the site. Are you sure that you want to delete this site?

Name:	NewSite
Description:	New Site
Publication ID:	1175267208427

3. Click **Delete Site**.

Managing Site Users

This section shows you how to grant users access to a site, view site users, change their role assignments on the site, and remove users from a site.

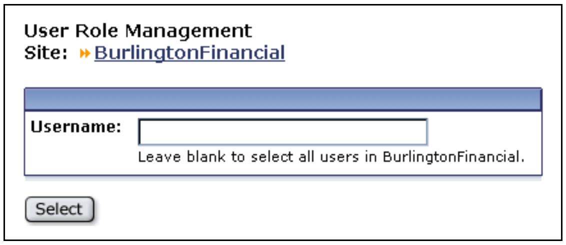
Granting Users Access to a Site (Assigning Roles to Users)

For a user to have access to a site, the user and the site must be assigned the same role (or roles). The assigned roles associate the user with the site, allow the user to log in to the site, and describe the user's place in the workflow processes that were (or will be) created for that site.

To give a user access to a site

1. In the **Admin** tab, expand **Sites**, then the desired site.
2. Double-click **Users**.

   The "User Role Management" form appears.



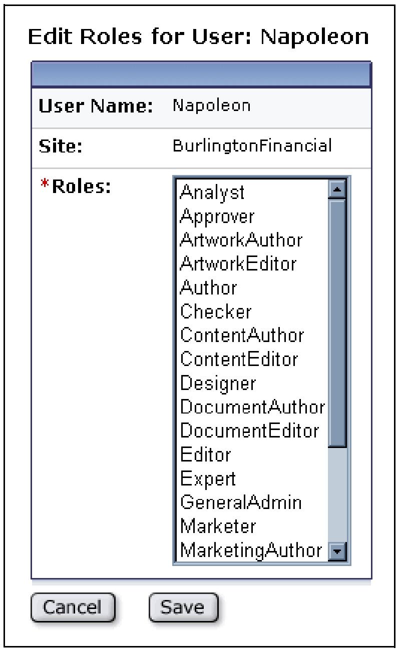
User Role Management
Site: BurlingtonFinancial

Username:
Leave blank to select all users in BurlingtonFinancial.

3. Enter the desired user name and click **Select**.

   The "User Role Management" form displays the user name with no assigned roles.
4. Click the **Edit** icon next to the user name.

Content Server displays the “Edit Roles” form:



Edit Roles for User: Napoleon

User Name:	Napoleon
Site:	BurlingtonFinancial
*Roles:	Analyst Approver ArtworkAuthor ArtworkEditor Author Checker ContentAuthor ContentEditor Designer DocumentAuthor DocumentEditor Editor Expert GeneralAdmin Marketer MarketingAuthor

5. Select one or more roles from the list. To select multiple roles, **Ctrl-click** each desired role; you can also select a range of roles by **Shift-clicking** the first and last role in the range.

> **Note**
>
> Remember that the user cannot perform the functions available to a given role unless the user has adequate ACL permissions.

6. Click **Save**.
7. Give the user access to site components by following procedures in Chapter 8, “Managing Access to CM Site Components.”

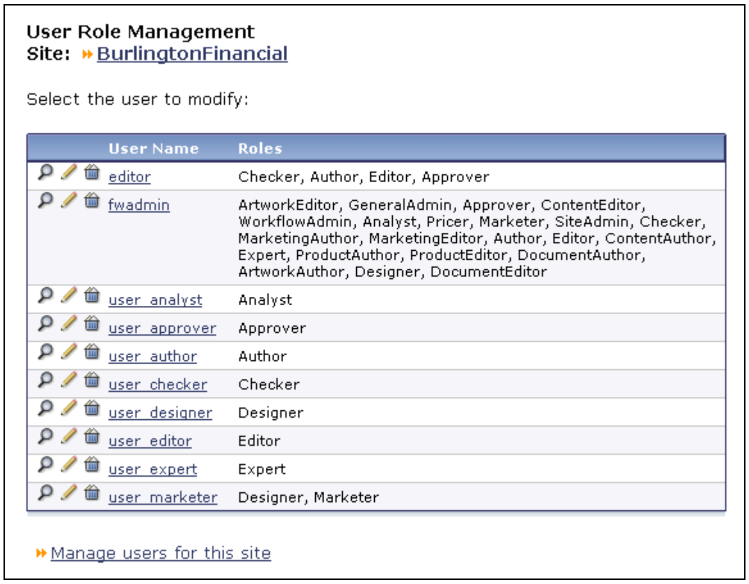
> **Note**
>
> After you have created and enabled the new user, be sure to give that user the following information:
>
> - The user name/password combination of the user account.
> - The URL to Content Server (and the CS content applications) on the management system.

Viewing Site Users and Changing Role Assignments

To view the list of users and roles currently defined for a given site, and to change role assignments

1. In the **Admin** tab, expand **Sites**, then the desired site.
2. Under the desired site, double-click **Users**.
3. In the “User Role Management” form, leave the user name field blank and click **Select**.

   Content Server displays a list of all the users currently assigned roles in the selected site:



User Role Management
Site: BurlingtonFinancial

Select the user to modify:

User Name	Roles
editor	Checker, Author, Editor, Approver
fwadmin	ArtworkEditor, GeneralAdmin, Approver, ContentEditor, WorkflowAdmin, Analyst, Pricer, Marketer, SiteAdmin, Checker, MarketingAuthor, MarketingEditor, Author, Editor, ContentAuthor, Expert, ProductAuthor, ProductEditor, DocumentAuthor, ArtworkAuthor, Designer, DocumentEditor
user_analyst	Analyst
user_approver	Approver
user_author	Author
user_checker	Checker
user_designer	Designer
user_editor	Editor
user_expert	Expert
user_marketer	Designer, Marketer

Manage users for this site

4. (Optional) If you want to modify the user’s roles, do the following:
   a. Navigate to the desired user name and click its **Edit** icon.
   b. In the “Edit Roles” form, select one or more roles from the list, then click **Save**.
   c. Repeat steps a and b for each additional user, as necessary.

Deleting Users from a Site

You can delete a user from a site without deleting that user from the system.

To delete a user from a site

1. In the **Admin** tab, expand **Sites**, then the desired site.
2. Under the desired site, double-click **Users**.
3. In the “User Role Management” form, enter the desired user name and click **Select**.
4. In the form that appears, click the **Delete** (trash can) icon next to the user.

   Content Server displays a warning message.
5. Click **Delete User from Site**.

Managing Asset Types

This section shows you how add asset types to a site (enable the asset types), and delete them from a site.

Enabling Asset Types for a Site

Enabling an asset type means associating the asset type with a chosen site or sites. Once the assignment is made, the asset type itself must be made available to users through Start Menu items, access permissions, and tree tabs, as shown in Chapter 8, "Managing Access to CM Site Components."

> **Note**
>
> The following procedure describes how to enable an asset type for use within the Dashboard and Advanced interfaces. Enabling an asset type for use with CS-Desktop is described in "Configuring the Word Asset Types for CS-Desktop," on page 312.

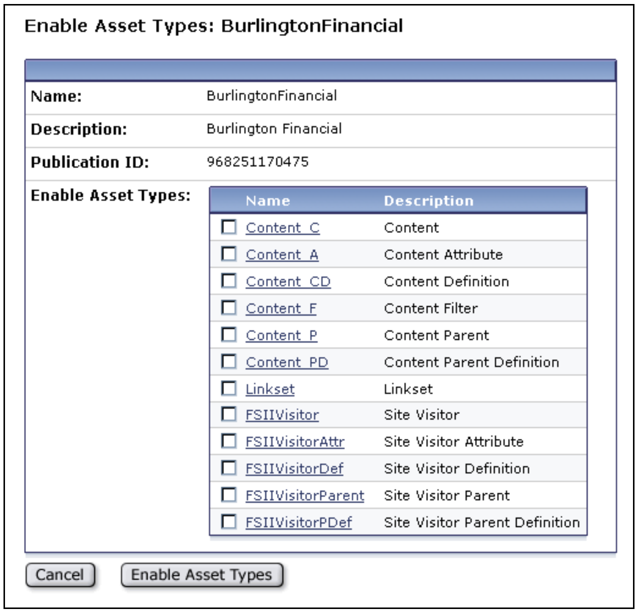
To enable asset types for a site

1. In the **Admin** tab, expand **Sites**, then the desired site.

> **Note**
>
> If you have the SiteAdmin role, but not the GeneralAdmin role, select the **Site Admin** tab.

2. Under the desired site, expand **Asset Types** and double-click **Enable**.

   Content Server displays the "Enable Asset Types" form, listing asset types that have not yet been enabled in this site:

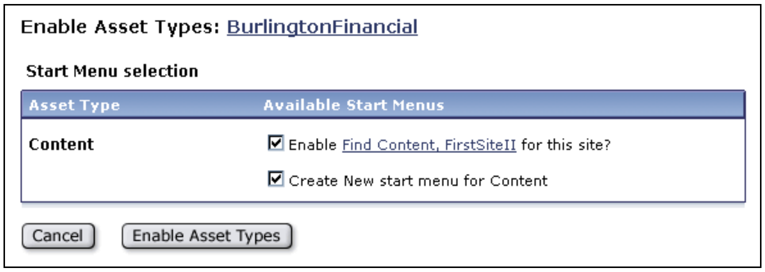


Enable Asset Types: BurlingtonFinancial

Name:	BurlingtonFinancial
Description:	Burlington Financial
Publication ID:	968251170475
Enable Asset Types:	

Name	Description
☐ Content_C	Content
☐ Content_A	Content Attribute
☐ Content_CD	Content Definition
☐ Content_F	Content Filter
☐ Content_P	Content Parent
☐ Content_PD	Content Parent Definition
☐ Linkset	Linkset
☐ FSIIVisitor	Site Visitor
☐ FSIIVisitorAttr	Site Visitor Attribute
☐ FSIIVisitorDef	Site Visitor Definition
☐ FSIIVisitorParent	Site Visitor Parent
☐ FSIIVisitorPDef	Site Visitor Parent Definition

3. Select the asset types which you want to enable for this site.
4. Click **Enable Asset Types**.
5. (Optional) In the form that follows, check the box next to each Start Menu Item that you would like Content Server to create. (For more information on Start Menu items, see "Managing Access to Asset Types," on page 108.)



Enable Asset Types: BurlingtonFinancial

Start Menu selection

Asset Type	Available Start Menus
Content	☑ Enable Find Content, FirstSiteII for this site? ☑ Create New start menu for Content

If you choose not to generate Start Menu items at this time, you must manually create them later. No one can create assets of the asset types you just enabled until the corresponding Start Menu items are created.

6. Click **Enable Asset Types**.
7. The asset types are now enabled for the site. If you need to create Start Menu items for them, continue to "Managing Access to Asset Types," on page 108.

Removing Asset Types from a Site

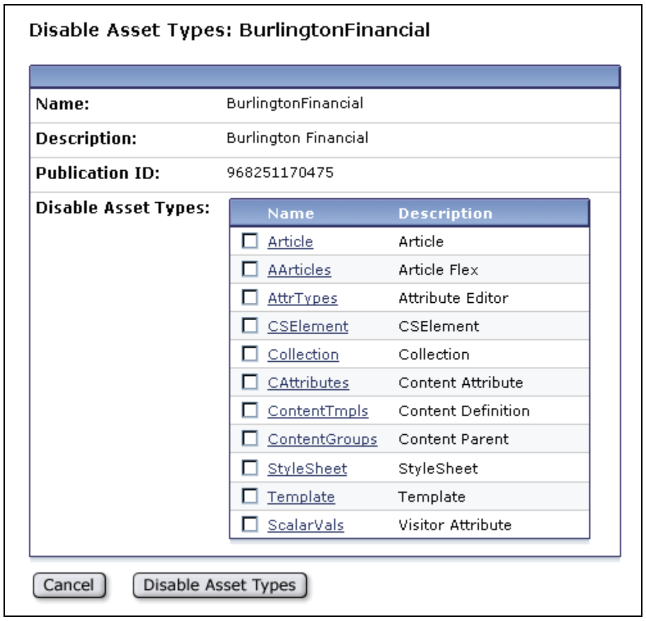
To remove asset types from a site

1. In the **Admin** tab, expand **Sites**, then the desired site.

> **Note**
>
> If you have the SiteAdmin role, but not the GeneralAdmin role, select the **Site Admin** tab.

2. Under the desired site, expand **Asset Types** and double-click **Disable**.

Content Server displays the “Disable Asset Types” form, showing asset types that have been enabled for the selected site:



Disable Asset Types: BurlingtonFinancial

Name:	BurlingtonFinancial
Description:	Burlington Financial
Publication ID:	968251170475
Disable Asset Types:	

	Name	Description
□	Article	Article
□	AArticles	Article Flex
□	AttrTypes	Attribute Editor
□	CSElement	CSElement
□	Collection	Collection
□	CAttributes	Content Attribute
□	ContentTmpls	Content Definition
□	ContentGroups	Content Parent
□	StyleSheet	StyleSheet
□	Template	Template
□	ScalarVals	Visitor Attribute

3. Select the asset types that you want to remove from the site.
4. Click **Disable**.

> **Note**
>
> If there are start menu items or workflow processes that apply to an asset type that you disabled for a site, be sure to edit the start menu item or workflow process so that it reflects the change as well.

Enabling and Managing User Interfaces

If you plan to enable the Desktop and DocLink interfaces for users on certain sites, follow instructions in Chapter 18, “Configuring the User Interfaces.”

Chapter 8

Managing Access to CM Site Components

This chapter shows you how to manage access to site components. It is assumed that users have already been given the permission to access the site containing those components.

This chapter contains the following sections:

Overview

Once users are granted permission to access a site (as shown in Chapter 7, “Assembling and Organizing CM Sites”) their ability to work with the site’s content is managed through

- Start menu items, which determine the users’ access to asset types on the site; and
- Permissions to the tree, including
    - Tabs in the tree, and
    - Nodes in the tabs.

Roles are the components that you invoke in order to associate users with interface functions and content that the users must manage.

Start Menu Items

Start menu items are the entries in the Content Server interfaces representing the types of assets you can create, copy, and search for. In the Advanced interface, these entries are shown when you click **New** or **Search** in the button bar, respectively. In the Dash interface, the “New” start menu entries appear in the “Create New” drop down list and as icons in the quick access pane, while the “Search” start menu items determine your permissions to search for assets of the corresponding types.

You configure the start menu items to determine the following about the new assets that are created with them:

- Which roles have access to the start menu item. Specifying roles controls which content providers can create or search for assets of this type.
- Field/value pairs. You can supply values for certain fields so that those fields are automatically filled in whenever a content provider creates a new asset with the shortcut.

  For example, for basic assets, it might be useful to have the start menu item set a pre-determined value for the **Template**, **Category**, **Subtype**, or **Source** field (or some combination). If your basic assets have named associations that can be different based on the subtype of the asset, be sure to create start menu items that set a subtype for new assets so that the only the appropriate named associations appear in the “New” form.

  For flex assets, it is useful to have start menu items that set the flex definition for new flex assets so the content providers have fewer steps.
- Which workflow process to assign by default to new assets created with the start menu item and the users who participate in that workflow process.

Start menu items also determine which options are available for an asset type that is listed in a tree tab (in the Advanced interface). If an asset type is displayed on a tree tab, but there is no start menu item created for it, the options listed in the right-mouse menu for the asset type are **Refresh, Edit**, and **Inspect** only. To have the **New** option included in the right-mouse menu, there must be a **New** start menu item created for it.

For Different Kinds of Users

If you have different groups of content providers who create different kinds of content, you could create start menu items for each type.

For example, suppose that you have a group of writers who create business articles. You could create a start menu item that creates a new article, sets the Category to Business, sets the template to the correct template for articles on the business page, and places it in a workflow for business articles. Additionally, if you create a role for business writers, you can configure the start menu item so that only the users with the business writer role can use this start menu item.

For Flex Assets

For flex assets, you can set the definition of a flex asset with the start menu item.

For example, suppose that your public site were a catalog of household goods and that your flex assets were products. You could create a start menu item named “Toaster” that sets the product definition to “toaster” for new toaster products, a start menu item named “glassware” that sets the appropriate product definition for new glassware products, and so on. A start menu item that sets the flex definition in this way eliminates a step for your content providers, thereby reducing the chance that they will select the wrong definition by mistake.

Tree Tabs (Advanced interface only)

A tree tab is a tab that appears in the left-hand panel of the Advanced interface. Access to tree tabs is, again, controlled through roles.

There are several default tree tabs that are core to Content Server:

- **Active List** – displays the assets that you selected and placed on your Active List. All users see this tab as soon as they use the **Move to Active List** button to place an asset on their active list.
- **Admin** – displays the administrative functions that affect all of the sites in the system. By default, only users with the default system role named GeneralAdmin have access to this tab.
- **Design** — serves as the source for creating pages on your site. Some of these sources are: Templates, Product Definition, Content Definition, and other sources for the creating pages.
- **History** – displays the assets that you worked with during the current session. All users see this tab as soon as they create, inspect, edit, or copy their first asset.
- **Query** – provides access to the query asset type. You can run queries and see their results on the tab.
- **Site Admin** – holds a subset of the administrative functions that apply only to the site that you are currently logged in to. By default, only users with the default system role named SiteAdmin have access to this tab. This tab is useful if you have individuals who manage which existing users have access to individual sites, but who do not need to create new users or new sites.
- **Site Plan** – displays a graphical and hierarchical representation of the pages, collections, queries, and content assets that make up the online site that you are making available on the delivery system. By default, all of the sample site roles and system default roles grant access to this tab.
- **Workflow** – has the workflow configuration functions. By default, only users with the Workflow Admin role have access to this tab.

You can create new tree tabs that give your users another method of creating or editing assets of specific types (in addition to the start menu lists, that is.)

For example, the **Design** tab (a core tab displayed on the GE Lighting sample site) gives users with the Designer role access to attribute editors, product definitions, and so on. The **HelloContent** tab in the HelloAssetWorld sample site gives users with the HelloAuthor, HelloEditor, and HelloDesigner roles access to HelloArticles and HelloImages.

When you create a new site, Content Server adds that site to the list of sites that are enabled for all the tree tabs. However, the roles that create for your site cannot be automatically assigned to a tab—you must specify which tabs a new role has access to.

Managing Access to Asset Types

Managing user's access to asset types on a site entails creating start menu items for the asset types.

Creating Start Menu Items

A start menu item is a way of coupling a user with the asset types he is permitted to work with. When you create a start menu item, you specify

- the sites the item is for, and
- the roles at those sites that are allowed to create or search for assets of those types.

In other words, start menu items determine which users can create and search for assets of specific types on a specific site.

CS supports four kinds of start menu items:

- **New** – creates a link for the asset type on the list that is displayed when you click **New** in the button bar (Advanced interface) or in the "Create New" drop-down list and quick access pane (Dashboard interface). If you do not create **New** items for the asset types that are enabled for a site, no one can create assets of those types. For information about creating **New** start menu items, see "Creating "New" Start Menu Items," on page 109.
- **Search** – creates a link for the asset type on the list that is displayed when you click **Search** in the button bar (Advanced interface). If you do not create **Search** items for the assets that are enabled for a site, no one can search for assets of those types in any of the CS interfaces. For information about creating **Search** start menu items, see "Creating "Search" Start Menu Items," on page 112.
- **CS-Desktop** – makes the asset type available to content providers who use the CS-Desktop interface instead of one of the browser-based interfaces. For information about creating CS-Desktop start menu items, see "Configuring the Word Asset Types for CS-Desktop," on page 312.
- **CS-DocLink** – makes the asset type available to content providers who use the CS-DocLink interface instead of one of the browser-based interfaces; also configures document assets for the portal. For information about creating CS-DocLink start menu items, see "Configuring the CS-DocLink Asset Types," on page 319.

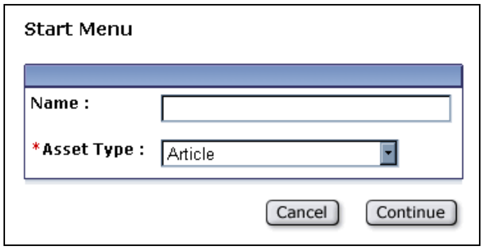
Creating “New” Start Menu Items

Before you begin creating start menu items, note the following:

- Be sure to create start menu items for only the asset types that are enabled for that site. If the assets are not enabled, go back to Chapter 7, “Assembling and Organizing CM Sites” and complete the steps in “Enabling Asset Types for a Site,” on page 101.
- If you want to specify a default workflow process, you should create the workflow process first. For information about workflow, see Chapter 9, “Creating and Managing Workflow Processes.”
- Talk to your developers to find out which fields are used by queries, collections, or other design elements for the online site. That way, you can be sure to specify default values for those fields.
- Talk to your developers to find out which templates are appropriate for the various subtypes of asset types that you are creating start menu items for.
- You **cannot** set values for the following kinds of fields:
  - An upload field or any field that writes data to a URL column.
  - Date fields
  - Fields that accept more than one value.
  - Association fields.
  - Flex attribute fields for flex and flex parent assets.

To create a “New” start menu item

1. In the **Admin** tab, expand **Start Menu**, then double-click **Add New**.
2. In the “Start Menu” form, do the following:



Start Menu

Name :

*Asset Type : Article

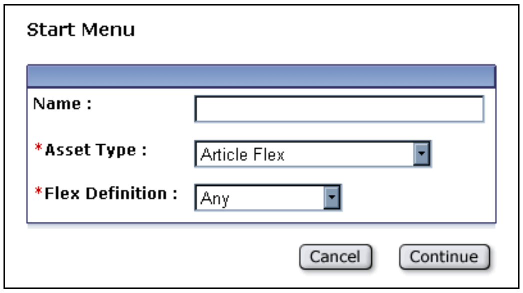
   a. In the **Name** field, enter the name of the item, using up to 64 characters. Although it is not required, it is a good idea to start the name with the word “New” (for example, “New Article” or “New Template”). The name that you enter will be displayed on the **New** list.

   b. In the “Asset Type” field, select the type of asset for which you are creating the Start Menu item.

   c. Click **Continue**.

3. If you have chosen a flex asset type, continue with this step. Otherwise skip to step 4.

If you have chosen to a create a start menu item for a flex asset type, you are presented with the following form:



Start Menu

Name :

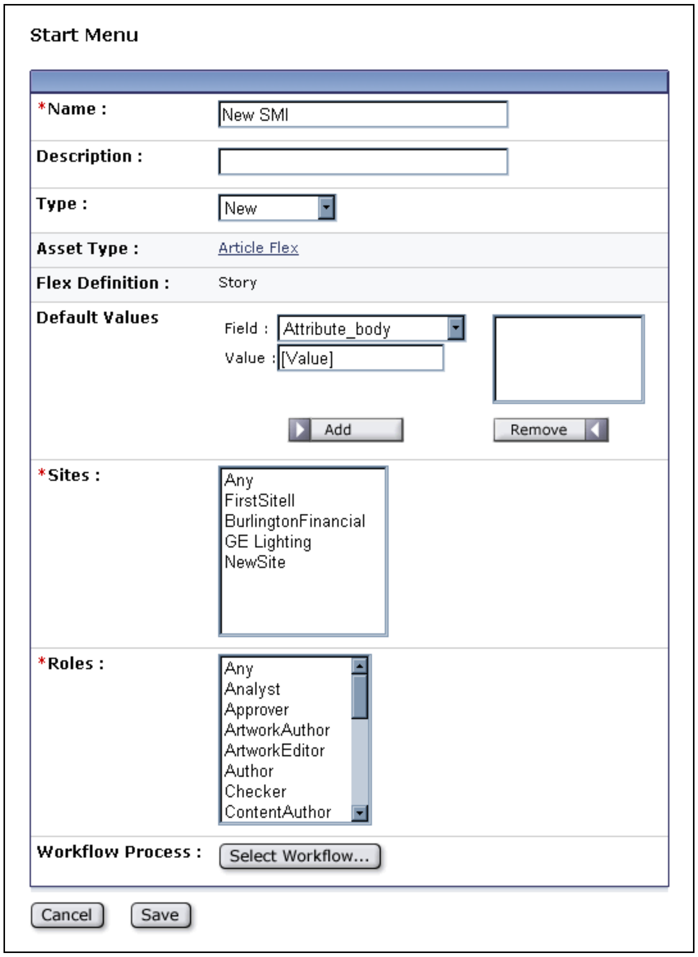
*Asset Type : Article Flex

*Flex Definition : Any

a. In the **Flex Definition** field, select the asset subtype.

b. Click **Continue**.

4. In the next “Start Menu” form, fill in the fields as explained below:



Start Menu

*Name : New SMI

Description :

Type : New

Asset Type : Article Flex

Flex Definition : Story

Default Values

Field : Attribute_body

Value : [Value]

*Sites :

*Roles :

Workflow Process :

a. In the **Description** field, enter a short description, using up to 255 characters.

b. In the **Type** field, select **New**.

c. Note that the **Asset Type** field is an information-only field that displays your choice of asset type.

d. (Optional). In the **Default Values** section, complete the following steps if you want field values to be set automatically when a content provider creates an asset with this start menu item.

1) Select a field from the **Field** drop-down list. This list displays all the fields for assets of this type; for example, category, source, and template.

2) In the **Value** field, enter or select a value.

> **Notes**
>
> - Selecting certain default fields, such as category and source, automatically produces a drop-down list of options in the **Value** field. For other fields, you must provide your own value. **All values are case-sensitive**.
> - Content Server does not validate the value against the field that you are setting it for. Therefore, you must be sure that you have entered an appropriate value.

3) Click the **Add** arrow to add the value to the field.

4) Repeat the previous three steps, as necessary.

> **Note**
>
> When you set a field value through a start menu item, the field does not become a read-only field. That is, no matter what value you set in the "Start Menu" form, that value can be overridden by the content provider who creates an asset through the start menu item. The values that you specify are for convenience only.

e. In the **Sites** field, select which sites can use this start menu item. Use **Ctrl-click** to select more than one site.

f. In the **Roles** field, select the roles that can have access to the start menu item. Use **Ctrl-click** to select more than one role.

> **Note**
>
> If you are assigning a default workflow with this start menu item, the roles that you select in this step must match the roles that are (or will be) authorized for the start step in the workflow process.

g. (Optional) To configure workflow process details for assets created with this start menu item, complete the following steps:

1) Click **Select Workflow**.

2) In the “Select Workflow” form, select the appropriate workflow process from the drop-down list and then click **Select Workflow**.

3) In the “Set Participants” form, select at least one user for each role that appears, and click **Set Participants**.

4) Click **Continue**.

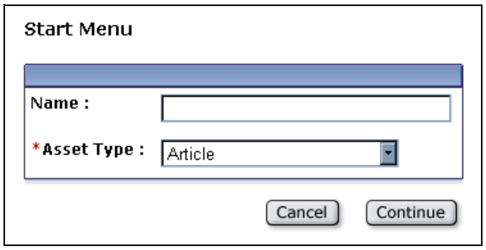
The system saves the workflow information and redisplays the “Start Menu” form.

5. Click **Save**.

Creating “Search” Start Menu Items

To create a “Search” start menu item

1. In the **Admin** tab, expand **Start Menu**, then double-click **Add New**.
2. In the “Start Menu” form, do the following:



Start Menu

Name :

*Asset Type : Article

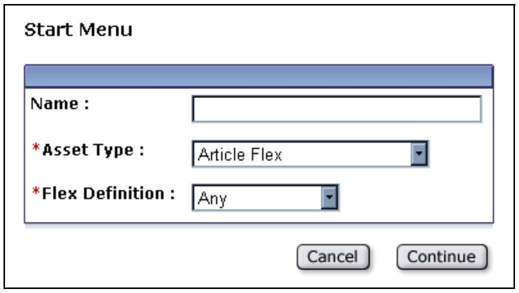
a. In the **Name** field, enter the name of the item, using up to 64 characters. Although it is not required, it is a good idea to start the name with the word “Find” (for example, “Find Article” or “Find Template”). This is the name that is displayed on the **Search** list.

b. In the “Asset Type” field, select the type of asset for which you are creating the Start Menu item.

c. Click **Continue**.

3. If you have chosen a flex asset type, continue with this step. Otherwise skip to step 4.

If you have chosen to a create a start menu item for a flex asset type, you are presented with the following form:



Start Menu

Name :

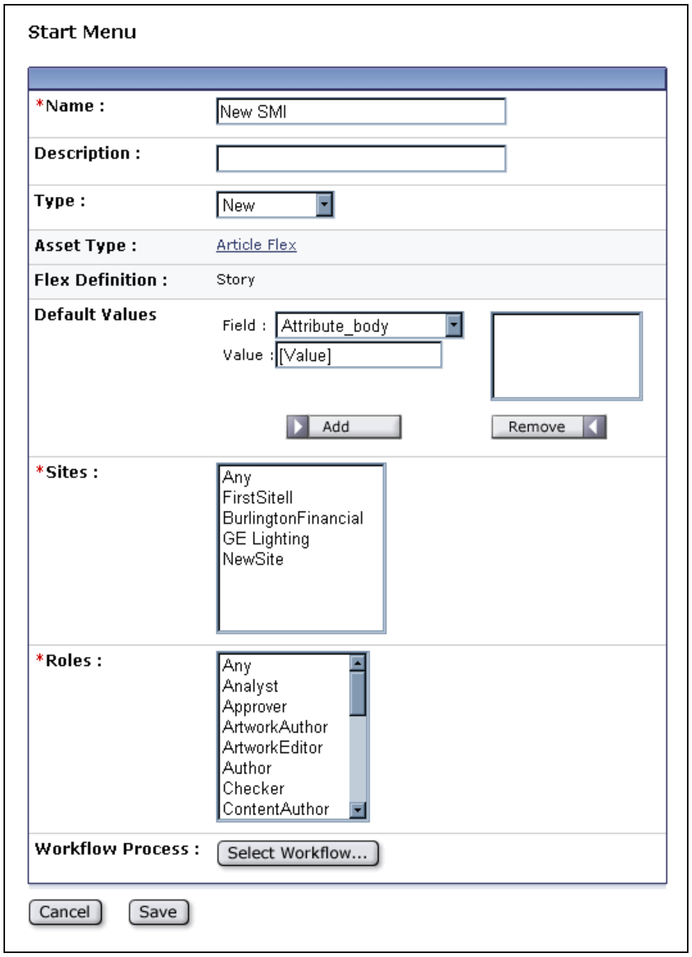
*Asset Type : Article Flex

*Flex Definition : Any

a. In the **Flex Definition** field, select the asset subtype.

b. Click **Continue**.

4. In the next “Start Menu” form, fill in the fields as explained below:



Start Menu

*Name :	New SMI
Description :	
Type :	New
Asset Type :	Article Flex
Flex Definition :	Story
Default Values	Field : Attribute_body Value : [Value]
*Sites :	Any FirstSiteII BurlingtonFinancial GE Lighting NewSite
*Roles :	Any Analyst Approver ArtworkAuthor ArtworkEditor Author Checker ContentAuthor
Workflow Process :	Select Workflow...

5. In the **Description** field, enter a short description for the start menu item.
6. In the **Type** field, select **Search**.
7. In the **Sites** field, select which sites can use this start menu item. Use **Ctrl-click** to select more than one site.
8. In the **Roles** field, select all the roles that can have access to the start menu item. Use **Ctrl-click** to select more than one role.
9. Click **Save**.

Creating Start Menu Items for CS-Desktop

For information about creating CS-Desktop start menu items, see “Configuring the Word Asset Types for CS-Desktop,” on page 312.

Creating Start Menu Items for CS-DocLink

For information about creating CS-DocLink start menu items, see “Configuring the CS-DocLink Asset Types,” on page 319.

Displaying a List of All Start Menu Items

In addition to the individual start menu nodes that appear for each of the asset types that exist on your CS system, there is a top-level **Start Menu** node in the **Admin** tab. Use this node when you want to see a list of all the start menu items in the system.

Additionally, there is a link to the list of all the start menu items from the “Inspect” form for any individual start menu item.

Setting Access Permissions for Assets

Content Server provides a system by which you can set access permissions for assets that are not part of a workflow process. For such assets, you can create policies and, within them, define which roles have permissions to perform which functions on assets. For example, a policy for the function “copy” might specify which roles are allowed to copy an asset. A different policy for “copy” might specify which roles are denied the ability to copy an asset. For flex assets, inheritable policies can be set on parents, meaning that the policies will be inherited by the children.

Functions that can be performed on assets, by default, are the following: checkout, copy, edit, delete, rollback, share, approve, and build. (For definitions of the functions, see “Determine the Function Privileges,” on page 148).

All of the above-listed functions are defined in the `futuretense_xcel.ini` file, as values of the property `xcelerate.authorizefunction` (located in the **Authorization** tab, if you are using the Property Editor). You can specify additional functions (which are listed in the **Authorization** tab).

To summarize, access permissions serve a three-fold purpose:

- Extend the workflow function privileges system, making it possible to set permissions for assets when they are not in workflow
- Introduce new functions (authorize, inspect, and preview) for viewing the contents of assets that are not in workflow
- Introduce a hierarchical permission inheritance model for assets that are not in workflow.

Note

The “access permissions” feature does not apply to other entities, such as pagelets.

Permission Structure

Permission structure determines which functions (for example, edit and view) a role can perform. A user's permissions are managed by assigning him the roles that can perform the functions. Content Server implements a three-level permission structure: “Grant,” “Deny,” and “Inherited.” “Grant” and “Deny” denote the explicit permissions, that is, grant or deny the permission to perform a function. “Inherited” means that permissions are neither granted nor denied. Instead, the permissions are inherited from the parent asset’s policy, if one exists. If the parent’s policy does not exist, then the asset’s permissions are inherited from the policy of the parent’s parent, and so on.

If permissions are ambiguous, the following considerations may help resolve the ambiguity:

- A permission is granted if at least one of the user's roles grants the permission.
- A permission is denied if at least one of the user's roles denies the permission, and none of the user's roles grant the permission.
- "Grant" overrides "Deny."

  If one role grants a permission, but another role denies the same permission, the permission is granted.

  The preceding scenarios are summarized in the following table:

	Role 1	Role 2	Result
Permission	Granted		Granted
	Denied		Denied
	Granted	Denied	Granted

- Assets can inherit permissions from parent assets.

  For example, if a parent asset named "Money Market Funds" grants the "delete" permission to a role named `Editor`, and user Bob has that role, then Bob can delete the "Money Market Funds" asset and all of its child assets.
- Asset permissions can override the parent assets' permissions.

  A "Prime Money Market semi-annual report" asset denies the "delete" permission to a certain role. That same role is granted the "delete" permission in the "Prime Money Market" parent asset. This means that Bob cannot delete the report, even though the parent asset grants the "delete" permission.
- If there are no applicable permissions found on a particular asset, then the administrator needs to examine the `futuretense_xcel.ini` file for this permission. In the `futuretense_xcel.ini` file, permissions for a particular function are set for all assets, rather than for a single asset.

  For more information about setting permissions in the `futuretense_xcel.ini`, see "To set access permissions from futuretense_xcel.ini," on page 119.
- Asset permissions overrides `futuretense_xcel.ini` settings.

  For example, if there is a permission `xcelerate.copy.grant` in the `futuretense_xcel.ini` file and the permission to delete asset A is revoked, then the permission to delete asset A is denied.

Types of Authorization

Three types of authorization can be defined on a particular function and a particular asset: workflow function privileges, access permissions, and authorization properties in `futuretense_xcel.ini`, all independent of each other. (For information about workflow function privileges, see Chapter 9, "Creating and Managing Workflow Processes.")

When a user attempts to perform a function on an asset, the system performs a series of authorization checks in the following order, for that function:

For assets in workflow:

1. The system checks the workflow function privileges for the function in the workflow process in which the asset is entered.
2. If no function privilege is defined for the function, the system checks the access permission policy for that function, as defined in the asset's access permissions interface.
3. If no access permission is set for the function (including inheritance of the parent's policy) as defined in step 2, then the system checks the authorization properties in `futuretense_xcel.ini`.
4. If none of the above types of authorization are defined, permission to perform the function on the asset is granted, by default.

Note

The above types of authorization are checked in the order shown. If two or more types of authorization are defined, the authorization that is found first overrides all other types of authorization. For example, for a specific function such as Edit, if both workflow function privileges and access permissions are defined on an asset, then function privileges on Edit override access permissions on Edit.

For assets that are not in workflow:

1. The system checks the access permissions policy for the function, as defined in the asset's access permissions interface.
2. If no access permission is set for the function (including inheritance of the parent's policy), then the system checks the authorization properties in `futuretense_xcel.ini`.
3. If none of the above levels of authorization are defined, permission to perform the function on the asset is granted, by default.

Summary

The different types of authorization described above are independent of each other, which means that only one type of authorization can apply to an asset at any given time for any given function. The following guidelines can help you decide which type of authorization to use:

1. If a function does not require authorization, do not use any of the authorization types. All roles will have permission to perform that function.
2. If you need a simple authorization policy across multiple assets, set authorization properties in `futuretense_xcel.ini`.
3. If you need a simple authorization policy on a particular asset, set access permissions.
4. If you need a more complex type of authorization that allows workflow to control the permissions on an asset, set workflow function privileges.

Setting Access Permissions

Setting access permissions means creating a policy for an asset by specifying which roles can perform which function(s) on the asset.

There are several ways to set access permissions on assets:

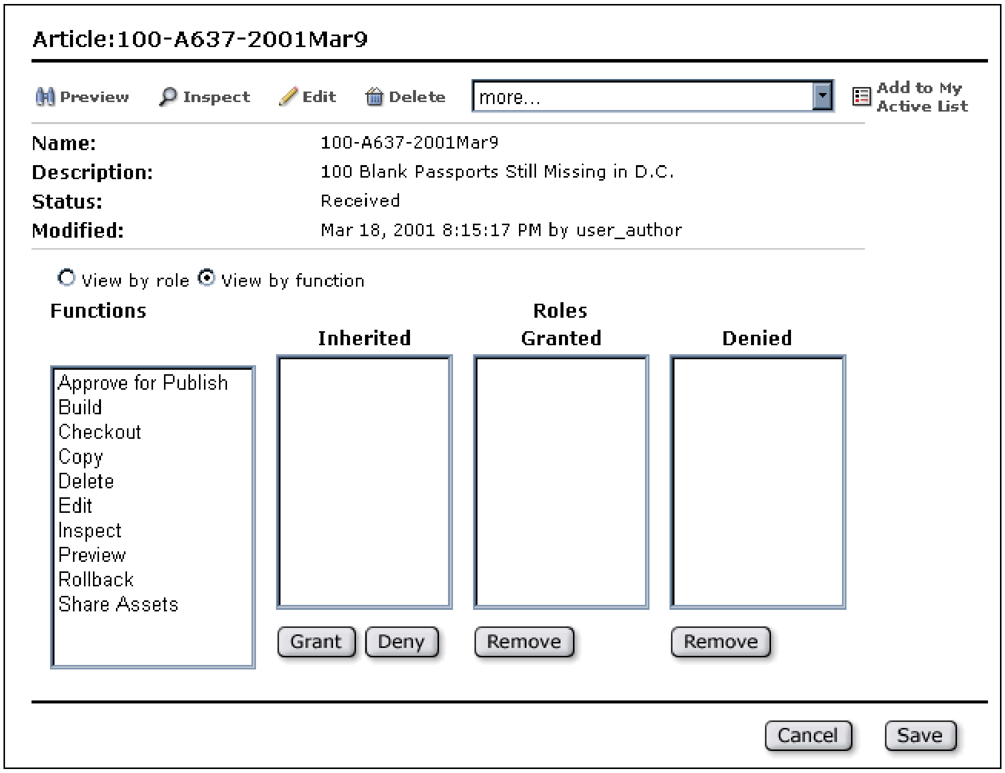
- Using the Advanced interface, to set permissions on individual assets
- Using the `futuretense_xcel.ini` file, to set permissions on all assets at once

> **Note**
>
> If your asset is entered in to a workflow for which function privileges are defined, the function privileges override the asset's access permissions. For information about function privileges, Chapter 9, "Creating and Managing Workflow Processes."

Setting Access Permissions from the Advanced Interface

To set access permissions from the Advanced interface

1. Locate the asset for which you will set access permissions.
2. In the asset's "Inspect" form, select **Access Permissions** from the drop-down menu list in the action bar.
3. Select one of the radio boxes, click **View by role** or **View by function**. In this example, we use **View by Function**.



Article:100-A637-2001Mar9

Name:	100-A637-2001Mar9
Description:	100 Blank Passports Still Missing in D.C.
Status:	Received
Modified:	Mar 18, 2001 8:15:17 PM by user_author

○ View by role ● View by function

Functions

Approve for Publish
Build
Checkout
Copy
Delete
Edit
Inspect
Preview
Rollback
Share Assets

> **Note**
>
> In this procedure, we are using **View by function**, which allows you to select a function and then specify the roles that are granted or denied permission to perform it. If you select **View by role**, the order will be reversed: you will have to select the role first, then specify the functions it is granted or denied.

4. Select a function.

Roles that can perform the function are now displayed in the “Inherited” list box, by default. “Inherited” refers to the permissions policy that the asset inherits from its parent, or its parent’s parent, and so on, depending on which asset in the hierarchy has a policy.

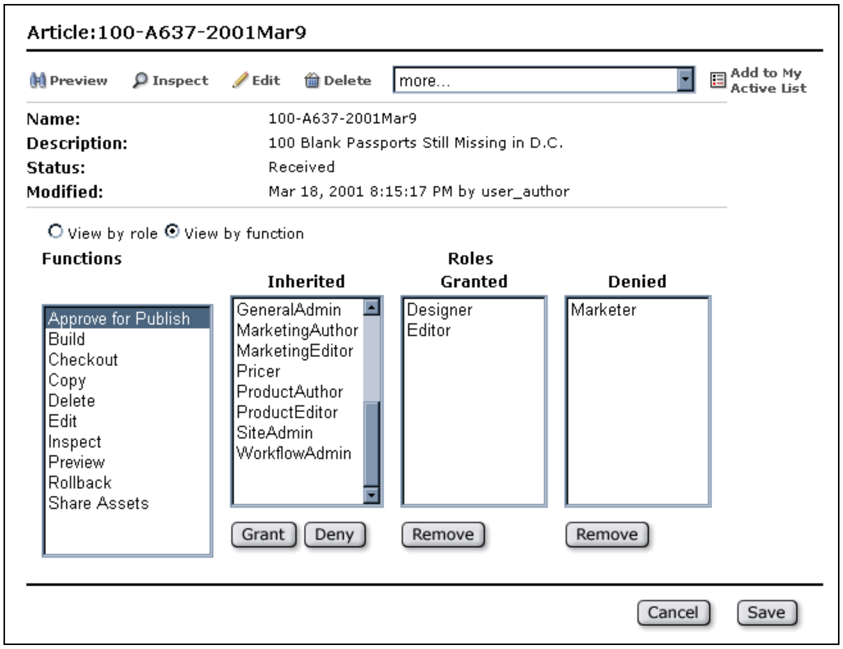
5. Select a role (or roles) to which you will deny or grant the permission to perform the function. (To select a block of roles, **Shift-click** the extremes of the block. To select non-adjacent roles, **Ctrl-click** each role).

6. Select either **Grant** or **Deny** to allow or forbid the role(s) to perform the function:

- **Grant** explicitly allows the selected role(s) to perform the selected functions on the specific asset.
- **Deny** explicitly forbids the selected role(s) to perform the selected function on the specific asset.
- Selecting neither **Grant** nor **Deny** leaves the roles in the “Inherited” list box to indicate that access permissions are inherited.

7. Repeat steps 4–6 for each function and role that must be granted or denied or which you want to grant or deny permission.

When you have completed selecting functions and roles, your access permissions screen will look similar to the one below:



Article:100-A637-2001Mar9

Name:	100-A637-2001Mar9
Description:	100 Blank Passports Still Missing in D.C.
Status:	Received
Modified:	Mar 18, 2001 8:15:17 PM by user_author

○ View by role ● View by function

Functions	Roles: Inherited	Roles: Granted	Roles: Denied
Approve for Publish Build Checkout Copy Delete Edit Inspect Preview Rollback Share Assets	GeneralAdmin MarketingAuthor MarketingEditor Pricer ProductAuthor ProductEditor SiteAdmin WorkflowAdmin	Designer Editor	Marketer

8. Click **Save** when finished.

   Users who now access this asset will be able to perform the functions that their roles allow.

Setting Access Permissions from Property Files

To set access permissions from `futuretense_xcel.ini`

In `futuretense_xcel.ini`, either grant or deny permissions by setting the following properties:

- `xcelerate.grant.`*`functionname`* `=` *`rolelist`*
- `xcelerate.deny.`*`functionname`* `=` *`rolelist`*

  where

  `functionname` is the name of the function, as shown in the `futuretense_xcel.ini` file, on the “Authorization” tab if you are using the Property Editor.

  `rolelist` is a comma-separated list of roles for which the permission is either denied or granted.

If the permission is granted or denied in the `futuretense_xcel.ini`, it is granted or denied for all assets. These permissions can be overridden by specifically setting access permissions on an asset-by-asset basis, using Content Server’s access permissions feature.

Configuring Other Options for Asset Types

In the **Admin** tab, there are several other items that configure asset types. These items contribute to the design and implementation of the asset types, which means that they are typically used by the developers. Administrators typically do not manage the items.

However, after an asset type is designed and installed, you might be asked to maintain the following items:

- sources
- categories
- subtypes
- associations
- mimetypes

For information about these items and procedures for creating and editing them, see the section called “Data Design” in the *Content Server Developer’s Guide*.

Managing Access to the Tree (Advanced interface only)

You have several ways to configure the tree that is displayed in the left-hand panel of the Advanced interface:

- You can configure the tree to be either displayed by default in the main window or hidden by default.
- You can control whether users who toggle a tree off are allowed to toggle the tree back on.
- You can configure tabs within the tree, adding items or functionality to them, and determining which users have access to them based on their roles.
- You can configure the tabs' nodes to specify the number of items that are displayed under any given node.

Content Server supports two general categories of tree tabs:

- System default tabs, which provide administrative functionality (**Admin**, **Site Admin**, **Workflow**) and access to individual assets (**History**, **Active List**, **Site Plan**).
- Custom tabs
  - Tabs that you create to make it more convenient for users in different roles to create the most commonly used asset types.
  - Tabs that developers create to offer customized behavior or new functionality.

Note

The **Workflow Groups** tab is a hybrid between a system default and a custom tab. Although the **Workflow Groups** tab is not installed by default and you must create it to use it, the element that provides its logic is installed. That is, while you must manually create this tab, your developers do not have to code a new element for you.

This section describes how to add roles to the system default tabs, how to create basic tree tabs that provide access to asset types, and how to configure whether the tree is displayed by default.

For information about creating custom tree tabs that implement new functionality, see the section called "Management System Features" in the *Content Server Developer's Guide*.

Displaying and Hiding the Tree

Any user can toggle the tree on and off whenever he needs less or more space in the main window to display the assets that he is working on.

If you want, you can configure your CS system so that the tree is toggled off by default when users first log in to the system. You can also control whether users are allowed to toggle the tree back on.

The Content Server features for working with assets are available in right-mouse menus on the tabs as well as through the icons at the top of the window and the action bars displayed in the **New**, **Edit**, and **Inspect** forms for individual assets. Therefore, it is not necessary to display the tree for content providers. However, because the administrative functions are available only through the **Admin**, **Site Plan**, and **Workflow** tabs, administrative users must always have access to the tree.

To configure display of the tree and toggling rights

1. Start the Property Editor and open the `futuretense_xcel.ini` file.
2. Select the **Preference** tab.
3. To specify that the tree should be toggled off by default when users first log in to the Advanced interface, set the value of the `xcelerate.showSiteTree` value to `false`.
4. (Optional) To configure the system so that only administrative users (that is, users who have the xceladmin ACL assigned to their user accounts) are allowed to toggle the tree back on, set the `xcelerate.restrictSiteTree` property to `true`.
5. Save the property file and close the Property Editor.
6. Restart the application server.

Creating Tree Tabs

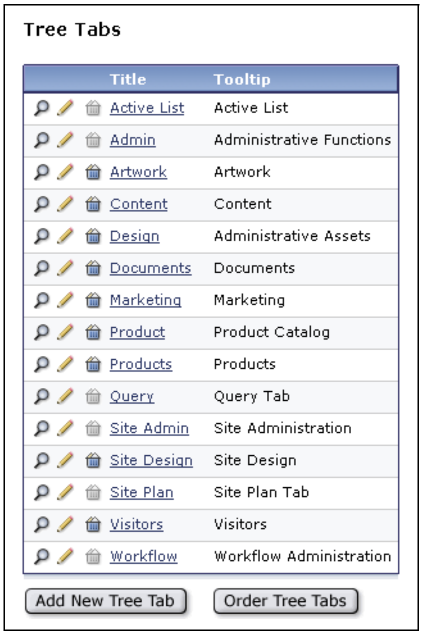
This section shows you how to create new tree tabs, and the **Workflow Groups** tab.

Creating a New Tree Tab

To create a new tree tab

1. In the **Admin tab,** double-click **Tree**.

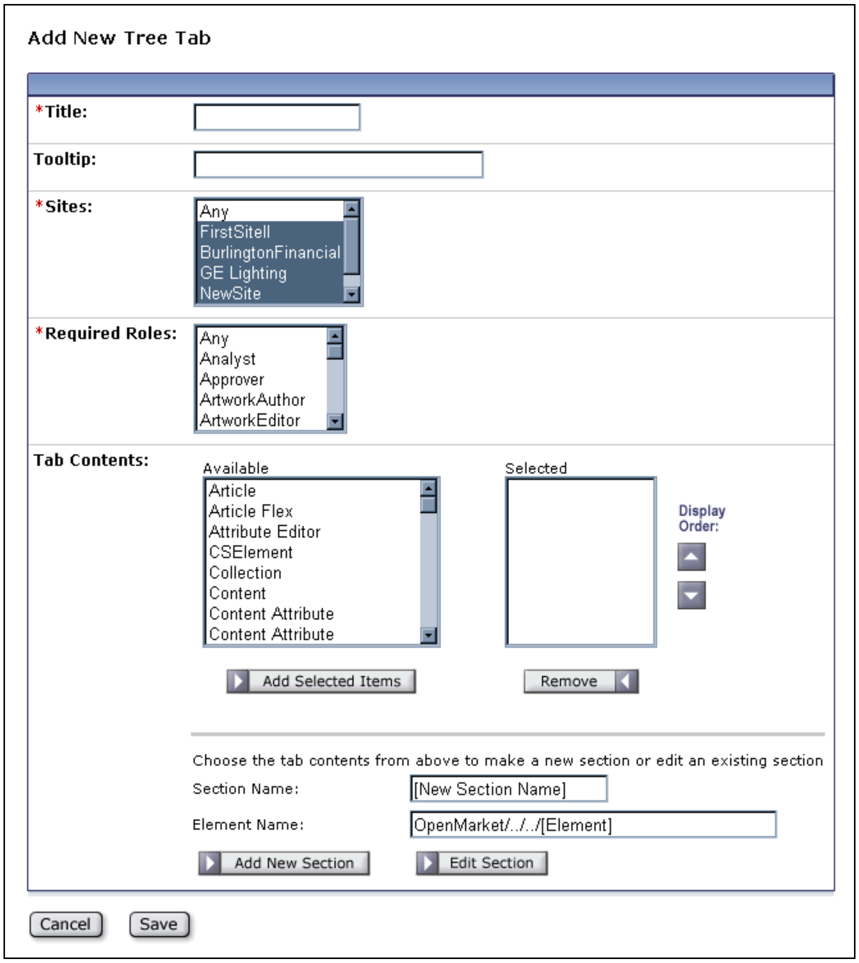
   Content Server displays the “Tree Tabs” form.



Tree Tabs

Title	Tooltip
Active List	Active List
Admin	Administrative Functions
Artwork	Artwork
Content	Content
Design	Administrative Assets
Documents	Documents
Marketing	Marketing
Product	Product Catalog
Products	Products
Query	Query Tab
Site Admin	Site Administration
Site Design	Site Design
Site Plan	Site Plan Tab
Visitors	Visitors
Workflow	Workflow Administration

2. In the form, click **Add New Tree Tab**.



Add New Tree Tab

*Title:

Tooltip:

*Sites: Any, FirstSiteII, BurlingtonFinancial, GE Lighting, NewSite

*Required Roles: Any, Analyst, Approver, ArtworkAuthor, ArtworkEditor

Tab Contents:

Available: Article, Article Flex, Attribute Editor, CSElement, Collection, Content, Content Attribute, Content Attribute

Selected

Choose the tab contents from above to make a new section or edit an existing section

Section Name: [New Section Name]

Element Name: OpenMarket/../../[Element]

a. In the **Title** field, enter a short, descriptive name of up to 64 characters.

b. In the **Tooltip** field, enter a short, informative description of the tab, up to 255 characters. This description is displayed when the mouse hovers over the tab.

c. In the **Site** field, select the sites that will display the tab.

d. In the **Required Roles** field, select which roles can access the tab.

e. In the **Tab Contents** field, select the asset types that will be displayed on the tab and then click **Add Selected Items**.

f. (Optional) If you want to add custom functionality to this tab, use the **Section Name** and **Element Name** fields at the bottom of the form. You need help from your developers for this step. See the *Content Server Developer's Guide* for more information.

3. Click **Save**.

The new tab is displayed in the tree (refresh the display, if necessary).

Creating the Workflow Groups Tab

The **Workflow Groups** tab displays the workflow groups that exist for a site and the assets that are included in those groups. For information about workflow groups, see "Workflow Groups," on page 132. This tab does not appear by default. If you want to use it, you must create it.

To create the Workflow Groups tab

1. In the **Admin** tab, double-click **Tree**.

   Content Server displays the "Tree Tabs" form.
2. In the form, click **Add New Tree Tab**.
3. In the **Title** field, enter a short, descriptive name of up to 64 characters.

   For example: **Groups**.
4. In the **Tooltip** field, enter a short, informative description of the tab, up to 255 characters.

   For example: **Workflow Groups**.
5. In the **Site** field, select the sites that will display the tab.
6. In the **Required Roles** field, select which roles can access the tab.
7. In the **Tab Contents** area, scroll down to the section area. In the **Section Name** field, enter the following name exactly as it appears here: `Groups`.
8. In the **Element Name** field, enter the following name exactly as it appears here: `OpenMarket/Gator/UIFramework/LoadWorkflowGroups`
9. Click **Add New Section**.

   The word "`Groups`" appears in the **Selected** column.
10. Click **Save**.

Editing Tree Tabs

After you create roles for your sites, you must edit the following system default tabs to add your roles, so that your users have access to the tabs:

- Site Plan
- History
- Active List

To edit a tree tab

1. In the **Admin** tab, double-click **Tree**.
2. In the "Tree Tabs" form, navigate to the desired tab and click its **Edit** icon.
3. In the "Edit Tree Tab" form, make the necessary changes. For example, select your roles from the **Roles** list.
4. Click **Save**.

Be sure to add your roles to all of the default system tabs so that your users have access.

Deleting Tree Tabs

To delete a tree tab

1. In the **Admin** tab, double-click **Tree**.
2. In the “Tree Tabs” form, navigate to the desired tree tab and click its **Delete** icon.

   Content Server displays a warning message.
3. Click **Delete Tree Tab**.

   The tree tab has been deleted.

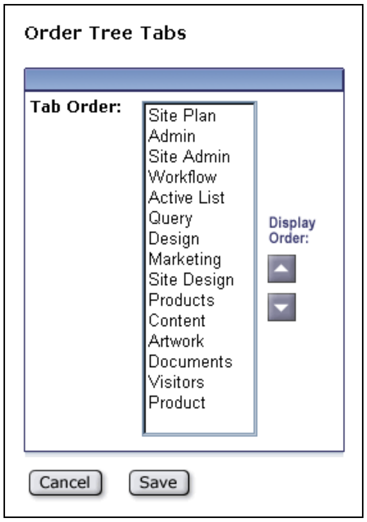
Changing Tree Tab Order

You can change the order in which the tree tabs are displayed.

To change the order of tree tabs

1. In the **Admin** tab, double-click **Tree**.
2. At the bottom of the “Tree Tabs” form, click **Order Tree Tabs**.

   Content Server displays the “Order Tree Tabs” form.



Order Tree Tabs

Tab Order:

Site Plan
Admin
Site Admin
Workflow
Active List
Query
Design
Marketing
Site Design
Products
Content
Artwork
Documents
Visitors
Product

   The form lists the tabs in the order in which they will appear in the tree.

   The tree arranges its tabs into rows. When there are more tabs than a single row can accommodate, an additional row is formed behind it. The tabs in each row are ordered from left to right. Rows are ordered front-to-back.
3. To move a tab in the tree, select the tab from the list and then use the arrow buttons to move the tab to the correct position. Repeat this step for each tab that you want to reposition.
4. Click **Save**.

Configuring the Number of Items Displayed Under Nodes on Tabs

To optimize Content Server's response time, you may want to limit the number of asset names that are displayed under their respective node in a tree tab. Specifically, when the Content Server database holds many assets of any one type, Content Server can take a long time to load all the assets and then display the asset names under their node in the tab. To minimize the loading time, you can limit the number of items to be displayed under the node by setting a threshold. Then, when the number of assets or other items for a node exceeds the threshold, and a user attempts to access the assets, the Advanced interface prompts the user to enter search criteria that restricts the number of items.

For example, suppose that a node contains more than 300 articles, and the display threshold is set to 100. When a user expands the node icon that represents the articles, Content Server displays a small search criteria form that prompts the user to enter criteria that Content Server uses to limit the number of article assets that it returns.

To configure the number of items under a node in a tree tab

1. Start the Property Editor and open the `futuretense_xcel.ini` file.
2. Select the **Preference** tab.
3. Select the `xcelerate.treeMaxNodes` property and specify a value for it. By default, this property is set to 100, which means that up to 100 items can be displayed under a node.
4. Save and close the property file.
5. Restart the application server.

Chapter 9

Creating and Managing Workflow Processes

Workflow is a feature provided by Content Server that you use to manage the work on an asset when more than one person participates in its creation. For example, if assets of a certain type must be reviewed by an editor or a legal representative before it can be approved for publishing, the workflow feature can route those assets to the appropriate people at the appropriate time.

This chapter contains the following sections:

Overview

The end goal for any content provider is to have content assets published. Before an asset can be published, it must be approved for publishing. The workflow feature routes assets through whatever series of tasks you deem necessary in a **workflow process** that ushers assets from creation to approval.

You can configure a workflow process with as many or as few tasks as necessary to reflect the way the work at your organization is accomplished. You can configure e-mail messages that Content Server sends to notify people when assets are assigned to them and to remind them that a deadline is approaching or has been missed.

Because there are so many configuration possibilities, it is typical to create a separate workflow process for each asset type that you plan to use workflow with rather than attempt to create one process for more than one asset type.

Workflow Participants

When you begin creating a workflow process, the first general question is this: what are the job titles of the people who work on assets of this type? For example, are they authors, editors, marketers, graphic artists, product managers, lawyers?

The job titles of the people who participate in a workflow process are considered **roles** in the Content Server interfaces. Roles describe the function of an individual on a site. When you enable a user for a site, you assign that user the roles that he fulfills for that site.

When you create a workflow process, you determine which roles are appropriate for each task. Then, when an individual asset is going through that workflow process, only the users who have the appropriate role are allowed to complete the task. The individual user who is selected from the pool of users who have the correct role at any point is called a workflow **participant**.

Workflow States

Next, what are the tasks that are performed for assets of this type? For example, writing, pricing, editing, fact checking, legal review, and so on.

These tasks are called workflow **states**. A state is a point in the workflow process that represents the status of the asset at that point. For example, writing article, reviewing image, legal review, and so on. Participants complete the work that the state represents while the asset is in that state.

An asset that a participant is working on (or is supposed to be working on) is called an **assignment**. A user’s **assignment list** is displayed In the “My Assignments” form in the Advanced interface. An asset appears on a user’s assignment list as soon as it enters a state for which the user has a role to fulfill as a workflow participant.

Should an asset in a specific state remain in that state for only a specific amount of time? If so, assign a deadline to the state. You can then configure the workflow process to send e-mail messages that remind a participant when the deadline is approaching or has been missed. These e-mail messages are examples of **timed actions**.

You can create one or more timed action for each state.

Workflow Steps: Moving Assets from State to State

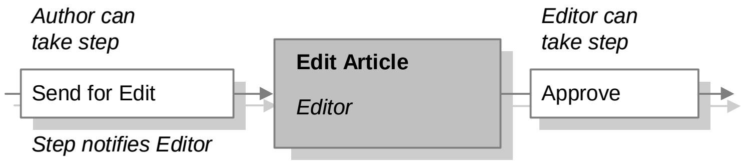
Next, how does the asset move from state to state? Does a marketing writer send a prospectus asset to a legal reviewer? Does a graphic artist send an image asset to an editor? And then what?

The movement of an asset between states is called a **step**. Because creating the steps in a process links together states in a specific order, creating the steps in your workflow process is what organizes your process.

When you create a step in a process, you specify which state a step moves the asset from (the **From State**), which state that step moves the asset to (the **To State**), and which roles can take the step.

How Steps Work

A step places an asset in a state and notifies participants from the appropriate roles. This operation creates the assignment. You then need a step for the participants from those roles to take that moves the asset to the next state. In other words, the roles notified by the previous step should be the roles authorized to take the next step. For example:



How does a user take a step in a workflow process? By specifying that he or she has finished the assignment (except for the start step, which is described below). When a user selects the **Finish My Assignment** option for an asset that is assigned to him or her, that option invokes the step, moves the asset to the next state, and assigns the asset to the next participants.

When more than one participant is assigned an asset in the same state, using the **Finish My Assignment** option is also referred to as **voting**. Each participant “votes” to move the asset to the next state.

The first step in a workflow process is a **start step**. A start step is one which has no From State. That is, the start step begins the workflow process, moving the asset to the first state in the series of states. This is the step that is invoked when the workflow process is assigned to the asset. Only users who have the roles authorized for the start step can start the workflow process for an asset.

Step Actions and Conditions

Step actions are events that occur when the step is completed. For example, when a step occurs, the asset is assigned to a participant. Should the participant be notified that an asset has been assigned to him or her? If so, you can configure a step action to send an e-mail message to the new participant. You can specify one or more step actions for each step.

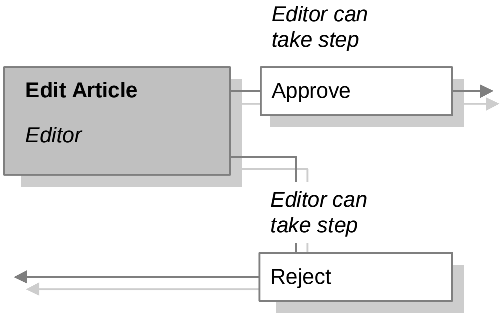
Another example of a step action is the **ApproveForPublish** action. It is a default action delivered with Content Server that approves the asset. Typically you use this action in the final step of a workflow process.

Finally, are there any requirements that must be met for an asset in a state before the step can move the asset to the next state? If so, configure and assign a **step condition** to the step. For example, you could configure a condition that verifies that the asset has all of its association fields filled—that an article has the associated images that it needs—before it progresses to the next state.

Multiple Paths for the Asset

When a workflow process includes a state in which people are reviewing an asset, typically there is more than one path possible for an asset in that workflow because the reviewer can either accept it the way it is or reject it.

In such a case, you create two steps for moving the asset from the review state. When the asset is in that state, the "Finish My Assignment" form lists both options and the participants select the appropriate step when they finish their assignments. For example:



What happens if more than one participant are reviewing the same asset in the same state and they choose different steps (that is, vote differently)? That depends on how you configured each step. There are several possibilities:

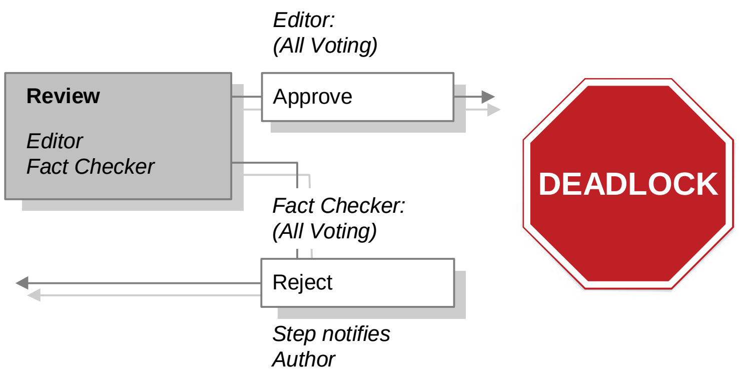
- Configure the steps so that the first participant to finish the assignment (vote) determines the direction the asset takes.
- Configure the steps so that all participants have to select the same step (vote the same way) in order for the asset to progress in either direction. A step that all the participants must select before that step can be completed is called an **all-voting step**. Note that this option can result in **deadlocks**, described in the next section.
- Use a combination of the preceding possibilities: configure one all-voting step (all participants must agree), and configure the other step(s) so that if any participant selects it, the step completes and the asset takes that path.

Managing Deadlocks

A deadlock occurs when the following conditions are true:

- There is more than one step from a state.
- Two or more of the steps require the participants to perform that step. That is, they are all-voting steps.
- An asset in that state is assigned to more than one participant.
- The participants select different all-voting steps when they finish the assignment.

This diagram illustrates a deadlock:



Note that if even one of the steps is not all-voting and a participant selects that step, the asset will not become deadlocked.

Resolving Deadlocks

An asset that is in a deadlocked state cannot progress through the workflow process until the deadlock is resolved. To resolve the deadlock, the participants must confer with each other and agree on that path that the asset should take. Then, the participants who must change their selection can do one of the following to resolve the deadlock:

- Finish the assignment and select the step that they all agreed to take.
- Select the **Abstain from Voting** option from the **Workflow Commands** drop-down list on the asset’s “Status” form.

Preventing Deadlocks

Before configuring a workflow process that can result in a deadlock, be sure that it is absolutely necessary to have complete agreement on all the possible steps from the state. As you can see from this description, deadlocks cause additional work for all the participants so be sure that you use this feature only when you need to.

For example, consider a review state with two possible steps: “return for revisions” and “approve for publish.” If you configure the steps so that “return for revisions” does not require a unanimous vote but “approve for publish” does, you have created a desirable control—all the reviewers must agree before the asset can be published and any rejection stops the asset from being published—without risking a deadlock.

Notifying Participants When Deadlocks Occur

If you do need to create a workflow process that can result in a deadlock, be sure to configure and assign a **deadlock action** that notifies the participants of the deadlock for each of the steps that can cause the deadlock. The default deadlock action is an e-mail message that Content Server sends to the appropriate participants when a step causes a deadlock.

Workflow Groups

Is there ever a situation in which several assets are so closely connected that they need to be thought of as one unit of work or they need to be approved at the same time? In such a case, you can use the **workflow group** feature.

Using Workflow Groups

Workflow groups enable content providers to send a defined set of assets though the workflow together. While it is the content providers who create workflow groups and select the assets that are assigned to the group, you, the administrator, still need to know what kind of assets will be included in workflow groups. Why? So that you can configure the workflow processes appropriately.

For example, you can configure workflow steps that allow each asset in the group to progress to the next state when it is finished or you can configure a step that requires all the assets in the group to reach that point before any of them can progress. (This second example, called a **synchronize step**, is described next.)

Adding a Synchronize Step

When creating a workflow process that will be used with workflow groups, it is usually best to configure the process so that it has only one synchronize step. Multiple synchronize steps can slow down the work on those assets unnecessarily. Assess the business process that is reflected by the workflow process and determine which steps must truly be synchronized: perhaps all the assets should go to legal review in one batch, or perhaps all the assets should be approved for publishing at the same time, for example.

Managing Group Deadlocks

If you create a workflow process to be used with workflow groups and any of the steps can result in a deadlock, be sure to configure and assign a **group deadlock action** that notifies participants when there is a group deadlock and assign it to the process.

Delegating and Clearing Assignments

People go on vacation, get reassigned to new work groups, and move on to different jobs. What happens to the assets that they are working on? They can **delegate** their assignments to other participants who have the appropriate roles.

Additionally, each workflow process can have a **workflow administrator**. The administrator of a workflow process can delegate assignments on behalf of the other participants.

When an asset is delegated to a new participant, should that person receive an e-mail notice? If so, configure a **delegate action** to send an e-mail message to new assignees when assets are delegated to them. You can specify one or more delegate actions for each workflow process.

If an asset no longer needs to be assigned or if it is easiest to clear the assignment and then start over, you can use the **Clear Assignments** feature in the **Admin** tab.

Placing an Asset in Workflow

An asset begins its participation in a workflow process in one of the following ways:

- A user selects a workflow process from the **Workflow Commands** field on the "Status" form for the asset.

  Selecting the workflow process invokes the start step for the process, which places the asset into the first state.

- A user creates an asset and the start menu **New** item for assets of that type is configured such that there is a default workflow process.

  In this case, saving the asset invokes the start step for the process, which automatically places the asset into the first state.

Because steps are enabled for specific roles, only the users who have a role that is assigned to the start step of a process can select that process. This means that if you are using start menu items to place assets in workflow, you must be sure that the roles assigned to the start menu item are the same roles that are assigned to the start step of the default workflow.

Note

Versions of Content Server prior to Version 4 assigned a default workflow to assets through a property in the `futuretense_xcel.ini` file. The ability to assign a default workflow through a start menu item has replaced that method of determining a default workflow.

Restricting Access to Assets While They Are in Workflow

Although workflow routes an asset through a business process, sending it to the appropriate users at the appropriate times, the fact that the asset is assigned to a specific user does not stop other users from modifying or even deleting that asset.

If you want to restrict who has access to an asset while it is in workflow, use the workflow feature called **function privileges**. These are restrictions set on functions such as edit, copy, approve, delete, show versions, and so on in the context of workflow states and workflow roles.

There are three parts to a function privilege:

- The function being restricted.
- The roles allowed or not allowed to perform the function.
- The state during which users with those roles are allowed or not allowed to perform the function.

When a function privilege is in effect, it means that a user can perform that function only when the following conditions are true:

- The user has an appropriate role.
- The asset is in the correct state.
- The asset is **assigned** to the user.

This means that even if the user has the correct role and the asset is in the correct state, the user cannot perform that function on that asset unless the asset is assigned to that user.

Function Privileges and Step Actions

Function privileges restrict access to a function from the user interface only. This means that you can program step actions that invoke a function when a step is taken regardless of what the function privilege is set to at that moment.

The ApproveForPublish step action is an example. Even if you specify function privileges that restrict users from using the **Approve** option in the user interface, those same users can approve an asset with a workflow step if the workflow step invokes the ApproveFor Publish action and the user has the correct role to take the step.

In other words, you can use function privileges to prevent users from selecting and changing assets by mistake and use actions to invoke those functions in a highly controlled way.

Function Privileges: All or None

You cannot create just one function privilege: if you create even one function privilege that allows or restricts access to a function, you must create function privileges that cover all the other possible conditions for your workflow process.

For example, suppose that the only function that you want to restrict is the Delete function—you want to allow only the editors to delete article assets and only then if the article is in the Review state. If you create a function privilege that allows editors to delete article assets while it is in the Review state but you do not create any other function privileges, the only task that can be completed for an article when it is in the workflow is that editors can delete it while it is in the Review state. That is, no one can edit it, approve it, copy it, or even finish their assignments.

To implement the preceding condition—the Delete function is accessible to editors only when the article is in the Review state—you need to create the following function privileges:

- Delete – allowed for editors when the asset is in the Review state.
- Edit – allowed for all roles in all states.
- Abstain from Voting – allowed for all roles in all states.
- Copy – allowed for all roles in all states.

Continue down the list of functions, until you have at least one privilege specified for each function privilege listed in the “Functions” form for the workflow process.

Deadlines

There are two kinds of workflow deadlines:

- **Assignment deadlines** – which specify how long an asset should remain in an individual state. When you set a value for the **Estimated Time** field of a state, that creates a deadline for the assets that are assigned to workflow participants when the assets are in that state.
- **Process deadlines** – which specify how long it should take for an asset to go through the entire process.

Note that these different types of deadlines do not interact with each other—they are calculated separately and are mutually exclusive. Most likely you will use either one kind or the other, but not both for the same workflow process.

Setting and Overriding State Deadlines

When you create a workflow state, you can set an Estimated Time for it and determine whether reminder e-mails should be sent when assignments miss the deadline that is calculated from the Estimated Time.

When you create a workflow process that uses the state, you specify whether the Estimated Time for the state can be changed (overridden) when a user takes the step that moves the asset into that state.

If the deadline can be changed, any participant who takes the step can override the deadline unless you configure function privileges that restrict their ability to do so. Note that any participant who has a role that was designated as an administrator role for the workflow process can always override an assignment deadline if the deadline can be changed.

Setting Process Deadlines

When you create a workflow process, you determine whether a process deadline can be set. A process deadline is set when an asset is first placed in workflow, in the “Select Workflow” form. Unlike an assignment deadline, however, you cannot configure the process to send reminder e-mails when a process deadline is approaching.

If a process deadline can be set, any participant who places the asset in the workflow can set a process deadline for that asset—unless you configure function privileges that restrict their ability to do so. Additionally, any participant who has a role that was designated as an administrator role for the workflow process can always set a process deadline, if a deadline can be set.

Scheduling a Deadline Calculation

There are two kinds of actions:

- Actions that are invoked by a step. These actions are events that Content Server completes when a step is taken.
- Actions that are triggered by a deadline. These actions are queued and are triggered only after Content Server calculates the deadlines for the assets and determines which timed actions (if any) should be invoked.

You specify how often the deadlines are calculated by configuring the **Timed Action Event**. This is an event that invokes a background calculation process of all deadlines. It is similar to the publishing event that invokes the background approval calculation process.

Just as the publishing process calculates approvals to determine which assets should be published, the deadline calculation process that is invoked by the **Timed Action Event** calculates the deadlines for all assets that are participating in workflow processes—to determine whether any reminder messages should be sent for assignment deadlines and to determine the times that should be displayed for assets in the Due and Process Deadline columns of assignment lists.

You can configure the **Timed Action Event** on your management system to run as often as you find it necessary.

How Does a Workflow Process End?

A workflow process ends when there are no more states for the asset to progress through. This occurs when the final participant takes the **end step** for the workflow process.

An end step is the opposite of a start step—it has a From State but no To State. When a user takes the end step, it is moved to a “stateless” state, which means the asset is no longer in workflow and any function privileges set for that workflow process no longer apply.

Roles Required to Configure Workflow Processes

The workflow building blocks are located on two tree tabs in the Advanced interface:

- **Admin** – holds e-mail objects, actions, conditions, and the **Timed Action** Event.
- **Workflow** –holds workflow states and processes.

For access to the **Admin** tab, you must be assigned the GeneralAdmin role and have the xceladmin ACL assigned to your user account. For access to the **Workflow** tab, you must be assigned the WorkflowAdmin role for the site you want to create workflow processes for.

For more information about access rights in the Content Server interfaces, see Chapter 4, “Working with ACLs and Roles” and Chapter 5, “Configuring Users, Profiles, and Attributes.”

Planning Your Workflow Processes

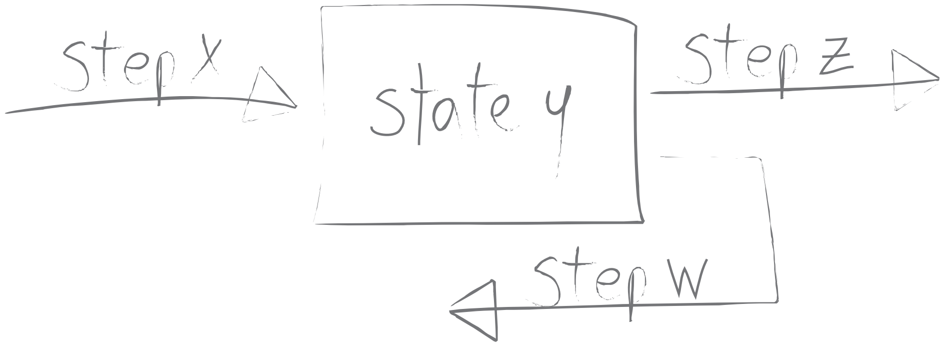
When you create a workflow process, you create steps that link together the states for that process. This means that you or someone else must create the workflow components—roles, e-mail messages, the various kinds of actions, step conditions, and states—before you can create a workflow process.

This section provides more details about the configuration of each of the workflow components so that you can plan and implement your workflow processes.

Start with a Sketch

Where do you begin? With sketches of the business processes that you need to implement as workflow processes:

- Use boxes to represent the states.
- Use arrows to represent the steps that connect the states.



As you read through the descriptions in this section, write on your sketches the details about which roles, actions, conditions, deadlines, and so on are appropriate for each state or step in the process.

Then, refer to your sketches as you use the Advanced interface to create your workflow processes, described in "Configuring Your Workflow Processes," on page 159.

Determine Roles and Participants

When you start planning your workflow processes, begin with the roles. What kind of functional groups participate in workflow? Then, determine how the individual users from those roles will become the participants in a workflow process for a specific asset.

Planning Roles

To determine the roles that you should create for your processes, ask yourself the following questions:

- What are the job titles or roles of the content providers who are using your CS management system?
- What do the people in each role do? (You can map the tasks that they complete to your states.)

- How are the roles organized? For example, is there one group of reviewers who review everything but several groups of content providers who are organized by subject (for example, sports writers, financial writers, marketing writers, and so on)
- Do certain groups of people work with specific asset types but not with others? If so, which roles should have access to each asset type that you plan to create a workflow process for?
- Would you ever need to notify someone who is not participating in a workflow about the status of an asset that is in workflow? If so, that person will need a role so a step or timed action can send an e-mail message to that person.

On your sketches of your workflow processes, write the names of the roles that will have the asset assigned to them in each state.

For information about creating roles, see Chapter 4, "Working with ACLs and Roles."

Selecting Individual Participants

When you configure a workflow process, there are several ways to determine which specific users participate in the workflow for each individual asset that goes through the workflow:

- When the workflow is assigned.

  You can configure the process so that the person who first assigns the workflow to the asset must select all the users who will participate. For each state, they select users from a list. The list includes all the users who have the correct role for that state.
- When a participant finishes the assignment.

  You can configure the process so that each participant determines who the next participant is when he or she finishes the assignment. That participant selects a user name from a list that includes all the users who have the correct role for the next state.

You specify how participants will be selected when you configure the steps.

Cross-Site Assignments and Participants

Assets can be shared between sites. If a shared asset is entered into a workflow process, should users from all the sites that have access to the asset be considered as candidates for participating in the workflow? If the answer is yes, enable the cross-site assignments feature.

When you use the cross-site assignments feature, users see all of their assignments from all the sites that they have access to in their assignment list no matter which site they are currently logged into. Having one, consolidated assignment list is very convenient for your users.

Keep in mind, however, that if your users have different roles in different sites and you are using function privileges in your workflow processes, they might not be able to work on an asset that they can see in their assignment lists. For example, say that some of your users have the author role in one site and the editor role in another. If you are using function privileges to restrict editing to editors in a certain state, they can see an asset that they aren't allowed to edit in their assignment list when they are logged in to the site where they function as authors.

To enable this feature, you set the value of the `xcelerate.crosssiteassign` property to true. This property is in the `futuretense_xcel.ini` property file. For information about the property, see the *Content Server Property Files Reference*.

Determine the E-mail Objects, Actions, and Conditions

An **action** is an event that is triggered in one of three ways:

- A step invokes it (step action).
- A deadline triggers it (timed action).
- A workflow situation triggers it (deadlock, group deadlock, and delegate actions)

A **condition** is an event that is assessed when a step is attempted. If the condition is not met, the step cannot be completed.

When you create an action or a condition, you identify an element. **Elements** are named pieces of code that are stored in the `ElementCatalog` table. It is the element that invokes the function represented by the action. If the element is coded to expect variables or arguments, you identify values for them when you create the action and those variables are then passed to the element when the action is triggered.

As an administrator, you are not responsible for coding elements. If your workflow needs cannot be met by the default workflow elements that are provided, your developers can code the functionality that you need. See the *Content Server Developer's Guide* for information about customizing workflow.

Content Server provides the following default workflow elements:

Element Name	Variables It Expects
OpenMarket/Xcelerate/Actions/Workflow/StepActions/ApproveForPublish	`target` The name of the publishing destination that the asset is to be approved for.
OpenMarket/Xcelerate/Actions/Workflow/StepActions/SendEmailToAssignees	`emailname` The name of the e-mail object to send.
OpenMarket/Xcelerate/Actions/Workflow/StepConditions/ExampleStepCondition	
OpenMarket/Xcelerate/Actions/Workflow/AssignmentActions/SendEmail	`emailname` The name of the e-mail object to send.
OpenMarket/Xcelerate/Actions/Workflow/DeadlockActions/SendEmailToAssignees	`emailname` The name of the e-mail object to send.
OpenMarket/Xcelerate/Actions/Workflow/GroupActions/SendEmailToAssignees	`emailname` The name of the e-mail object to send.

These elements are described in detail in the *Content Server Developer's Guide*.

With the exception of `ApproveForPublish` and `ExampleStepCondition`, these elements take a variable called `emailname` and send the e-mail message that is identified by that variable.

How do you determine the value of the `emailname` variable? By creating workflow **e-mail objects**. The names of the e-mail objects that you create in the **Admin** tab in the

Advanced interface are the names that you specify as the arguments with the `emailname` variable when you create actions that send e-mail messages.

About E-Mail Objects

E-mail objects are building blocks separate from the actions so that you can use them with more than one action. For example, the default deadlock action and the default group deadlock action use the same e-mail message (named Deadlock Message).

You create e-mail objects by giving them a name, a description, a subject line, and text for the body. When the message is sent, the text provided for the subject is placed in the subject line and the text provided for the body is placed in the body of the message.

Workflow E-Mail Variables

There are several variables that you can use in the subject line and body text which makes it easier for you to write e-mail messages that are personalized for each recipient.

The following table lists the default workflow variables that you can use with any e-mail message:

Variable Name	Description
`Variables.assetname`	The name of the asset. Use this variable in every e-mail message so the recipient knows which asset is being referred to.
`Variables.assigner`	The user name of the participant who assigned the asset to the person receiving the e-mail. Use this variable in e-mail messages for step actions that notify participants that a new asset has been assigned to them.
`Variables.time`	The time specified in the **Estimated Time** field for a state. Use this variable for timed actions.
`Variables.instruction`	The text that the previous participant entered in the **Action to Take** field of the “Finish My Assignment” form when he or she finished the assignment. Use this variable in e-mail messages for step actions that notify participants that a new asset has been assigned to them.

For examples of custom e-mail variables, see the e-mail object named Deadlock Message and the corresponding deadlock action element named `OpenMarket/Xcelerate/Actions/Workflow/DeadlockActions/SendEmailToAssignees`.

Default Workflow E-Mail Objects

The default workflow e-mail objects are these:

- Assignment Due Reminder – specifies the asset and the time that it is due.
- Assignment Message – specifies the asset, the person who assigned it, and in the message from the **Action To Take** box on the “Finish My Assignment” form.

- Deadlock Message – describes how the deadlock occurred by listing the users and the steps that they took.
- Rejection Message –is similar to the Assignment Message, but states that the asset was rejected by the previous participant (the assigner).

Planning Your E-mail Objects

To determine the e-mail objects that you need to create, you must also determine the actions that will send the e-mail messages held in the objects. By using the e-mail variables in the subject and body, it is likely that you can use the same e-mail object with more than one timed action or step action.

Typically, you need to compose e-mail messages that specify the following kinds of things:

- An asset has been assigned to a participant. (For step actions.)
- An asset has been delegated to a new participant. (For delegate actions.)
- A deadline is approaching for an asset. (For timed actions.)
- A deadline for an asset has been missed. (Also for timed actions.)
- An asset is in a deadlock. (For a deadlock action.)
- A group of assets are in a deadlock. (For a group deadlock action.)

In the **Admin** tab, double-click **Email**. Examine the default e-mail objects to determine whether you can use them. You can modify the default messages or create your own e-mail messages.

In one corner of your sketches of the workflow processes, list the e-mail messages that you will need for that workflow.

About Timed Actions

Timed actions are actions that are based on the deadline of a state. If you specify a deadline for a state, you can specify a timed action to remind the participants about the deadline.

You create a timed action by giving it a name and a description, specifying an element, and then providing argument values for the element, if necessary.

To create a timed action that sends e-mail notices about a deadline, specify the `OpenMarket/Xcelerate/Actions/Workflow/AssignmentActions/SendEmail` element and use the **Argument** field to specify which e-mail message to send.

When you create a state, you specify which timed actions to use and when to trigger them, relative to the deadline specified for the state. You can specify more than one timed action for each state.

When the Timed Action Event runs and calculates the deadlines for all the assets currently participating in workflow, the appropriate timed actions are triggered.

Default Timed Actions

There is one default timed action: Send Email. It uses the `OpenMarket/Xcelerate/Actions/Workflow/AssignmentActions/SendEmail` element to send the Assignment Due Reminder e-mail message.

Planning Timed Actions

When planning your timed actions, you must also consider the following workflow components:

- The states that you will create, because these actions are triggered in terms of the deadlines that you set for your states.
- The e-mail objects that you need to create, because it is likely that your timed actions will send e-mail messages

Because the time that a timed action is triggered is determined outside of the action itself (you do this in the form for the state), you can probably create a small number of generic timed actions—whose only difference is the e-mail message they send—and use them repeatedly with your states.

In the **Admin** tab, expand **Workflow Actions**, then **Timed Actions**. Examine the Send Email action to determine whether you can use it. You can modify the default timed action or create your own.

On each of your sketches of your workflow processes, list the timed actions that you need for that process near the list of the e-mail messages.

About Delegate Actions

When you assign a delegate action to a workflow process, the action is triggered whenever a participant (or the workflow administrator) delegates an assignment to another participant.

You create a delegate action by giving it a name and a description, specifying an element, and then providing argument values for the element, if necessary.

You can specify one delegate action per workflow process.

Default Delegate Action

There are no default delegate actions provided.

Planning Delegate Actions

It is likely that your delegate action will send an e-mail message to the participant to whom an assignment is delegated. If this is the case, you can create a delegate action that uses any of the default workflow elements that send e-mail and create a new e-mail object for that element to send.

On each of your sketches of your workflow processes, write the name of the delegate action that you will use for that process. (You can assign one for each process.)

About Step Actions

When you assign a step action to a step, the action is triggered when a participant takes the step.

You create a step action by giving it a name and a description, specifying an element, and then providing argument values for the element, if necessary.

There are two general categories of step actions: those that complete functions and those that send e-mail messages to the participants who are being assigned the asset by the step.

Step actions are completed regardless of function privileges. That is, if you create a step action that performs a Content Server function, the action does not check the function privileges of the participant taking the step. This means that you can restrict access to a function from the user interface and require participants to use a step in a workflow in order to complete a specific function (approve for publish, for example).

You can specify one or more step actions for each step.

Default Step Actions

The default step actions are these:

- Approve for Publish, which uses the `OpenMarket/Xcelerate/Actions/Workflow/StepActions/ApproveForPublish` element to approve the asset. Then, the next time a publishing process runs, the asset is published (as long as all of its dependencies are also approved).

  To use this action for your workflow processes, you must specify the publishing destination(s) that the asset is to be approved for by using the `targets` argument.
- Send Assignment Email, which uses the `OpenMarket/Xcelerate/Actions/Workflow/StepActions/SendEmailToAssignees` element to send the Assignment Message e-mail object to all the participants assigned the asset (that is, who are specified in the **Assignment Method** for the step).
- Send Rejection Email, which uses the `OpenMarket/Xcelerate/Actions/Workflow/StepActions/SendEmailToAssignees` element to send the Rejection Message e-mail message to the new assignee (the participants specified in the **Assignment Method** for the step).

Planning Step Actions

To determine what kind of step actions you need to create, you must consider the following workflow components:

- The steps that you will create for the process. Do people need to be notified that a step has assigned an asset to them? If so, you need a step action that sends an e-mail message.
- The states that the steps move the assets into. Does the state represent the asset after a function has been completed? If so, you need a step action that implements that function.

In the **Admin** tab, select **Workflow Actions**, then **Step Actions**. Examine these actions to determine whether you can use them. You can both modify the default step actions and create your own.

On each of your sketches of your workflow processes, write the name of the step actions next to the appropriate steps. You can assign one or more step actions for each step.

About Step Conditions

When you assign a step condition to a step, the condition is assessed when the step is taken. The condition determines whether the step can be completed or not.

You create a step condition by giving it a name and a description, specifying an element, and then providing argument values for the element, if necessary. If you want to use conditions in your workflow processes, talk to your developers about the conditions you need them to implement through elements.

You can specify one or more conditions for each step.

Default Step Conditions

There is one default step condition: Example Step Condition. It is a “hello world”-style example that illustrates how to code an element to check for a condition. You and your developers should examine it to learn how it works and then your developers should create their own condition elements.

Planning Step Conditions

When thinking about step conditions, you must consider the steps and the state. If you determine that conditions are necessary for your workflow processes, ask your developers to create them.

On each of your sketches of your workflow processes, write the name of any step conditions that you need next to the appropriate steps.

About Deadlock Actions

When you assign a deadlock action to a step that can result in a deadlock, the action is triggered whenever an asset becomes deadlocked during the step.

You create a deadlock action by giving it a name and a description, specifying an element, and then providing argument values for the element, if necessary.

You can assign one or more deadlock actions per step.

Default Deadlock Action

There is one default deadlock action: Send Deadlock Email. It uses the `OpenMarket/Xcelerate/Actions/Workflow/DeadlockActions/SendEmailToAssignees` element to send the Deadlock Message e-mail message.

Planning Deadlock Actions

When thinking about possible deadlock actions, you need to consider the steps that you will create for your workflow processes. If you plan to design your steps so that deadlocks cannot occur, there is no need to create any deadlock actions.

In the **Admin tab**, expand **Workflow Actions**, then **Deadlock Actions**. Examine this action to determine whether you can use it. You can modify the default deadlock action or create your own.

In your sketches of your workflow processes, write the name of the appropriate deadlock action next to any step that can result in a deadlock.

About Group Deadlock Actions

A group deadlock action is triggered when workflow group becomes deadlocked. Whoever creates the workflow group selects the group deadlock action. That person can select one or more for each group.

The group deadlock actions are invoked in addition to any other actions that are associated with the states or the steps in the workflow process.

Default Group Deadlock Action

There is one default group deadlock action: Send Deadlock Email. It is identical to the default deadlock action.

Planning Group Deadlock Actions

The same considerations apply for group deadlock actions as for deadlock actions. However, because it is a content provider who selects the group deadlock action for a workflow group, be sure that you give a group deadlock action a meaningful, unambiguous name.

Determine the States

When you create a state, you specify the following kinds of information:

- Name and description. The name of a state should be meaningful and should represent the kind of work that is being done while the asset is in that state.
- Amount of time an asset should remain in this state (the **Estimated Time**). The deadline is calculated in terms of the number of hours or days since the asset entered the state.

  Note that you set the estimated time (the deadline) in terms of hours or days rather than as a specific date because a state deadline is calculated for each asset as it passes through the state.
- The timed actions that should occur and when they should occur, in terms of days or hours before or after the deadline.

There are no default states, although there are example states provided with all of the sample sites. On a system where the sample sites are installed, select **Workflow** > **States** and then select a state to examine to learn about how the sample site states are configured.

Planning Your States

To determine the states that you need to create, ask yourself the following kinds of questions:

- Which roles participate in each state? On your sketches, write the names of the roles next to the states.
- Do any of these states apply to more than one asset type? If so, it’s possible that you can reuse the same state in more than one workflow process.
- How long should it take to complete each state?
- Does this amount of time differ by asset type or by site? If yes, you cannot reuse the same state in more than one workflow process or share the workflow process with another site.
- If there is a deadline, should Content Server notify the participant when the deadline is approaching? If yes, when? The day before? Several hours before? And which timed action (which determines which e-mail message) should be used?
- Should there be a notice when a deadline is missed? If yes, when? And which timed action should be used?

On the sketches of your workflow processes, list all of this information next to each state.

Depending on your goals, you might consider creating a workflow process that does not really end. For example, the Hello Asset World sample site workflow process ends after a HelloArticle asset has been approved. However, in a case like this, someone could open and save the asset by mistake, which would mean that it is no longer approved and won’t get published.

To keep assets from being edited after they have been approved and before they have been published, you could create a final state that holds assets after they have been approved with a function privilege on the state that restricts anyone from editing the asset.

Be sure that if you do create a state like this that you create a step that can move the asset back into an editing state in the workflow so someone can edit it on purpose.

Determine the Steps

You create steps within a process. You use the steps to link together the states. This is how the process is actually created.

You specify the following kinds of information for each step in your process:

- The name of the step. The name should be meaningful and should describe the path being taken. For example, Send for Review, Send for Approval, and so on.
- The **From State**, which is the state that the step is moving the asset from. If the step has no **From State**, it is the start step that begins the workflow process. A **From State** is required for each step other than the start step.
- The **To State**, which is the state that the step is moving the asset to. If the step has no **To State**, it is the end step the ends the workflow process.
- Which roles are authorized for the step.

  If this is the very first step, the roles you specify determine which users can select this workflow for an asset. If you are using a Start Menu item to assign the workflow process to an asset, be sure that the roles assigned to this first (start) step match the roles assigned to the Start Menu item.

  For subsequent steps after the first (start) step, the roles that you select must either match or be a subset of the roles that were selected to be notified by the previous step. Otherwise, the asset cannot leave the state because no one who is assigned the asset is allowed to move it to the next state.
- Who gets the asset next? (That is, when this step is completed). You specify the assignee for the To State by selecting one of the following assignment methods:
  - Retain “From State” assignees.

    This option assigns the asset to its current assignees—that is, to the user who completes the step.

    This option is useful when you are creating a step that returns to the same state (which creates an iterative state) or when it is the start step and you want the asset to be assigned to the person who selects the workflow process (whether through a start menu item or by selecting it on the “Status” form).
  - No assignments; control actions with function privs.

    This option keeps the asset in the workflow process, which means that function privileges are enforced, but the asset is not actually assigned to anyone.

> **Note**
>
> When an asset is assigned to the current assignees (Retain "From State assignees option) or assigned to no one (No assignments option), the workflow system does not record workflow history for that step.

  - Assign from a list of participants.

    With this option, you select roles. When a user assigns the workflow process to an asset, a list of users with the roles that you selected is displayed. The user selects one or more users from the list, and those users are assigned the asset when it is in the state that this step moves the asset to.

  - Choose assignees when the step is taken.

    You also select roles with this option. This option means that the person who takes this step (by completing the assignment) during the workflow process selects the user who is assigned the asset next from a list of users with the roles selected for the step.

- Whether the estimated time for the state that this step moves the asset to can be changed or not. (Called an Assignment Deadline.)
- Any step actions that should occur.
- Any step conditions that should be assessed.
- If the step can be taken by more than one user (assignee), whether all the assignees must take this step before the asset can move to the next state. (The **All assignees must vote** field.)
- If the step can result in a deadlock, what the deadlock action should be.
- If the process will be used for a workflow group, whether all the assets in the group must complete this step before any of the assets can move to the next state. (The **Step is group synchronized** field.)

There are example steps in all of the sample site workflow processes. On a system where the sample sites are installed, select **Workflow** > **Processes** and examine a process to learn about how the sample workflow steps are configured.

Planning Your Steps

When determining what your steps should be, consider the following questions:

- Every process needs a start step. Before you create a start step, you must first determine how the asset is created and then placed into workflow.

  Does the first participant create the asset, place it into workflow, and then keep working on the asset? If so, configure the step so that it is assigned to the person taking the start step (choose the **Retain "From State" assignees** option) and remember to create a start menu item that assigns the workflow to the asset by default.

  Or does a supervisor create the asset and then assign to it a participant? If so, configure the start step so that the person taking the start step has to select the assignees.

- What is the order of the states?

- How many paths do you need between each state? This answer determines how many steps you need.
- Do you need to create an iterative state? If so, configure the step that takes the asset back to that state so that it is assigned to the person taking the step (choose the **Retain "From State" assignees** option).
- For steps that move an asset from a state in which more than one person is working on that asset, must all the participants complete their assignment before the step can be taken? If so, configure the step to be an all-voting step.
- Does the asset have more than one possible path from a state? (That is, there needs to be more than one step with the same From State.) If yes, do any of the steps from the state require that all participants vote the same way before the asset can progress? If yes, try to design the process so only one of the steps that lead from that state require that all the participants vote the same way. If you have two all-voting steps from the same state, deadlocks can occur.
- For each step, should Content Server notify the affected participants when the step is completed and the asset progresses to the next state? If yes, which step action (which determines the e-mail message) should the step use?
- Do groups of related assets ever need to be worked on at the same time? If so, can you configure your steps to work appropriately for both a single asset and a group of assets? If not, create separate workflow processes for the workflow groups.
- For a workflow group, are there any states that all the assets in the group must enter at the same time? (For example, all the assets should be approved at the same time.) If so, configure that step to be a synchronize step. Note that it's best to have only one synchronize step per workflow process.
- What should happen to the asset if you decide not to publish it, after all? Should you have a cancel step? How many cancel steps do you need?

On your sketches of your workflow processes, right next to the box that represents a step, list the step actions, the roles who can take the step, the roles who should be notified when the step is taken, whether the step is an all-voting step, and whether the step is a synchronize step.

Determine the Function Privileges

When you create a function privilege, users are either allowed or not allowed to perform the function in the user interface when the asset is in the state specified by the privilege. You create one set of function privileges for each workflow process (if you are using this feature, that is).

The following table lists the Content Server functions that you can control access to in the user interface during a workflow process:

Function	Description
Abstain from Voting	Removes a user from the workflow process. Appears as an option in the **Workflow Commands** drop-down list on the asset's "Status" form.

Function	Description
Approve Asset for Publishing	Marks an asset as approved for publishing. Appears as an option in the drop-down list in the action bar on an asset's **Edit**, **Inspect**, and **Status** forms.
Authorize	Allows users who are granted roles that permit this function to modify permissions to the asset.
Build	Builds a collection asset. Appears as an option in the drop-down list in the action bar on an asset's **Edit**, **Inspect**, and **Status** forms.
Checkout	When revision tracking is enabled for the asset type, this function checks out an asset to the user. Appears as the **Check Out** button at the top of an asset's **Edit** and **Status** forms.
Copy	Copies an asset. Appears as an option in the drop-down list in the action bar on an asset's "Inspect" form.
Delegate Assignment	Delegates an asset that was assigned to a user through a workflow process to another user (one who has the appropriate role for the asset while it is in that state). Appears as an option in the **Workflow Commands** drop-down list on the asset's "Status" form.
Delete	Deletes an asset. Appears as an icon in the action bar on an asset's **Edit**, **Inspect**, and **Status** forms. It also appears as an icon next to the asset in lists.
Edit	Displays the asset in the "Edit" form. Appears as an icon in the action bar on an asset's **Edit**, **Inspect**, and **Status** forms. It also appears as an icon next to the asset in lists
Finish Assignment	Invokes the next step in a workflow process, which assigns the asset to the next participant. Appears as an option in the **Workflow Commands** drop-down list on the asset's "Status" form.
Inspect	Allows users who are granted roles that permit this function to view fields that are otherwise restricted. Users without the "Inspect" permissions cannot view any asset-type-specific data, and instead see a generic page that contains only standard, unrestricted, fields (such as ID, name, and description).
Place Page	Places a page on the **Site Plan** tab. Appears in as an option in the right-mouse menu on the **Site Plan** tab.

Function	Description
Remove from Group	Removes an asset from a workflow group. Appears as an option in the **Workflow Commands** drop-down list on the asset’s “Status” form.
Remove from Workflow	Removes the asset from a workflow process. Appears as an option in the **Workflow Commands** drop-down list on the asset’s “Status” form.
Rollback	Reverts an asset back to one of the previous versions of the asset that are stored by the revision tracking system. Appears as the **Rollback** button at the top of an asset’s “Edit” and “Status” forms.
Set Assignment Deadline	Sets a deadline for how long the asset should remain in the next state. A deadline set with this option overrides any deadlines set for the state in the workflow process. Appears as an option in the “Select Workflow” form and in the “Finish My Assignment” form if the workflow process is configured to allow someone to override the estimated time for a state.
Set Export Target Path/ Filename	Restricts users from filling in the **Path** and **Filename** fields on an asset’s “New” or “Edit” forms.
Set Participants	Sets the participants for a workflow process. Appears as an option in the **Workflow Commands** drop-down list on the asset’s “Status” form. Additionally, this function prompts a user to select participants when the workflow process is configured so that the participant finishing an assignment must select the next participant (person being assigned the asset).
Set Process Deadline	Sets a deadline for how long it should take a specific asset to go through the entire workflow process. Appears as an option in the “Select Workflow” form for assets and when a workflow group is created or edited if the workflow process is configured to allow someone to set a process deadline.
Share Assets	Shares the asset with another CM site. Appears as an option in the drop-down list in the action bar on an asset’s “Edit,” “Inspect,” and “Status” forms when the asset type is enabled for more than one CM site.
Show Participants	Displays a list of the participants in the workflow process that is currently assigned to the asset. Appears as an option in the **Workflow Commands** drop-down list on the asset’s “Status” form.

Function	Description
Show Status	Displays the “Status” form for an asset. Appears as an option in the drop-down list in the action bar on an asset’s “Edit,” “Inspect,” and “Status” forms.
Show Version	Lists information about each the versions of the asset that the revision tracking system is storing. Appears as the **Show Versions** button at the top of an asset’s “Edit” and “Status” forms when revision tracking is enabled for this asset type.

Planning Function Privileges

For each workflow process, determine which functions you want to restrict access to, during which states you want to restrict those functions, and which roles should or should not have access to those functions when assets that are assigned to them are in those states.

List any restrictions that you want to enforce next to the appropriate states on your sketches of your workflow processes.

Remember that in order for your function privileges to enforce the restrictions that you intend, the roles of the users specified for the state in the function privilege must match the roles of the users associated with those states in the workflow process itself. If the roles don’t match, the result can be a condition in which no one can move the asset out of a state because the users who are allowed to work with the asset by the privilege are not allowed to by the state.

Also, remember that you cannot create just one function privilege: if you create even one function privilege that allows or restricts access to a function, you must create function privileges that cover all the other possible conditions for your workflow process.

Implementing Simplified Access Control

Because function privileges are associated with workflow processes, you can restrict individual users’ access to specific functions only when an asset is participating in a workflow process.

What can you do if you do not want to design and implement workflow processes but you still want to control access to specific functions? Create a simplified workflow process with one state and use the **No assignments; control actions with function privs** assignment method.

When a step moves an asset into a state using the **No assignments** assignment method, the asset does not appear on anyone’s assignment list, it just stays in the state which means that any function privileges assigned to that state are enforced.

To implement access control in this way, follow these general steps:

1. Create a state. It needs a name and a description. It does not need a deadline or any timed actions.
2. Create a new workflow process. Select all the roles you want to enforce restrictions for and all the asset types you want to use this workflow process for.
3. Create one step— the start step that puts assets of those types into the state. For the step, select the **No assignments** option. Enable the step for the roles that you want to be able to create assets of this type and assign this workflow to.

4. Configure the function privileges that you want to enforce for assets in the state that you created.
5. Create start menu items for the asset types that automatically assign this workflow process. Be sure that the roles who can use the start menu item match the roles who are assigned to the start step in the workflow process.

Now when users select **New >** ***asset type***, the new asset is automatically placed into your single-state workflow. Because the workflow has only the start step which places all assets of that type into the single state, those assets do not leave the state and the function privileges are always enforced.

Determine Additional Workflow Process Details

If you have been sketching your workflow processes, filling in all the details about actions, conditions, e-mail messages, states, steps, and function privileges, by now you have planned nearly the entire design of your workflow processes.

Although the main task when creating a process is to create the steps that link the states (which is how your business process is represented), you must also specify the following kinds of information for each workflow process:

- Its name and description. The name should be descriptive so that users select the correct workflow process for the correct asset types.
- Which sites can use the process.
- Which asset types can use the process.
- Which roles can participate. This is a superset of all the roles that are designated in the steps.
- Which role will serve as the administrator of the workflow. A workflow administrator can delegate assignments on behalf of other participants.
- Whether a process deadline can be set for assets that participate in this workflow process.
- What the delegate action is.
- What the steps are. For each step, you specify the information described in the section named “Determine the Steps,” on page 146.
- Whether any of the CS content application functions should be restricted to users in certain roles while they are working on assets in specific states. For function privileges, you specify the information described in “Determine the Function Privileges,” on page 148.

There are example workflow processes delivered with all of the sample sites. They are described in detail in the next section.

Sample Site Workflow Processes

Each sample site has a workflow process. Before creating a workflow process of your own, it is a good idea to examine the sample site workflow processes. There are five:

- **HelloArticle Process**, the sample workflow process for the **HelloArticle** asset type that the HelloAssetWorld sample site provides.

- **BF: Normal Article Process**, the sample workflow process for the **article** asset type that the Burlington Financial sample site provides.
- **GE Lighting Process**, the sample workflow process for the **product** asset type that the GE Lighting sample site provides.
- **BF: Fund Category Process**, the sample workflow process for the **product parent** asset type that the extension to the Burlington Financial sample site provides. (Available only when Engage is installed.)
- **BF: Segment Process**, the sample workflow process for the **segment** asset type that the extension to the Burlington Financial sample site provides. (Available only when Engage is installed.)

The following sections describe the first three workflow processes: HelloArticle Process, BF: Normal Article Process, and GE Lighting Process.

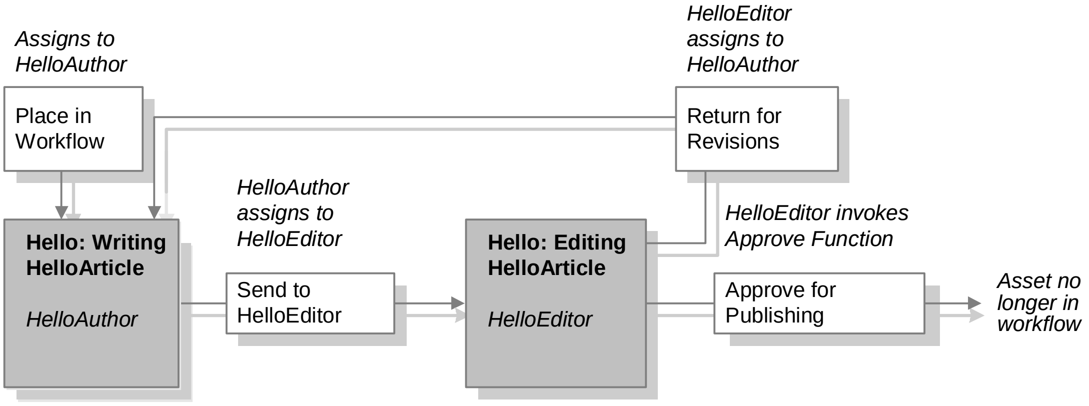
HelloArticle Process

At HelloAssetWorld, the authors, Joe and Moe, write HelloArticles. When their HelloArticles are ready to be reviewed, Joe and Moe send them to their editor, Flo.

Flo reads and edits the HelloArticle assets sent to her. If she decides that a HelloArticle asset needs more work, she sends it back to the author for revisions. If the HelloArticle asset is ready to be published, Flo approves it.

These work practices are reflected in the HelloArticle workflow process, which has two states and four steps, as follows:

When sketched as a flow chart, the HelloArticle workflow process looks like this:


Assigns to HelloAuthor
HelloEditor assigns to HelloAuthor
Place in Workflow
Return for Revisions
HelloAuthor assigns to HelloEditor
HelloEditor invokes Approve Function
Hello: Writing HelloArticle
HelloAuthor
Send to HelloEditor
Hello: Editing HelloArticle
HelloEditor
Approve for Publishing
Asset no longer in workflow

1. The **Place in workflow step** puts the HelloArticle into the **Hello: Writing HelloArticle state**.

   The start menu item named New HelloArticle is enabled for users who have the HelloAuthor role. This start menu item has the HelloArticle Process specified as the default workflow for assets of that type. When either Joe or Moe (the two users with the HelloAuthor role) selects **New > HelloArticle**, the HelloArticle Process is assigned to the asset, which invokes the first step in the process (Place in workflow).

   The Place in workflow step is configured so that the HelloArticle asset is assigned to the person who creates the asset.

Roles: HelloAuthor

Assign to: “From State” assignees, which means that the person who creates the asset is assigned the asset.

Actions: None

2. When an author is done with his article, he finishes his assignment, which invokes the **Send to editor step**. This step moves the HelloArticle asset to the **Hello: Editing HelloArticle state**, and assigns it to Flo, the only user with the editor role.

   Roles: This step is enabled for users with the HelloAuthor role only. This means that although editors and designers are allowed to create articles in order to troubleshoot and test the design, they cannot write articles that are meant to be published to the delivery system.

   Assign to: HelloEditor

   Actions: SendAssignmentEmail, which sends the default Assignment Message e-mail message to the editor participant.

3. When Flo, the editor, is done reviewing the HelloArticle asset that Joe or Moe assigned to her, she has two choices (steps that are enabled for users in the editor role) when she finishes her assignment:

   a. The **Return for revisions step** sends the HelloArticle asset back to **Hello: Writing HelloArticle state** and assigns it back to the author who originally assigned it to her so that he can incorporate her comments.

      Role: HelloEditor

      Assign to: HelloAuthor

      Action: SendRevisionNoticeEmail, which sends the Revisions Required Message e-mail to the original author participant when the editor takes the Return for revisions step.

   b. The **Approve for publishing step** invokes the Approve for publishing function, which approves the asset. The next time the publishing process runs, the HelloArticle asset is published (as long as all of its dependencies are met).

      Role: HelloEditor

      Assign to: No one.

      Action: ApproveForPublish, which approves the HelloArticle for publishing.

      The Approve for publishing step has no To state, which means that after this step is completed, the workflow process has ended for this asset.

This simple process has no step conditions or timed, deadlock, group deadlock, or delegate actions.

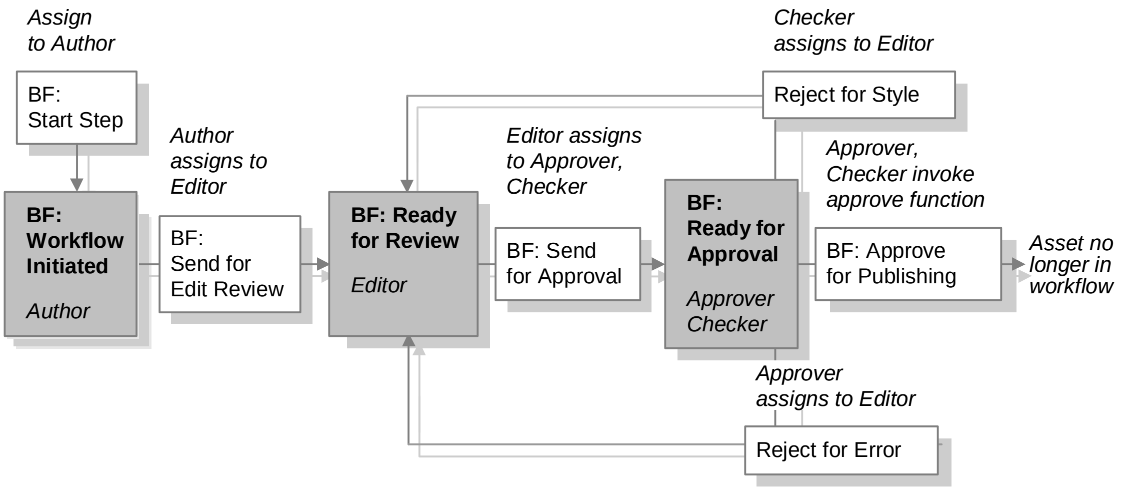
BF: Normal Article Process

At Burlington Financial, authors write articles and then send them to an editor. The editors not only review the articles, they actually modify and edit them. When they are finished with their input into the article, the editors send the article to a fact checker and an approver.

If either the fact-checker or the approver decide that there is some reason that the article cannot be published, they can reject the article, which sends it back to the editor. The editor then fixes the article and sends it back to the checker and approver.

Both the approver and the fact-checker must approve the asset before it is considered approved.

These work practices are reflected in the BF: Normal Article Process, which has six steps and three states, as follows:



1. The **BF:Start Step step** places the article asset into the **Workflow Initiated state**.

   The Start Step step is enabled for users in the Author, Editor, Checker, and Approver roles. This means that any of the users in those roles can select the Normal Article Process from the **Workflow Process** drop-down list on the “Status” form for an article asset.

   When a user selects the BF: Normal Article Process, Content Server invokes the BF:Start Step step that then starts the workflow process for the asset

   The user who assigns the workflow to the asset must then select the participants. for each state from a list of users who have the appropriate roles for the process. The user from the author role is assigned the asset for the BF: Workflow Initiated state.

   Roles: Author, Editor, Checker, Approver

   Assign to: Author

   Actions: Send Assignment Email, which sends the Assignment Message e-mail object to the author assigned the article by the BF:Start Step step.

   Timed Actions: SendEmail, which sends the Assignment Due Message e-mail object one day before the deadline of the Workflow Initiated state.

2. When the author is finished writing the article asset, he or she finishes the assignment which invokes the **BF: Send for Edit Review step** that moves the article into the **BF: Ready for Review state**.

   The article is now assigned to the editor who was selected when the workflow was first assigned to the article.

   Roles: Author

Assign to: Editor

Actions: Send Assignment Email, which sends the Assignment Message e-mail object to the editor assigned the article by the BF: Send for Edit review step.

Timed Actions: SendEmail, which sends the Assignment Due Message e-mail object one day before the deadline of the BF: Ready for Review state.

3. The editor edits the article. When the editor is finished, he or she finishes the assignment, which invokes the **BF: Send for Approval step** that moves the article asset into the **BF: Ready for Approval state**.

   The article is now assigned to the checker and the approver who were selected when the article was placed into workflow.

   Roles: Editor

   Assign to: Checker, Approver

   Actions: Send Assignment Email, which sends the Assignment Message e-mail object to the checker and approver who are assigned the article by the BF: Send for Approval step.

   Timed Actions: SendEmail, which sends the Assignment Due Message e-mail object one day before the deadline of the BF: Ready for Approval state.

4. The checker and the approver examine the article asset. When they are completed, they finish the assignment. However, because both of them have two possible steps, the “Finish My Assignment” form displays those steps as options. They must select the step (option) that represents their decision about the status of the article.

   The approver can take either of the following steps:

   - The **BF: Reject for Error step**, which moves the article back to the **BF: Ready for Review state** and assigns it the original editor.

     Roles: Approver

     Assign to: Editor

     Actions: Send Rejection Email, which sends the Rejection Message e-mail object to the editor assigned the article by the BF: Reject for Error step.

     Timed Actions: Now that the article is back in the Ready for Review state, a new review deadline is set and the SendEmail timed action sends the Assignment Due Message e-mail object one day before the new deadline for this state.

   - The **BF: Approve for Publishing step**. This is an “all-voting” step that is enabled for both the checker role and the approver role, which means that the step will not be completed until both the checker and the approver choose this step for the article.

     Roles: Checker, Approver

     Assign to: No one.

     Actions: ApproveForPublish, which marks the asset approved by the checker.

   The checker can take either of the following steps:

   - The **BF: Reject for Style step**, which moves the article back to the **BF: Ready for Review stat**e and assigns it the original editor.

     Roles: Approver

     Assign to: Editor

Actions: Send Rejection Email, which sends the Rejection Message e-mail object to the editor assigned the article by the Reject for Style step.

Timed Actions: Now that the article is back in the Ready for Review state, a new review deadline is set and the SendEmail timed action sends the Assignment Due Message e-mail object one day before the new deadline for this state.

- The Approve for Publishing step.

When both participants have taken the Approve for Publishing step, the article is approved for publishing. The Approve for Publishing step has no To State, which means that after the step is completed, the workflow is over for this asset.

Note that because neither the BF: Reject for Error step or the BF: Reject for Style step are all-voting, if either the approver or the checker rejects the article, it immediately returns to the editor.

This process has no step conditions, delegate actions, or deadlock actions.

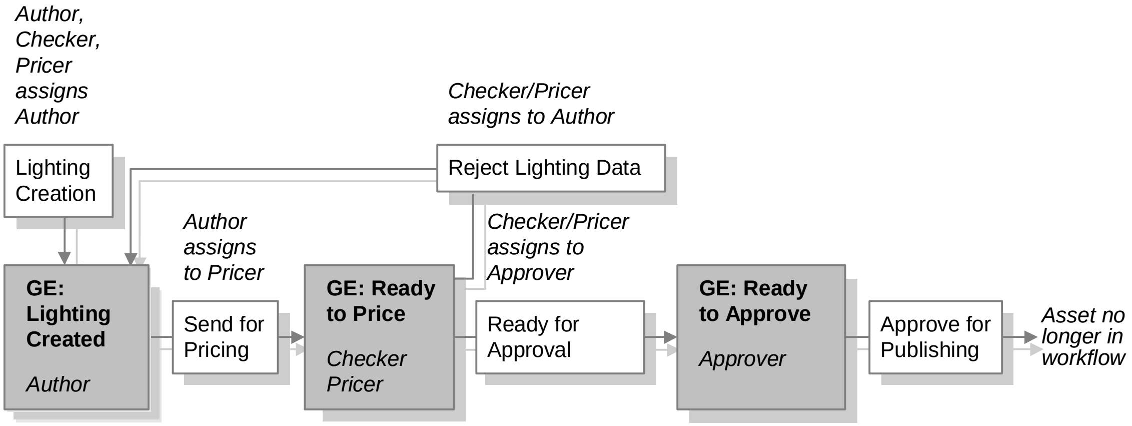
GE Lighting Process

At GE Lighting, authors create product assets but do not determine their prices. Users with the pricer role are responsible for pricing all products. Therefore, after authors have created a new product, they assign the product asset to a pricer so that it can obtain a price and to a checker to have all other data verified.

The pricer determines the price and the checker verifies the other data. Either one can send the asset back to the author for modifications or they can send it on to the approver.

The approver then approves the asset for publishing.

These work practices are reflected in the GE Lighting Process, which has five steps and three states, as follows:



1. The **Lighting Creation step** places the product asset into the **GE: Lighting Created** state and assigns it to an author.

   The Send for Price step is enabled for users in the author, checker, or pricer roles. Any user in one of those roles can she selects the GE Lighting Process on a product asset’s “Status” form. Selecting this process invokes the Lighting Creation step that then places the asset into the workflow process.

The user who assigns the workflow to the asset must also then select the participants for each state from a list of users who have the appropriate role for each state.

The author participant selected for the GE: Lighting Created state is assigned the asset.

Roles: Author, Checker, Pricer

Assign to: Author

Actions: SendAssignmentEmail

2. When the author is finished working on the product, he or she finishes the assignment. Finishing the assignment invokes the **Send for Review step** that moves the product asset into the **GE: Ready to Price state**.

   The product asset is now assigned to the checker and the pricer who were selected when the product was first assigned the workflow.

   Roles: Author

   Assign to: Checker, Pricer

   Actions: SendAssignmentEmail

3. The checker and the pricer examine the asset. The pricer assigns a price and the checker verifies the rest of the data. When they are finished with the asset, they finish the assignment.

   Because there are two possible steps from the GE: Ready to Price state, the “Finish My Assignment” form displays two options:

   - The **Reject Lighting Data step**, which moves the product asset back to the **GE: Lighting Created state** and assigns it to the original author.

     Roles: Checker, Pricer

     Assign to: Author

     Actions: SendAssignmentEmail

   - The **Send for Approval step**, which moves the product asset to the **GE: Ready to Approve state** and assigns it to the approver who was selected when the asset was first assigned this workflow process.

     Roles: Checker, Pricer

     Assign to: Approver

     Actions: SendAssignmentEmail

     Because the **Send for Approval** step is an all-voting step, both the checker and the pricer must take this step before the asset is moved to the GE: Ready to Approve state.

4. The approver examines and approves the product by finishing the assignment. Finishing the assignment invokes the Approve for Publishing step. The Approve for Publishing step has no To State, which means that after the step is completed, the workflow is over for this asset.

   Roles: Marketer

   Assign to: No one

   Actions: ApproveForPublish, which marks the asset as approved for publishing to a specific destination.

Configuring Your Workflow Processes

This section presents the procedures for creating all the components for a workflow process and then stitching those components together into your workflow processes.

Remember that to create e-mail objects, actions, and conditions or to schedule the timed action event, you need access to the **Admin** tab, which means that you need the GeneralAdmin role and the xceladmin ACL. To create workflow states and workflow processes, you need access to the **Workflow** tab, which means that you need the WorkflowAdmin role.

Overview

Before you create a workflow process, you must create the individual workflow components that you need for that process. Here are the general steps that you must take presented in the order you perform them:

1. Plan your workflow process by drawing it. See section "Planning Your Workflow Processes," on page 137 for help with this step. Then refer to your sketch and notes throughout this section.
2. Create the roles that you need for your workflow processes. See Chapter 4, "Working with ACLs and Roles," for help with this step. Be sure that any users who will participate in workflow have user profiles created for them. Otherwise, they will not receive e-mail messages from the workflow process.

Note

If you want the pool of users who are candidates to be participants in a workflow to include folks from all the sites that an asset is shared to, be sure to enable the cross-site assignments feature. See "Cross-Site Assignments and Participants," on page 138 for details.

3. Create the e-mail objects that you need for your actions and enable the `xcelerate.emailnotification` property in the `futuretense_xcel.ini` file. See "Setting Up E-Mail Objects," on page 160.
4. Create the step actions, timed actions, deadlock actions, group deadlock actions, and delegate actions that you need. See "Setting Up the Workflow Actions and Conditions," on page 161.
5. Create your states. See "Setting Up the States," on page 165.
6. Create your process. While creating your workflow process, you create the steps for that process. The steps link together the states so they occur in the proper order. Additionally, while creating your process, you configure any function privileges that you need. "Setting Up the Workflow Processes," on page 166.
7. If your states have deadlines, be sure to configure the Timed Action Event so that the deadlines of assets are calculated regularly and the appropriate timed actions (if any) are invoked in a timely way. "Setting Up the Timed Action Event," on page 164.
8. Test your workflow processes. See "Testing Your Workflow Process," on page 175.

9. Set up your start menu shortcuts so that workflow processes are assigned automatically to assets when they are created. For help with start menu items, see "Creating Start Menu Items," on page 108.

Setting Up E-Mail Objects

You create e-mail objects so that your step and timed actions can send them to the appropriate participants at the appropriate times. Examine the sketches of your workflow processes, determine the e-mail messages you need, and then use the procedures in this section to create and edit them.

Your user account must give you access to the **Admin** tab in order for you to create e-mail objects.

Enabling the E-mail Feature

To ensure that your workflow process can successfully send e-mail messages, the following conditions must be true:

- In the `futuretense.ini` file, the properties on the **Email** tab must be configured to provide information about your e-mail server.
- In the `futuretense_xcel.ini` file, the `xcelerate.emailnotification` property must be set to `true`.

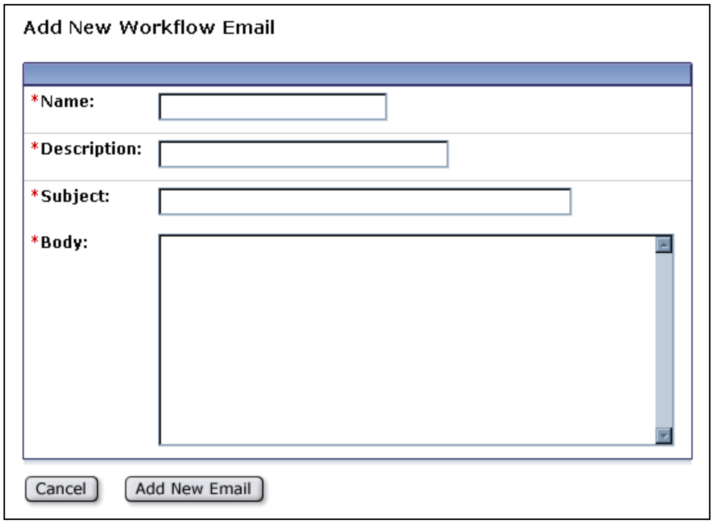
  For information about the properties cited above and using the Property Editor, see the *Property Files Reference*.
- The workflow participants must have e-mail addresses specified in their user profiles. (For information about creating user profiles, see "Creating and Editing a User Profile," on page 82.)

Creating E-Mail Objects

To create e-mail objects

1. In the **Admin** tab, expand **Email** and double-click **Add New**.

   Content Server displays the "Add New Workflow Email" form:



Add New Workflow Email

*Name:

*Description:

*Subject:

*Body:

2. In the **Name** field, enter a unique name of up to 36 characters. This is the name that you provide to the `emailname` variable when you use this e-mail object with an action.
3. In the **Description** field, enter a short, informative description of up to 36 characters.
4. In the **Subject** field, enter a short, informative subject for the e-mail message. (See "About E-Mail Objects," on page 140 for a list of variables that you can use.)
5. In the **Body** field, enter the text for the message. (See "About E-Mail Objects," on page 140 for a list of variables that you can use.)
6. Click **Save**.

Editing E-Mail Objects

To edit e-mail objects

1. In the **Admin** tab, expand **Email**, then double-click the desired e-mail object.
2. In the action bar, click **Edit**.
3. In the "Edit Workflow Email" form, make the necessary changes. (See "About E-Mail Objects," on page 140 for a list of variables that you can use.)
4. Click **Save**.

Deleting E-Mail Objects

To delete e-mail objects

1. In the **Admin** tab, expand **Email**, and double-click the desired e-mail object.
2. In the action bar, click **Delete**.

   Content Server displays a warning message.
3. Click **Delete Email**.

Setting Up the Workflow Actions and Conditions

When you create any kind of action or a step condition, you identify an element and you supply values for the variables that the element expects. If you are creating an action that sends an e-mail message, you identify which e-mail object to send with the `emailname` variable.

Before you begin creating actions or step conditions, be sure that you have the elements and e-mail objects that you need. Content Server provides several default action elements and e-mail objects. Use the Advanced interface to examine the e-mail objects and use Content Server Explorer to examine the `ElementCatalog` table. Determine which elements you plan to use and then write down the entire name of those elements. If you need additional e-mail messages, consult the previous section, "Setting Up E-Mail Objects," on page 160 and create the e-mail messages that you need.

The following procedures describe how to create, edit, and delete the workflow actions—step, timed, delegate, deadlock, and group deadlock action—and step conditions. Your user account must give you access to the **Admin** tab in order for you to create actions or conditions.

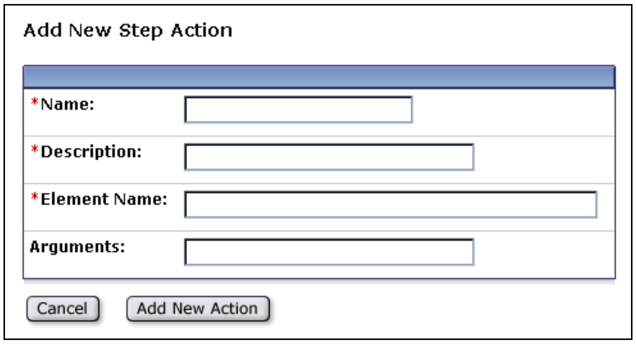
Creating Workflow Actions and Conditions

To create workflow actions and conditions

1. In the **Admin** tab, expand **Workflow Actions**.
2. Under **Workflow Actions**, expand the category describing the action you are creating, then double-click **Add New**.

   The available categories are: **Step Actions**, **Step Conditions**, **Timed Actions**, **Delegate Actions**, **Deadlock Actions**, or **Group Deadlock Actions**.

   Content Server displays the "Add New" form corresponding to the type of action you selected. The example below shows the "Add New Step Action" form:



Add New Step Action

*Name:	
*Description:	
*Element Name:	
Arguments:	

3. In the **Name** field, enter a unique name of up to 40 characters.
4. In the **Description** field, enter a short, informative description of up to 40 characters.
5. In the **Element Name** field, enter the name of the element in its entirety.

   For example, to use the default workflow element `SendEmailToAssignees`, enter:

```
OpenMarket/Xcelerate/Actions/Workflow/StepActions/
   SendEmailToAssignees
```

6. In the **Arguments** field, use the following convention to supply values for the arguments or variables that the element needs to function correctly:

   `name=value`

   For the `SendEmailToAssignees` element, for example, you must provide a value for the `emailname` variable (that is, the name of an e-mail object). For example:

```
emailname=AssignmentDueReminder
```

   If an element takes more than one variable, separate the name/value pairs with the ampersand (&) character. For example:

```
name1=value1&name2=value2
```

7. Click **Add New Action**.

Configuring Approve for Publish Step Actions

The approval process approves an asset to a specific publishing destination. To use the default Approve for Publish step action with your workflow processes, you specify the publishing destination for the assets that are approved by the action.

If all the assets for all of your workflow processes are published to the same destination, you can simply configure the existing Approve For Publish action. However, if some types of assets are published to a different destination than the others, you must create an additional Approve for Publish action for each publishing destination, or combination of publishing destinations.

The Approve for Publish step action uses the `OpenMarket/Xcelerate/Actions/Workflow/StepActions/ApproveForPublish` element which takes the `targets` variable. You can use this same element with as many additional approval step actions as you need.

For the default Approve for Publish step action, provide a value for the `targets` variable. Possible values for the `targets` variable are the names of any of the publishing destinations that have been created on this CS system.

For example:

`targets=serverX`.

To specify more than one publishing destination, separate each with a comma. For example:

```
targets=serverX,serverY
```

Note

If the name of a publishing destination that is referenced by the `targets` variable is changed, you must also change the value of the `targets` variable in your Approve for Publish steps.

Editing Workflow Actions and Conditions

To edit workflow actions and conditions

1. In the **Admin** tab, expand **Workflow Actions**.
2. Under **Workflow Actions**, expand the category describing the action you want to edit.

   The available categories are: **Step Actions**, **Step Conditions**, **Timed Actions**, **Delegate Actions**, **Deadlock Actions**, or **Group Deadlock Actions**.
3. Under the selected category, double-click the action you want to edit.
4. In the action bar, click **Edit**.
5. Make your changes, then click **Save**.

Deleting Workflow Actions and Conditions

To delete workflow actions and conditions

1. In the **Admin** tab, expand **Workflow Actions**.
2. Under **Workflow Actions**, expand the category describing the action you want to delete.

   The available categories are: **Step Actions**, **Step Conditions**, **Timed Actions**, **Delegate Actions**, **Deadlock Actions**, or **Group Deadlock Actions**.
3. Under the selected category, double-click the action you want to delete.
4. In the action bar, click **Delete**.

   Content Server displays a warning message.
5. Click **Delete Action**.

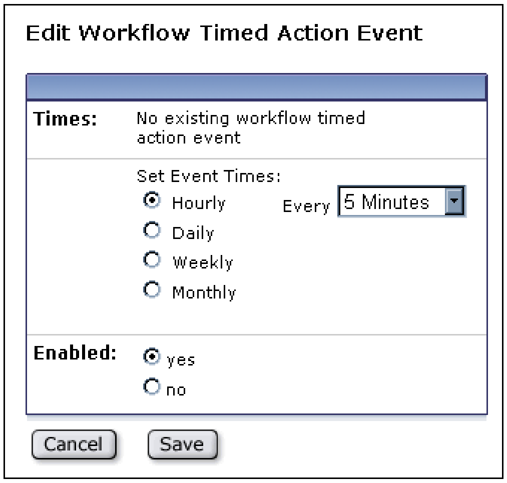
Setting Up the Timed Action Event

Before any of your timed actions can be triggered, you must configure the timed action event so that it calculates deadlines at regularly occurring intervals. Note that there can be only one timed action event per CS system.

To set up timed action events

1. In the **Admin** tab, double-click **Timed Action Event**.
2. In the action bar, click **Edit**.

   Content Server displays the “Edit Workflow Timed Action Event” form.



Edit Workflow Timed Action Event

Times: No existing workflow timed action event

Set Event Times:
● Hourly Every 5 Minutes
○ Daily
○ Weekly
○ Monthly

Enabled: ● yes ○ no

3. Set the schedule for how often the state deadlines should be calculated in terms of months, days, hours and minutes, depending on the selected interval.
4. In the **Enabled** field, select **yes**.
5. Click **Save**.

   A summary of the schedule is displayed.

Content Server uses the same abbreviations and codes to summarize the schedule for the timed action event that it does for your publishing events. For information about how to read the schedule, see “Reading the Schedule Abbreviations,” on page 288.

Setting Up the States

When you create a workflow state, you specify a deadline, select a timed action, and configure when the timed action should run. Therefore, before you begin creating your workflow states, be sure that you have created the timed actions that you need.

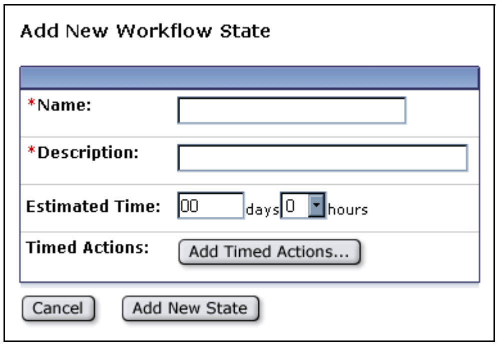
To work with workflow states, your user name must be assigned the WorkflowAdmin role for the site that you are working with.

Creating Workflow States

To create workflow states

1. In the **Workflow** tab, expand **States** and double-click **Add New**.

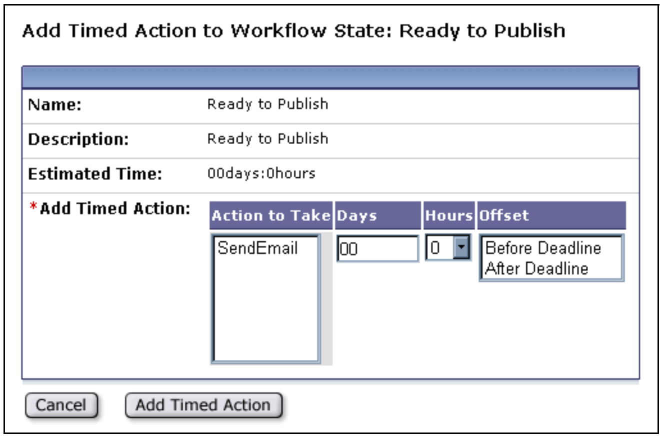
   Content Server displays the “Add New Workflow State” form:



Add New Workflow State

*Name:	
*Description:	
Estimated Time:	00 days 0 hours
Timed Actions:	Add Timed Actions...

2. In the **Name** field, enter a unique, meaningful name of up to 40 characters.
3. In the **Description** field, enter a short, informative description of up to 40 characters.
4. (Optional) In the **Estimated Time** fields, configure the deadline for assets in this state. You can specify a deadline in terms of days, hours, or a combination of the two.
5. (Optional) Click **Add Timed Actions** and complete the following steps:



Add Timed Action to Workflow State: Ready to Publish

Name:	Ready to Publish
Description:	Ready to Publish
Estimated Time:	00days:0hours
*Add Timed Action:	

Action to Take	Days	Hours	Offset
SendEmail	00	0	Before Deadline After Deadline

   a. In the **Action to Take** list, select a timed action.

   b. In the **Offset** field, specify whether the action should be triggered before the deadline or after the deadline.

   c. In the **Days** field and/or the **Hours** field, specify how many hours or days before or after the deadline (that you specified in step 5) that the action is to be triggered.
   d. Click **Add Timed Action**.
   e. Repeat this entire step for each timed action that you want to set up for this state.
6. Click **Add New State**.

Editing Workflow States

To edit workflow states

1. In the **Workflow** tab, expand **States** and double-click the state you want to edit.
2. In the action bar, click **Edit**.
3. In the "Edit" form, make the desired changes.
4. (Optional) To change the timed action or the time that it is scheduled to run, click **Add Timed Actions**, make the appropriate changes, and click **Add Timed Actions** again.
5. Click **Save**.

Deleting Workflow States

To delete workflow states

1. In the **Workflow** tab, expand **States** and double-click the state you want to delete.
2. In the action bar, click **Delete**.
3. Content Server displays a warning message.
4. Click **Delete State**.

Setting Up the Workflow Processes

When you create a workflow process, you specify global process information and then you create steps, assigning step actions to them as needed. The steps in the process create the flow of the process by linking the states in a specific order.

Before you can begin creating a workflow process, you must have already created your step actions, step conditions (if necessary), and states.

To work with workflow processes, your user name must be assigned the Workflow Admin role for the site that you are working with.

Note

If your content providers will use workflow groups, be sure to enable the Workflow Groups tab for them. For information, see "Creating the Workflow Groups Tab," on page 123.

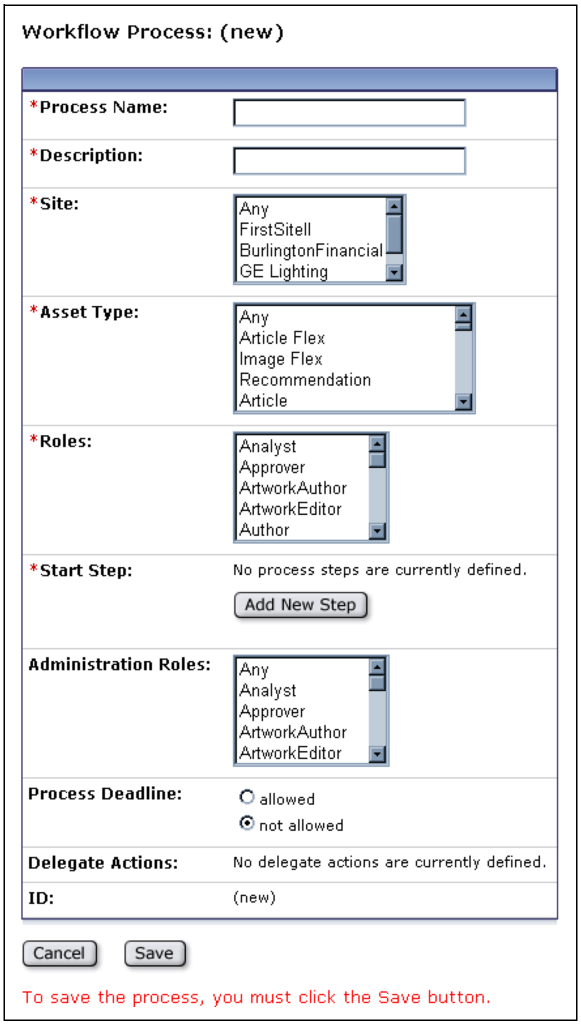
Creating Workflow Processes

When you create a workflow process, you configure three categories of information: global process settings, steps, and function privileges.

Step A: Name and Set Global Settings for the Process

1. In the **Workflow** tab, expand **Processes** and double-click **Add New**.

   Content Server displays the "Workflow Process: (new)" form.



Workflow Process: (new)

*Process Name:	
*Description:	
*Site:	Any FirstSiteII BurlingtonFinancial GE Lighting
*Asset Type:	Any Article Flex Image Flex Recommendation Article
*Roles:	Analyst Approver ArtworkAuthor ArtworkEditor Author
*Start Step:	No process steps are currently defined.
Administration Roles:	Any Analyst Approver ArtworkAuthor ArtworkEditor
Process Deadline:	○ allowed ● not allowed
Delegate Actions:	No delegate actions are currently defined.
ID:	(new)

To save the process, you must click the Save button.

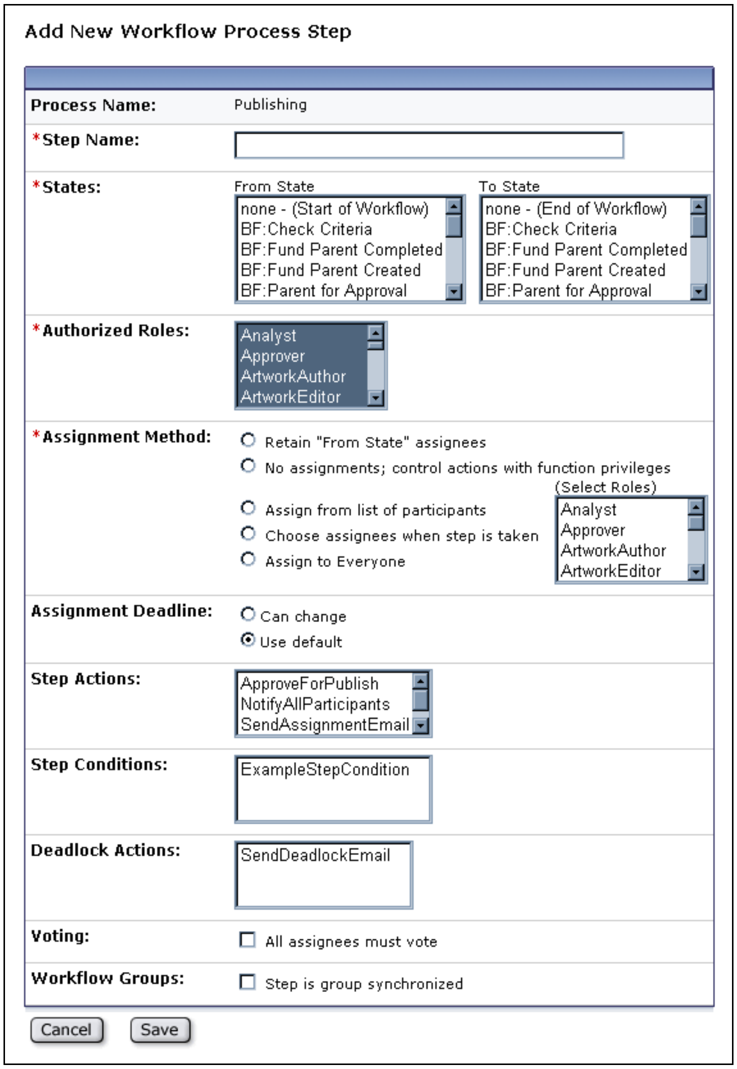
2. In the **Name** field and enter a unique name of up to 25 characters.
3. In the **Description** field, enter a short, informative description of up to 64 characters.
4. In the **Sites** list, select the sites that can use this workflow process.
5. In the **Asset Type** list, select the asset type(s) that this workflow process is for.
6. In the **Roles** list, select the roles that will participate.
7. In the **Administration Roles** list, select the roles that can act as the administrator for this workflow process when an asset is using it.
8. In the **Process Deadline** section, specify whether the workflow administrator (or other user if you configure function privileges that allow it) can set a process deadline for the assets that participate in this workflow process.

9. If you have configured one or more delegate actions, select the appropriate actions in the **Delegate Actions** list.

10. In the **Start Step** section, click **Add New Step**.

    Content Server displays the “Add New Workflow Process Step” form.

Step B: Create the Start Step



Add New Workflow Process Step

Process Name:	Publishing
***Step Name:**	
***States:**	From State none - (Start of Workflow) BF:Check Criteria BF:Fund Parent Completed BF:Fund Parent Created BF:Parent for Approval To State none - (End of Workflow) BF:Check Criteria BF:Fund Parent Completed BF:Fund Parent Created BF:Parent for Approval
***Authorized Roles:**	Analyst Approver ArtworkAuthor ArtworkEditor
***Assignment Method:**	○ Retain "From State" assignees ○ No assignments; control actions with function privileges ○ Assign from list of participants ○ Choose assignees when step is taken ○ Assign to Everyone (Select Roles) Analyst Approver ArtworkAuthor ArtworkEditor
Assignment Deadline:	○ Can change ● Use default
Step Actions:	ApproveForPublish NotifyAllParticipants SendAssignmentEmail
Step Conditions:	ExampleStepCondition
Deadlock Actions:	SendDeadlockEmail
Voting:	□ All assignees must vote
Workflow Groups:	□ Step is group synchronized

1. In the **Step Name** field, enter a unique, meaningful name of up to 64 characters.

2. Configure the states as follows:

    a. In the **From State** list, select **none (start of workflow)**.

    b. In the **To State** list, select the name of the first state in the workflow process.

3. In the **Authorized Roles** list, select the roles that are allowed to take this step by assigning this workflow to an asset.

4. In the **Assignment Method** section, specify which roles will be assigned the asset by selecting one of the following options:

- **Retain "From State" assignees**, which, for a start step, means that the user who assigns the workflow to the asset is assigned the asset
- **No assignments; control access with function privs**
- **Assign from list of participants**
- **Choose assignees when step is taken**

  If you select either of the last two options above, select the appropriate roles from the list that is to the right of those options. (For definitions of these options, see "Determine the Steps," on page 146.)
- **Assign to everyone**

5. In the **Assignment Deadline** section, determine whether the workflow administrator (or another user holding the appropriate function privileges) can override the Estimated Time (deadline) set for the state that this step moves the asset to.
6. (Optional) In the **Step Actions** list, select one or more step actions that should be invoked when this step is taken.
7. (Optional) In the **Step Conditions** list, select a step condition, if appropriate.
8. Click **Save**.

   The start step is added and the "Steps for Workflow Process" form is displayed.
9. Click **Save**.

   Content Server saves the process and displays it in the "Inspect" form. Note that this step appears as the **Start Step** in the form.
10. Continue to the next procedure.

Step C: Create the Subsequent Steps

Use the following procedure to create the rest of the subsequent steps, including the end step:

1. In the "Inspect" form of the workflow process, click **Edit** in the action bar.
2. At the bottom of the "Edit" form, click **Add/Edit Steps**.
3. In the "Steps for Workflow Process" form, click **Add New Step**.

   Content Server displays the "Add New Workflow Process Step" form.
4. In the **Step Name** field, enter a unique, meaningful name of up to 64 characters.
5. Configure the states as follows:
   a. In the **From State** list, select the name of the state that you selected as the **To State** for the previous step.
   b. In the **To State** list, select the name of the next state in the workflow process. (Note that if you want to create an iterative step, select the same state in the **From State** and the **To State** lists.)
6. In the **Authorized Roles** list, select the roles that are allowed to take this step by finishing the assignment. The roles that you select from this list should either match or contain a subset of the roles that you selected for the **Assignment Method** in the workflow step that immediately precedes this workflow step.
7. In the **Assignment Method** section, specify which roles will be assigned the asset by selecting one of the following options:

- **Retain "From Step" assignees**

  If the **From State** and the **To State** are different, the first person who takes this step (finishes the assignment) is assigned the asset, even if the previous step assigned the asset to more than one participant.

  If the **From State** and the **To State** are the same, every user who was assigned the asset by the last step is assigned the asset again with this step.

- **No assignments; control access with function privs**
- **Assign from list of participants**
- **Choose assignees when step is taken**

  If you select either of the last two options above, select the appropriate roles from the list that is to the right of those options. (For definitions of these options, see "Determine the Steps," on page 146.)

- **Assign to everyone**

8. In the **Assignment Deadline** section, determine whether the workflow administrator (or other user if you configure the appropriate function privileges) can override the Estimated Time (deadline) set for the state that this step moves the asset to.
9. (Optional) In the **Step Actions** list, select one or more step actions that should be invoked when this step is taken.
10. (Optional) In the **Step Conditions** list, select a step condition, if appropriate.
11. If the step moves the asset from a state in which more than one participant was working on the asset and you want all participants to finish their assignments before the step can complete, select the **All Assignees must vote** option.
12. If there are any other steps in this process with the same **From State** and any of those steps are also all-voting steps (**All Assignees must vote** is selected), select a **Deadlock Action**. For information about deadlocks and avoiding them, see "Managing Deadlocks," on page 130 and "About Delegate Actions," on page 142.
13. If this workflow process will be used for workflow groups and you want all the assets to progress at the same time to the **To State** that you selected in step 4 of this procedure, select the **Step is group synchronized** option. FatWire recommends that you create only one synchronized step in the process.
14. Click **Save**.

    The step is saved and the "Steps for Workflow Process" form is displayed.

15. Click **Save**.

    Content Server saves the process and displays it in the "Inspect" form.

16. Repeat this procedure for each step—except the end step—that you want to create for this process.

    To create the end step, continue with the next procedure.

    To configure function privileges, continue to "Step E: (Optional) Configure Function Privileges," on page 171.

Step D: (Optional) Create the End Step

An end step ends the workflow which means that the asset is no longer in a process and no longer has function privileges controlling access to it, if you are using function privileges. Use the following procedure to create an end step for the workflow process:

1. In the “Inspect” form of the workflow process, click **Edit** in the action bar.
2. In the “Edit Process” form, click **Add/Edit Steps** (a button at the bottom of the form).
3. In the “Steps for Workflow Process” form, click **Add New Step**.

   Content Server displays the “New Workflow Process Step” form.
4. In the **Step Name** field, enter a unique, meaningful name of up to 64 characters.
5. Configure the states as follows:
   a. In the **From State** list, select the name of the state that you selected as the **To State** for the previous step.
   b. In the **To State** list, select none **(end of workflow)**.
6. In the **Authorized Roles** list, select the roles who are allowed to take this step by finishing an assignment. The roles that you select from this list should either match or contain a subset of the roles that you selected for the **Assignment Method** of the workflow step that immediately precedes this workflow step in the process.
7. (Optional) In the **Step Actions** list, select one or more step actions that should be invoked when this step is taken.
8. (Optional) In the **Step Conditions** list, select a step condition, if appropriate.
9. If the step moves the asset from a state in which more than one participant was working on the asset and you want all participants to finish their assignments before the step can complete, select the **All Assignees must vote** option.
10. If there are any other steps in this process with the same **From State** and any of those steps are also all-voting steps (**All Assignees must vote** is selected), select a **Deadlock Action.** For information about deadlocks and avoiding them, see “Managing Deadlocks,” on page 130 and “About Delegate Actions,” on page 142.
11. If this workflow process will be used for workflow groups and you want this step to be performed on all the assets in the group at the same time, select the **Step is group synchronized** option. FatWire recommends that you create only one synchronized step in the process.
12. Click **Save**.

    The start step is added and the “Steps for Workflow Process” form is displayed.
13. Click **Save**.

    Content Server saves the process and displays it in the “Inspect” form.

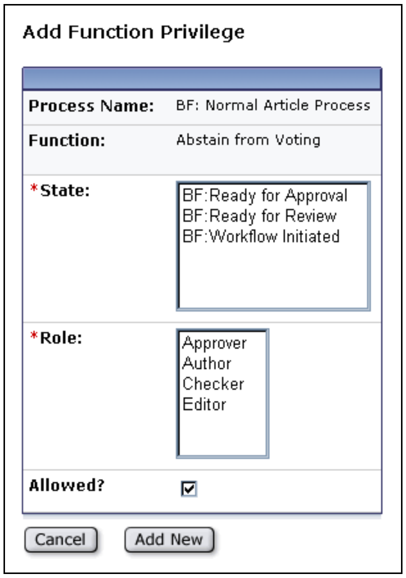
If you do not need to configure function privileges, this workflow process is completed. If you do need to configure function privileges, continue to the next procedures.

Step E: (Optional) Configure Function Privileges

To set up function privileges

1. Find the desired workflow process and open its “Inspect” form.
2. In the action bar, click **Edit**.
3. Click **Add/Edit Function Privs** at the bottom of the form.
4. In the “Functions for Workflow Process” form, scroll down to the function for which you want to set a privilege.
5. Click **New** next to the desired function.

Content Server displays the “Add Function Privilege” form:



Add Function Privilege

Process Name:	BF: Normal Article Process
Function:	Abstain from Voting
*State:	BF:Ready for Approval BF:Ready for Review BF:Workflow Initiated
*Role:	Approver Author Checker Editor
Allowed?	☑

6. Select the appropriate state from the **State** list.
7. In the **Roles** list, select the roles that are allowed or not allowed to perform this function when an asset is in this state.
8. Do one of the following:
   - To allow users with the selected roles to perform the function, select the **Allowed** check box.
   - To restrict users with the selected roles from performing the function, clear the **Allowed** check box.
9. Click **Add New**.

   Content Server saves the privilege and redisplays the “Functions for Workflow Process” form.
10. Repeat steps 3–8 for each function privilege that you need to configure. For more information about function privileges, see “Determine the Function Privileges,” on page 148.
11. After you have configured all of your function privileges, click **Save**.

    Content Server saves the workflow process and redisplays the “Workflow Process” form.

The workflow process is complete.

Editing Workflow Processes

To edit workflow processes

1. In the **Workflow** tab, expand **Processes** and double-click the workflow process you want to edit.
2. In the action bar, click **Edit**.
3. Make your changes as follows:

- To modify the name, description, or any of the other global settings, make your changes directly in the form.
- To edit the steps, see "Editing Steps," on page 173
- To edit the function privileges, see "Editing Function Privileges," on page 173.

4. When you are finished, click **Save**.

Editing Steps

To edit a workflow step

1. In the **Workflow** tab, expand **Processes** and double-click the workflow process you want to work with.
2. In the action bar, click **Edit**.
3. In the "Workflow Process" form, click **Add/Edit Steps** (at the bottom of the form).
4. In the "Steps for Workflow Process" form, click **Edit** next to the step you want to edit.
5. In the "Edit Workflow Process Step" form, make your changes, then click **Save**.

   Content Server saves the process and displays it in the "Inspect" form.

Editing Function Privileges

To edit a function privilege, you must delete it and then re-create it.

To edit a function privilege

1. In the **Workflow** tab, expand **Processes** and double-click the workflow process you want to work with.
2. In the action bar, click **Edit**.
3. In the "Workflow Process" form, click **Add/Edit FunctionPrivs** (at the bottom of the form).
4. In the "Functions for Workflow Process" form, click **Remove** next to the function that you want to change.
5. In the pop-up dialog box that appears, click **OK**.
6. Click **New** next to the function you just removed.
7. In the "Add Function Privilege" form, make your selections and click **Save**.
8. In the "Functions for Workflow Process" form, click **Save**.

   Content Server saves the workflow process and displays it in the "Inspect" form.

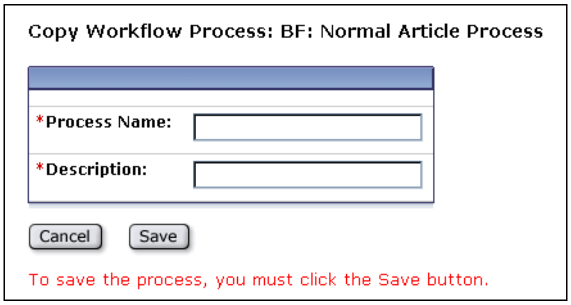
Copying Workflow Processes

You can copy a workflow process, which can save you some steps in configuring additional processes.

To copy a workflow process

1. In the **Workflow** tab, expand **Processes** and double-click the workflow process you want to copy.
2. In the action bar, select **Copy Process** from the drop-down list.

Content Server displays the “Copy Workflow Process” form.



Copy Workflow Process: BF: Normal Article Process

*Process Name:

*Description:

To save the process, you must click the Save button.

3. In the **Process Name** field, enter a unique name for the process.
4. In the **Description** field, enter a short, informative description of the process.
5. Click **Save**.
6. Edit the process as necessary. See the following procedures for help:
   - “Editing Workflow Processes,” on page 172.
   - “Editing Steps,” on page 173.
   - “Editing Function Privileges,” on page 173.

Deleting Workflow Processes

To delete a workflow process

1. In the **Workflow** tab, expand **Processes** and double-click the workflow process you want to delete.
2. In the action bar, click **Delete**.

   Content Server displays a waning message.
3. Click **Delete Process**.

   The process has been deleted.

Deleting Steps

To delete workflow steps

1. In the **Workflow** tab, expand **Processes** and double-click the workflow process you want to work with.
2. In the action bar, click **Edit**.
3. In the “Edit Process” form, click **Add/Edit Steps** (at the bottom of the form).
4. In the “Steps for Workflow Process” form, click **Remove** next to the step you want to delete.
5. In the pop-up dialog box that appears, click **OK**.
6. Click **Save**.

   Content Server saves the workflow process and displays it in the “Inspect” form.

Deleting Function Privileges

To delete function privileges

1. In the **Workflow** tab, expand **Processes** and double-click the workflow process you want to edit.
2. In the action bar, click **Edit**.
3. In the "Workflow Process" form, click **Add/Edit FunctionPrivs** (at the bottom of the form).
4. In the "Functions for Workflow Process" form, click **Remove** next to the function privilege you want to delete.
5. Click **Save**.

   Content Server saves the workflow process and displays it in the "Inspect" form.

Testing Your Workflow Process

Before you move your workflow process to the management system and implement it there, test it on your development system.

To test your workflow process

- Log in as a user with administrator access and configure start menu items that assign workflow processes to assets, if necessary.
- Log in as a user who has a role that has been authorized to take the start step of the workflow process. Create an asset, select the workflow process (only if the start menu item does not assign it), and then finish the assignment.
- Log in as the participant who has the assignment now. Finish the assignment and then log in as the next participant. Continue through the entire workflow in this manner.
- Verify that the e-mail messages that should be sent are being sent.
- Verify that your workflow sends the asset through the process correctly.

Moving Your Work

Typically you create and fine-tune a workflow process on a development system to ensure that it functions exactly as you need it to before you introduce that workflow process to the management system.

When your workflow is ready to be used by content providers, use the Initialize Mirror Target feature to move your workflow components to the management system.

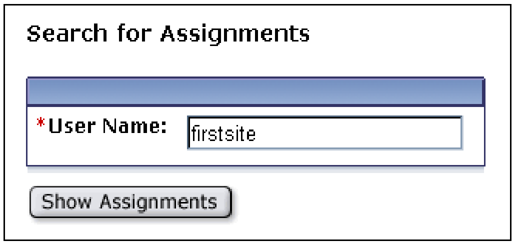
For information about this feature, see "Migrating a Site from One System to Another," on page 282.

Clearing Workflow Assignments

People go on vacation, get reassigned to new work groups, and move on to different jobs. What should happen to their workflow assignments in these situations? One option for handling work assignments that cannot be finished by the original assignee is to delegate the assignments to someone else. However, even with the delegate feature, you, the administrator, might still be called on to clear assignments from some user's assignment list.

To clear assignments for a user

1. In the **Admin** tab, double-click **Clear Assignments**.
2. In the "Search for Assignments" form, enter the desired user name. For example:



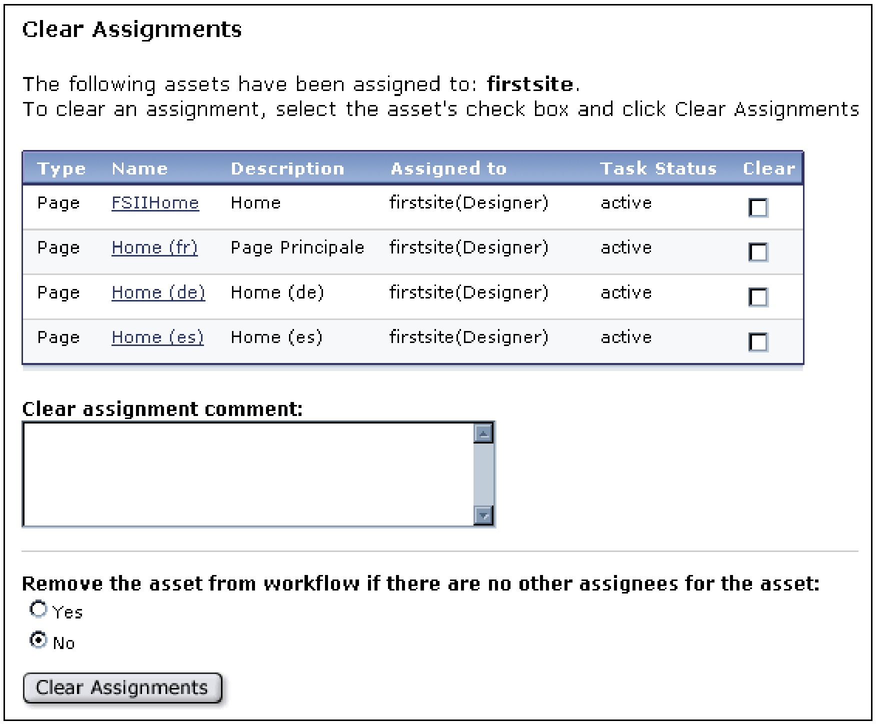
Search for Assignments

*User Name: firstsite

3. Click **Show Assignments**.

   Content Server displays a list of assets that are currently assigned to that user.

For example:



Clear Assignments

The following assets have been assigned to: **firstsite**.
To clear an assignment, select the asset's check box and click Clear Assignments

Type	Name	Description	Assigned to	Task Status	Clear
Page	FSIIHome	Home	firstsite(Designer)	active	□
Page	Home (fr)	Page Principale	firstsite(Designer)	active	□
Page	Home (de)	Home (de)	firstsite(Designer)	active	□
Page	Home (es)	Home (es)	firstsite(Designer)	active	□

Clear assignment comment:

Remove the asset from workflow if there are no other assignees for the asset:

○ Yes
● No

4. In the “Clear Assignments” form, select the **Clear** check box next to each assignment you want to clear.
5. (Optional) In the **Clear assignment comment** field, enter a brief explanation of the reason for which you are clearing the selected assignments. The explanation you enter in this field appears in the **Action Taken** field in the **Workflow History** section of the asset’s “Status” form.
6. Do one of the following:
   - If you want the asset to be removed from workflow if it is not assigned to anyone else, select **Yes**.
   - If you want the asset to remain on the assignment list of any other user who is also assigned the asset, select **No**.
7. Click **Clear Assignments**.

   Content Server displays a “Clear Assignments Report” summarizing your changes.

Chapter 10

Replicating CM Sites

To speed up the deployment of online sites, Content Server provides Site Launcher. This new feature enables you to replicate a suitably-prepared CM site and, in the process, either share or copy its components to the new site. You can then modify the new site as necessary, and deploy it. This chapter provides an overview of Site Launcher, replication guidelines, requirements and options, and procedures for enabling and using Site Launcher.

This chapter contains the following sections:

Site Launcher Overview

To minimize your effort in creating new sites, Content Server provides a site-replication utility called "Site Launcher." This utility is designed not for backing up CM sites, but for spinning them off.

For example, your management system hosts a dedicated CM site for a department named "Products." A new department named "Services" has been recently established and must be quickly introduced to the public. "Services" is similar to "Products" in size and structure, although its members are public relations specialists rather than advertising staff. Instead of creating a "Services" CM site from scratch, you can replicate the "Products" CM site directly on the management system, and modify the replicate to accommodate the newly established department.

Site Launcher replicates source sites directly on their native Content Server system, re-using the existing database schema. The sites are replicated quickly and easily, without the need for coding. However, while replication itself is a quick and straightforward procedure, it does require preparation and follow-up on the part of the administrator. How much, depends on the content management needs.

The rest of this chapter provides the steps that you need to follow in order to successfully replicate sites for use in any content management model—1:1, 1:*n*, and *x*:*n*, described in "Content Management Models," on page 30.

Note

Site replication can be carried out only by the general administrator. Site and workflow administrators cannot replicate the sites they manage, as they have no access to Site Launcher.

Preparing for Replication

Before attempting to replicate a site, you need to consider several recommendations regarding the nature of the site and its replicate. You must also make decisions about which components to copy or share, and finally ensure that the source site meets system requirements for replication. This section outlines our recommendations, your options, and the system requirements.

Ensuring the Source Site Meets Replication Requirements

Site Launcher can be used to replicate almost any CM site: small, large, functional, incomplete, independent of other sites, and overlapping other sites by the sharing of components.

However, to use Site Launcher most effectively, it is best for you to start with a site that is small, functional, and similar to the site that you need to have. In general, the source site should be a skeletal one, meaning that it has structure and design, but little content. These characteristics, beyond ensuring a tractable replicate, will help you conserve resources and minimize replication time.

In addition, make sure that the source site resides on the management system. (If you need to first create a replicable source site, follow the steps in "Site Configuration Steps," on page 55.)

Planning the New Site

This section outlines your site replication options and the system requirements.

Copying vs. Sharing

When designating a source site, you must decide whether to copy or share certain site components, and therefore, determine from the start, how the new site will function in relation to its source site—as a duplicate or a subset; as an independent site or an overlapping site. (On a bigger scale, you will also need to determine how the new site will function in relation to other sites in your content management model.)

The table below shows that when the source site is being replicated, most if its components are shared by default. Other components are shared or copied at your discretion. Users are neither copied nor shared.

Source Site Component	Replication method
asset subtypes	Shared
asset types	Shared
assets	Either copied or shared
associations	Shared
CS-Desktop and CS-DocLink configuration	Copied
publishing destinations	Shared
roles	Shared
start menu items	Shared
templates (design assets)	Either copied or shared
tree tabs	Shared
users	Neither shared nor copied
workflow processes	Shared

Note that copying and sharing of assets is selective at the asset type level, but not at the asset level—when you select an asset type to be copied (or shared), *all* assets of that type are copied (or shared). The same holds for templates (design assets).

Concerning assets and templates, your specific tasks are to determine:

- Which asset types must have their assets copied or shared.
- Whether template assets must be copied or shared.
- Whether assets will be previewed on the new site as formatted content.

To determine whether assets and templates must be copied or shared, use the following guidelines:

- Sharing an asset does not require you to share the template that renders the asset. The template can be copied.

  Whether you share or copy an asset depends on how the rendering logic in the template treats asset names. For example, template logic can be written in such a way as to assume either shared templates (constructed with explicit template names in the `render:satellitepage` and `render:callelement` tags) or copied templates (constructed with a template name using `Variables.site` prepended to the name in the `render:satellitepage` and `callelement` tags).

  The same reasoning applies to other asset types. If in your rendering logic you use asset names such as flex attribute name, or image name, then the template logic will assume that the asset type is either shared (explicit name) or copied (constructed name), and you need to designate the source site accordingly.

  If you decide to copy template assets, make sure that the template assets do not contain hard-coded template names, and a template is not named in the “Default Values” list of any start menu item.

- When assets are shared to a new site by Site Launcher, their data is not changed. The single exception is Template assets. A Template asset maintains a SiteCatalog entry for each site to which it is shared. Therefore, when a template is shared to a new site, the template’s list of SiteCatalog entries is updated (the new site is added to the list of SiteCatalog entries).
- Because data in a shared asset is not changed, asset references of that asset also remain unchanged. Therefore, if assets with asset references are to be shared, the asset references must also be shared. For example, if you have Article assets that have references to ImageFile assets, you may
  - share both asset types,
  - copy both asset types, and
  - copy the Articles and share the ImageFiles.

  However, you must not share the Article and copy the ImageFiles. Doing so will cause errors to be displayed when a copy of the site is launched.

- Start menus are shared among the source site and its replicates. Therefore, if a workflow process is set in the source site's start menu, that workflow will be set in the replicate sites. When users in the new site invoke the start menu item to create an asset, the workflow will take effect.

Your decisions in the planning stage are governed by the specifics of your installation as well as your own methods of managing the Content Server environment. The site that you create by replication must ultimately fit in to the content management model that is being (or has been) established on the management system. For example, if you have central sites and centrally managed sites, you need to determine how your new site will function in relation to them. For information about site modeling, see “Content Management Models,” on page 30.

Naming Assets

Assets that are copied to the new site are named algorithmically, by use of prefixes as follows:

- The prefix for the source is specified when you designate a site as a site launcher source site and is, by default, the name of the site.
- The prefix for the new site is specified in Site Launcher itself and is, by default, the name of the new site. When specifying a prefix, make sure it is short.
- If the source asset has a prefix, or the asset type requires a unique name, then a prefix will be used in the name of the new asset. If the source asset has a prefix, the new asset will have the new prefix in place of the old prefix.
- If the source asset does not have a prefix, then the prefix is prepended to the asset name to make the new asset name.

Planning Users

Given that users are not copied or shared to the new site during replication, you will have to manually add them to the new site.

If the users will be different from those on the source site, you will need to create new user accounts. In the process, consider the users' functions on the new site, the roles they will need, their access to start menu items, workflows, tree tabs, and so on.

Bear in mind that the original associations among components are preserved on the new site. If you redefine users' functions, you may have to redefine their associations with other components on the site. Accounting for new associations in the planning stages will help you to assess your post-replication workload and create a functional site that also meets your expectations.

Choosing the Replication Time

Because normal operations affect site replication, it is best to run Site Launcher when the source site is not being actively edited. Otherwise, the replicate site might not match the source site. Also, make sure that prior to replication, all assets in the source site are checked in to the database. Otherwise, if revision tracking is enabled and an asset to be shared is checked out, the sharing of the asset will fail.

Site Replication Steps

This section shows you how to enable and use Site Launcher. The procedure consists of three basic steps:

I. Ensure that Replication Requirements Are Satisfied

II. Enable the Source Site for Site Launcher, on page 184

III. Replicate the Source Site, on page 185

The steps are given in detail in the rest of this chapter.

I. Ensure that Replication Requirements Are Satisfied

1. Prepare for replication by ensuring that source sites and planned sites meet the requirements. For more information, see “Preparing for Replication,” on page 180.
2. Go to section “II. Enable the Source Site for Site Launcher” to enable the source site.

II. Enable the Source Site for Site Launcher

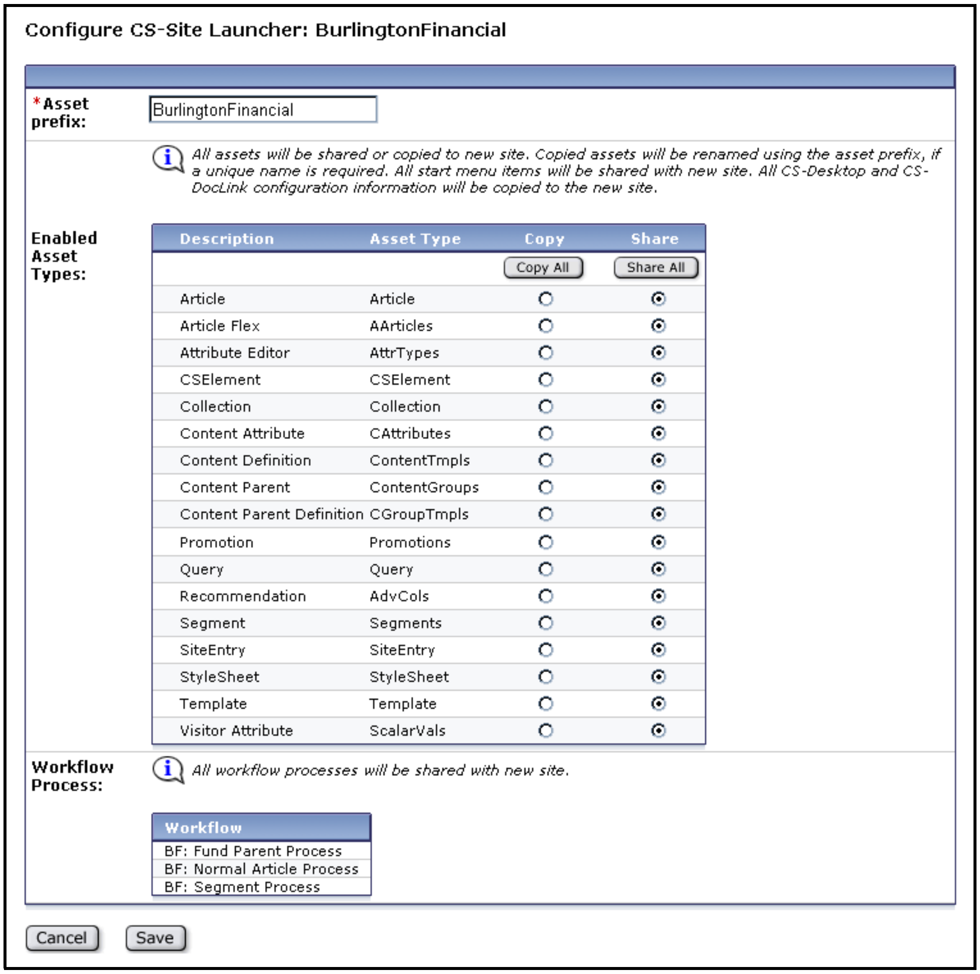
In order to replicate a source site, you must first configure Site Launcher.

To configure Site Launcher

1. Make sure that you have completed the steps shown above (“I. Ensure that Replication Requirements Are Satisfied”).
2. When you complete the steps in this section, be prepared to follow up immediately with the steps in the next section, “III. Replicate the Source Site.” Immediate follow-up ensures that the condition of the source site has not changed.
3. In the **Admin** tab, expand **Sites** and double-click the site you want to replicate. This is your source site.
4. In the site’s “Inspect” form, click **Configure CS-Site Launcher for this site** (at the bottom of the form).
5. In the “Configure CS-Site Launcher” form, do the following:
   a. If you want to copy assets, go to the **Asset prefix** field and either enter a prefix for the assets, or accept the default (the name of the source site).
   b. In the “Enabled Asset Types” panel, locate the asset types whose assets you want to copy or share, and click their **Copy** or **Share** radio buttons.

> **Note**
>
> If you want to copy or share all assets of all types, click the **Copy All** or **Share All** button.

c. Click the **Save** button to save your configuration options.



Configure CS-Site Launcher: BurlingtonFinancial

*Asset prefix: BurlingtonFinancial

All assets will be shared or copied to new site. Copied assets will be renamed using the asset prefix, if a unique name is required. All start menu items will be shared with new site. All CS-Desktop and CS-DocLink configuration information will be copied to the new site.

Enabled Asset Types:

Description	Asset Type	Copy	Share
		Copy All	Share All
Article	Article	○	●
Article Flex	AArticles	○	●
Attribute Editor	AttrTypes	○	●
CSElement	CSElement	○	●
Collection	Collection	○	●
Content Attribute	CAttributes	○	●
Content Definition	ContentTmpls	○	●
Content Parent	ContentGroups	○	●
Content Parent Definition	CGroupTmpls	○	●
Promotion	Promotions	○	●
Query	Query	○	●
Recommendation	AdvCols	○	●
Segment	Segments	○	●
SiteEntry	SiteEntry	○	●
StyleSheet	StyleSheet	○	●
Template	Template	○	●
Visitor Attribute	ScalarVals	○	●

Workflow Process: *All workflow processes will be shared with new site.*

Workflow
BF: Fund Parent Process
BF: Normal Article Process
BF: Segment Process

6. Go to the next section, "III. Replicate the Source Site," to complete the site replication procedure.

III. Replicate the Source Site

Once you have enabled the source site for replication, you can use Site Launcher to create as many copies of the source site as you wish.

To replicate the source site

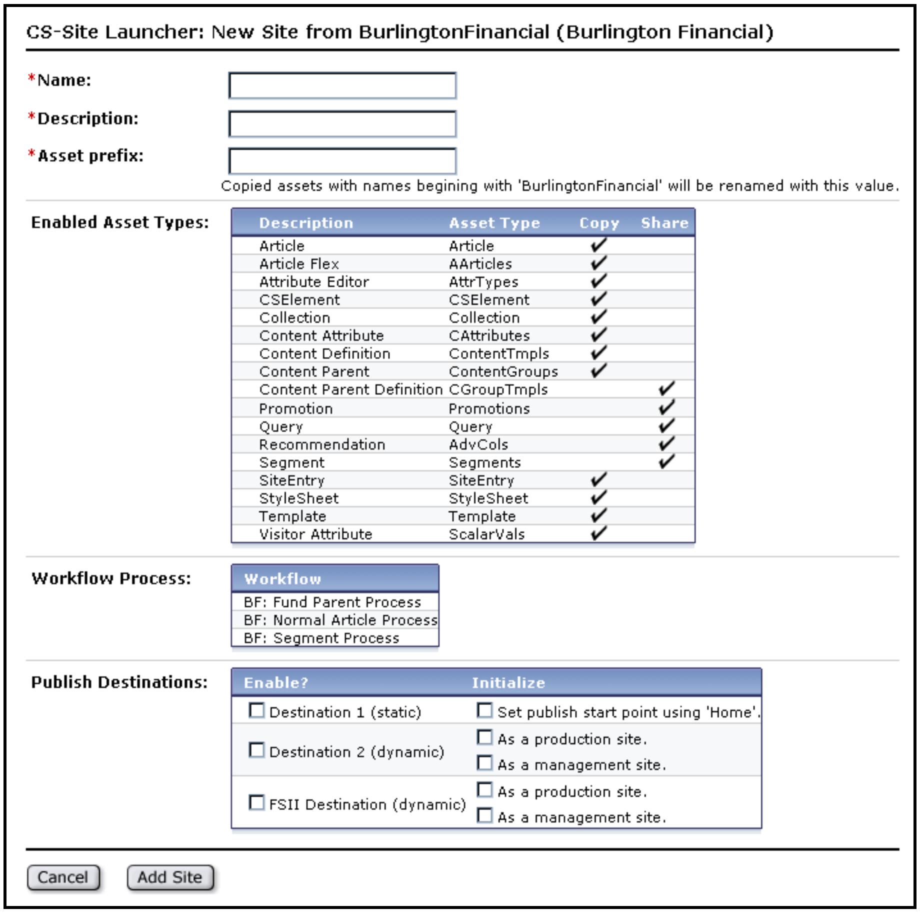
1. Make sure that you have completed the steps above ("I. Ensure that Replication Requirements Are Satisfied," and "II. Enable the Source Site for Site Launcher").
2. At the bottom of the "Inspect" form, click **Launch Copy**.
3. In the "Site Launcher" form, do the following:
   a. In the **Name** field, enter the name for the new site.
   b. In the **Description** field, enter the description for the new site.

c. In the “Publish Destinations” area:

1) Select the publish destinations you would like to be available for the new site.

2) Initialize the destinations by selecting the checkboxes, as necessary.

d. Click **Add Site** to copy the site.



CS-Site Launcher: New Site from BurlingtonFinancial (Burlington Financial)

*Name:	
*Description:	
*Asset prefix:	

Copied assets with names begining with 'BurlingtonFinancial' will be renamed with this value.

Enabled Asset Types:

Description	Asset Type	Copy	Share
Article	Article	✓	
Article Flex	AArticles	✓	
Attribute Editor	AttrTypes	✓	
CSElement	CSElement	✓	
Collection	Collection	✓	
Content Attribute	CAttributes	✓	
Content Definition	ContentTmpls	✓	
Content Parent	ContentGroups	✓	
Content Parent Definition	CGroupTmpls		✓
Promotion	Promotions		✓
Query	Query		✓
Recommendation	AdvCols		✓
Segment	Segments		✓
SiteEntry	SiteEntry	✓	
StyleSheet	StyleSheet	✓	
Template	Template	✓	
Visitor Attribute	ScalarVals	✓	

Workflow Process:

Workflow
BF: Fund Parent Process
BF: Normal Article Process
BF: Segment Process

Publish Destinations:

Enable?	Initialize
□ Destination 1 (static)	□ Set publish start point using 'Home'.
□ Destination 2 (dynamic)	□ As a production site. □ As a management site.
□ FSII Destination (dynamic)	□ As a production site. □ As a management site.

After the new site is created, Content Server displays a summary that indicates:

- Which assets were copied or shared. For each asset that is shared, Content Server adds a row in the `AssetPublication` table, indicating the site id (publication id) of the new site. For each asset that is copied, Content Server enters the copy into the table where the original asset is stored.
- The number of start menu items, workflow processes, and publish destinations that were shared with the new site.

4. At this point, you need to add users and otherwise ensure that the new site is properly configured. For instructions and guidelines, see the next section, “Post-Replication Tasks and Guidelines.”

Post-Replication Tasks and Guidelines

When the new site is established, you need to complete its configuration by completing the following steps:

1. Add existing users to the new site, or establish new users, depending on how you planned the new site.
   - To add existing users, follow instructions in "Granting Users Access to a Site (Assigning Roles to Users)," on page 98.
   - To establish new users, follow the guidelines in Chapter 3, "Site Configuration Guidelines."
2. If workflow processes exist, make sure that the correct asset types and roles are associated with the processes through Start Menu items.
3. Test the new site.
4. If necessary, create a site administrator. Be sure to assign the xceladmin ACL to the user account and give the user the SiteAdmin role for the new site.

Part 3
Managing the Publishing System

This part describes Content Server's publishing system, the publishing methods it supports, the approval system which validates content intended for publication. This part also contains procedures for configuring and managing the supported publishing methods.

This part contains the following chapters:

Chapter 11

Publishing with Content Server

Content is made available to the visitors of your online site by copying from the management system to the delivery system. Copying content from one system to another is called **publishing**.

The Content Server publishing system supports three publishing methods: static, dynamic, and Export to XML. Content Server also provides an approval system that determines which content gets published, a scheduling function that enables you to set the publication time, and a utility that allows you to configure site-specific publishing destinations.

This chapter describes the publishing and approval systems. It contains the following sections:

Overview

Before assets can be published, either you or Content Server must determine the following information:

- Which **publishing method** Content Server should use: static, dynamic, or Export Assets to XML.

  As an administrator, you specify the publishing method for the destination when you configure the publishing destination. For more information, see "Publishing Methods."

  Note

  In Content Server terms, static publishing is called "Export to Disk." Dynamic publishing is called "Mirror to Server." This guide uses the terms "static" and "dynamic."

- To which **publishing destination** Content Server should publish during a given session.

  As an administrator, you configure the publishing destinations for your system. For more information, see "Publishing Destinations," on page 195.

- Which assets have been approved and are ready to be published to the current destination. The **approval system** determines this information.

  When an asset is deemed ready for publication, a content provider marks it as "approved" for a specific publishing destination. The approval system validates the "approved" label to determine whether publishing the asset is likely to create broken links on the live site. If the potential for broken links exists, the asset is held back from the publishing session until its dependencies on other assets are resolved by the user approving those assets, as well. If the approved asset either has no dependencies or its dependencies are satisfied, the approval system releases the asset to the publishing system.

  The approval system's process is complex and varies from one publishing method to another. For detailed information about the workings of the approval system, see Chapter 12, "The Approval System" and Chapter 13, "Various Topics in Static Publishing and the Approval Process."

- When assets should be published.

  As an administrator, you set up the **publishing schedule**. The publishing process runs as a batch process that you schedule to occur at regularly-occurring intervals. On an as-needed basis, you can also override the schedule and publish on demand. For more information about publishing schedules, see "The Publishing Schedule," on page 196.

During a publishing session, the session information is recorded in log files. You can monitor the session through the **Publish Console** to determine both the publishing history and the status of currently running publishing sessions. You can also use your browser to complete other administrative tasks, as the publishing process runs in the background. For information about the events that occur during a publishing session, see Chapter 11, "What Happens During a Publishing Session?" For information on how to obtain publishing information, see Chapter 11, "Obtaining Information About a Publishing Session."

Publishing Methods

This section describes the publishing methods that Content Server supports. Figure 1, on page 194 summarizes the publishing methods.

- For **static publishing**, the delivery system is a web server. Publishing to this type of system is done as follows:

  > Approved assets in the CM system database are rendered by templates into HTML files. The files are saved to a file system and subsequently published to the web server by an administrator using a transfer protocol, such as FTP.

  When published content is requested by site visitors, the HTML files are served as pages to the browser.

- For **dynamic** publishing, the delivery system is a Content Server system. Publishing to this type of system is done as follows:

  > Approved assets and their database tables are mirrored from the CM system database to the delivery system database. Throughout the publishing session, the publishing system communicates and cooperates with the **CacheManager** on the delivery system. The CacheManager is a Content Server servlet that manages a system's page cache. CacheManager ensures caching of the pagelets or pages that refer to the assets which will be mirrored. After the publishing session concludes, CacheManager generates those pages again to display the updated content, and caches the new pages and pagelets.

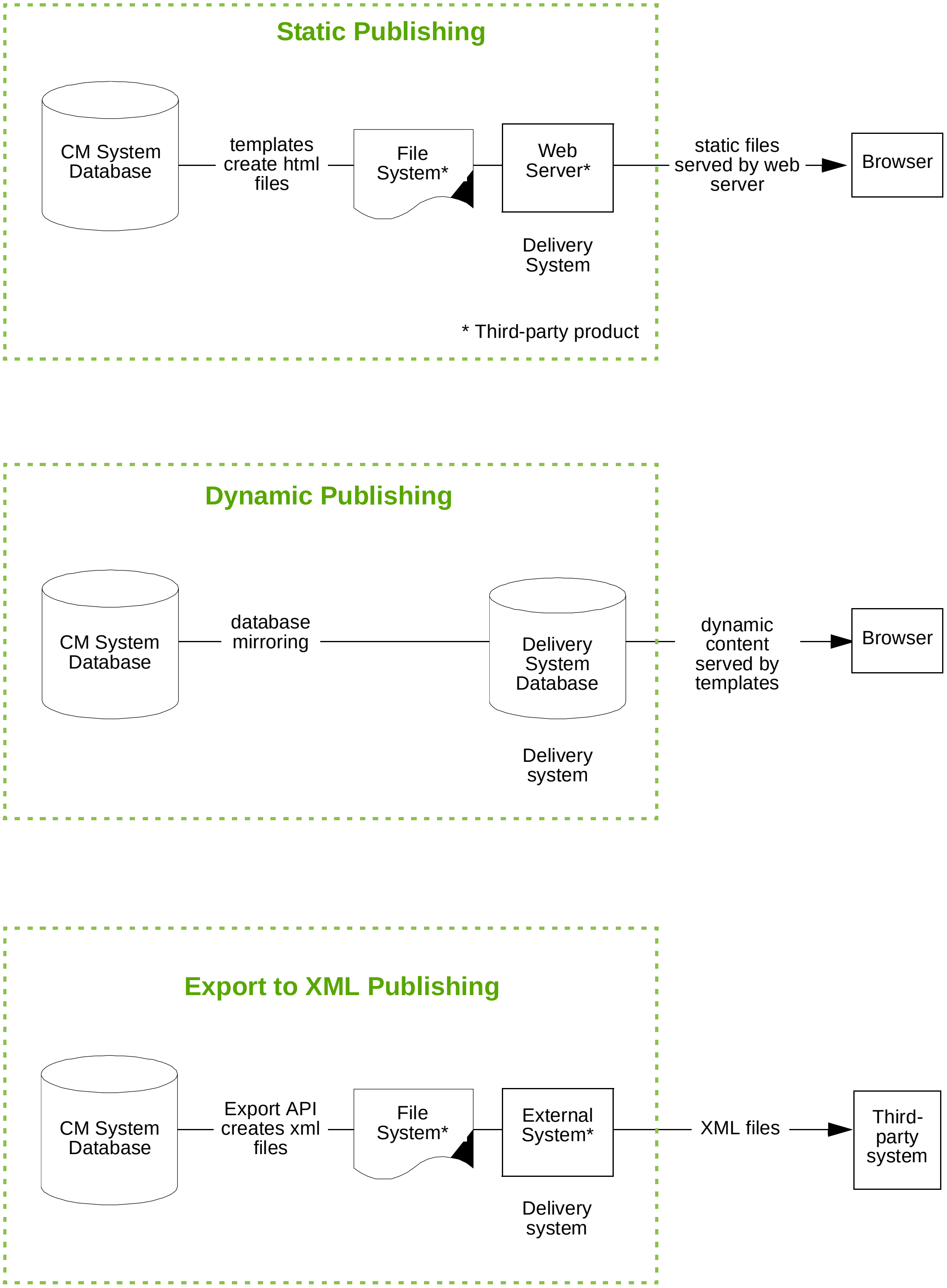
  When published content is requested by site visitors, it is provided on-the-fly. The content is drawn from the delivery system database by templates and served to the browser, where it is finally rendered as pages.

- For **Export Assets to XML**, the delivery system is a database or an external system running an application other than Content Server. The publishing method is a data transformation method that outputs XML files. Rather than creating pages that are ready to be displayed by a web server, this publishing method uses the Export API to create one XML file for each approved asset.

These chapters describe each publishing method in detail:

- Chapter 13, "Various Topics in Static Publishing and the Approval Process"
- Chapter 14, "The Static Publishing Process"
- Chapter 15, "The Dynamic Publishing Process"
- Chapter 16, "The Export Assets to XML Publishing Process"

Figure 1: Content Server’s Publishing Methods



Publishing Destinations

A publishing destination is either the Content Server database to which you are mirroring content, or a directory to which you are exporting. A publishing destination is expressed as a named object that defines the following parameters:

- A publishing method (dynamic, static, or Export Assets to XML)
- A location
  - For static publishing and Export Assets to XML, the location is the base directory to which the resulting HTML or XML files are saved after the publishing system converts the approved assets into files. You set this directory with the `cs.pgexportfolder` property in the `futuretense.ini` file.
  - For dynamic publishing, the destination is the **server name in URL format** (which includes the port number if it is anything other than 80) of the server that is supposed to deliver that content. (You set the server name in the **Destination address** field of the “Destination” form.)
- Various publishing arguments, depending on which publishing method you are using. For example, if you are publishing statically, you could specify a URL prefix for your internal links (HREFs).
  - For information about static publishing arguments, see “Static Publishing Arguments,” on page 242.
  - For information about dynamic publishing arguments, see “Dynamic Publishing Arguments,” on page 262.
  - For information about Export Assets to XML publishing arguments, see “Publishing Arguments for Export Assets to XML,” on page 278.
- The names of CM sites whose assets can be published to the destination.

For any publishing method, you can configure more than one publishing destination for your CS system. Depending on how your online sites are designed by the developers, you may need to publish both static and dynamic content. For this kind of dual implementation, consult with your developers to determine how to configure your publishing destinations.

In this document, the following terms are used when discussing publishing destinations and configuration:

- **Source.** The Content Server database that serves as the source for a publishing session. Because you can mirror assets from any CS system to any other CS system, the source is not always the content management system.
- **Destination.** Either the Content Server database that you are mirroring to or the directory that you are exporting to.

For information about creating publishing destinations, see “Step 3: Configure an Export Destination,” on page 253 and “Step 5: Create a Mirror Destination,” on page 267.

The Approval System

When assets are approved for publication on a specific destination, Content Server verifies that each asset is ready to be published by invoking the approval system. The approval system runs a pre-publication process that protects the online site against the possibility of broken links and outdated content. In this process, the approval system determines answers to the following questions:

1. Which assets are deemed to be ready for publishing—that is, which assets are marked by the user as approved?
2. Do the approved assets have dependencies? That is, must any other assets also accompany those approved assets? or, must any other assets also exist on the destination to ensure that all the links on the site will work correctly?

   For example, the Burlington Financial sample site has article assets that refer to image files. An approved article with an unapproved associated image file cannot be published until the image file is also approved.

Depending on the answers to these questions, the approval system either allows the publishing system to publish approved assets at the next publishing session, or it stalls the publication of assets that fail the dependency test.

> **Note**
>
> - In static publishing, the choice of templates can cause assets to bypass the approval system. For more information about the approval system in static publishing, see Chapter 13, "Various Topics in Static Publishing and the Approval Process."
> - It is typical to build an approval step into your workflow processes. For information about workflow, see Chapter 9, "Creating and Managing Workflow Processes."
> - For information about how to approve assets, see the chapter on publishing in the *Content Server User's Guide*.

Detailed information about the workings of the approval system is given in Chapter 12, "The Approval System" and Chapter 13, "Various Topics in Static Publishing and the Approval Process."

The Publishing Schedule

A publishing session (no matter what the publishing method) runs as a background, batch process called a **publish event**. As the administrator, you configure the schedule for publishing events.

You can schedule publishing events to occur daily, weekly, hourly, every 15 minutes, or in various combinations of these time increments. When scheduling multiple publishing sessions, bear in mind several rules:

- Publishing to a given destination must be done serially.

  Because each publishing session is a background event, a given destination can support only a single publishing session at a time. If publishing to a specific destination is still in progress when the next event for that destination is scheduled, the second event attempts to run and then fails, reporting that a publishing session to that destination is already underway.

Similarly, multiple sources must not be configured to publish simultaneously to the same destination. Doing so will cause publishing errors and problems with data integrity.

- Never schedule a publishing session to a destination for a time when the destination system could be publishing to another destination.

  For example, you publish from the development system to the management system, and from the management system to the delivery system. Do not schedule a publishing session from the development system to the management system while the management system is publishing to the delivery system.
- Publishing to multiple destinations can be done simultaneously. You can publish to multiple destinations at the same time by setting up events for the destinations and selecting the same time for them.

Because a publishing event completes database transactions, the publishing feature must have a user account specified for it. This user is called the **batch user** and you use the `xcelerate.batchuser` and `xcelerate.batchpass` properties in the `futuretense_xcel.ini` file to identify the batch user account for your CS system.

> **Note**
>
> Publishing also requires a mirror user, different from the batch user. For more information, see “Users and Dynamic Publishing,” on page 262.

What Happens During a Publishing Session?

When the publishing system begins publishing approved assets, Content Server does the following:

- Creates a publishing session by adding a row to the `PubSession` table and assigns the session a unique ID, called the PubSession ID.
- Runs a query to gather all the assets that are approved and are ready to be published to the destination that the session is publishing to.
- Locks the assets that are returned by the query, and also notifies the CacheManager about these assets, so that they cannot be edited while the publishing session is underway.
- Invokes the appropriate element for the publishing method (static or dynamic), passing it the list of assets that should be published.

Then, those assets are either rendered into files or mirrored to a destination database, depending on the publishing method in use. For information about what the static, dynamic, and Export Assets to XML publishing methods do with the list of approved assets, see the following chapters:

- Chapter 14, “The Static Publishing Process”
- Chapter 15, “The Dynamic Publishing Process”
- Chapter 16, “The Export Assets to XML Publishing Process”

When the publishing session concludes, the publishing system notifies the approval system of which assets were published and then it updates the page cache.

Obtaining Information About a Publishing Session

Content Server writes information about each publishing session to log files and to several tables in the Content Server database. Much of this information is displayed in the **Publishing Console** as the publishing history for a session. To display the **Publishing Console**, click the **Publish** button located in the top button bar in the main window. See also “Troubleshooting,” on page 295.

Note

You must have the `xcelpublish` ACL to view the **Publish Console**.

Chapter 12

The Approval System

This chapter explains the basic workings of Content Server's approval system. It does not contain procedures on performing specific operations. Rather, its purpose is to help the administrator gain a basic understanding of the approval system's process, which is both complex and highly interactive, especially in the case of static publishing.

Each section of this chapter is written to be as self-contained as possible, with many examples to illustrate the concepts. For continuity, users can read the sections sequentially, and for reference, turn to the section of interest. As this chapter progresses, it builds on the introductory material (especially approval-system terms and definitions), and ends with a comprehensive picture of how the approval system performs its function.

Note

Readers who would like a summary of the approval process can skip to the section "Putting It All Together," on page 221.

This chapter contains the following sections:

- Overview
- A Brief Look at Dependency Analysis
- Terms and Definitions
- Rules for Dependency Analysis
- Putting It All Together
- Reference: Approval States

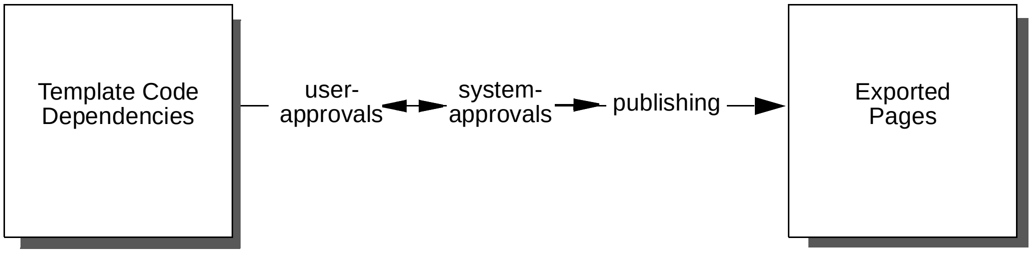
Overview

Before a publishing session can begin, the user is required to approve assets for the publishing destination. This task is far more intricate than it appears to be. Enterprise sites host enormous numbers of assets, making it impossible for users to track which assets are bound to each other by dependencies and therefore must be approved together in order to be published. As a result, users typically approve only some of the assets that require approval. The unapproved assets compromise the integrity of the approved assets by causing broken links among them. This is where the approval system takes over.

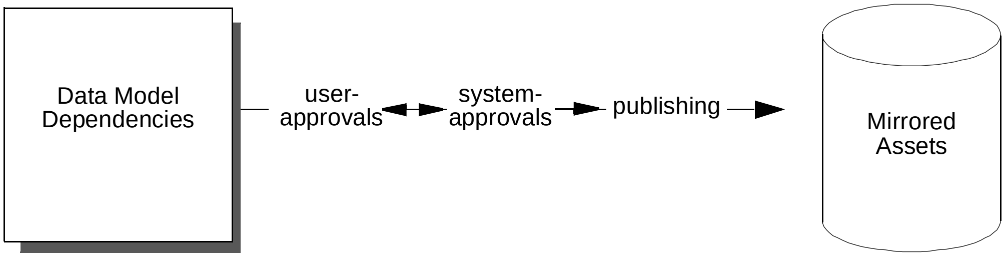
The ultimate purpose of the approval system is to protect the integrity of published content. To this end, the approval system does not automatically allow the publication of user-approved assets. Instead, it evaluates the condition of the user-approved assets in order to determine whether they are indeed ready for publication. If necessary, the approval system then assists the user in approving the supporting assets.

During its operation, the approval system ensures that asset dependencies, encoded in templates or the data model, are kept intact among approved assets, so the dependencies can be reproduced on the publishing destination when the approved assets are published. If an approved set of assets is free of broken links, it is likely to be free of broken links when published. If an approved set of assets upholds version matching, it is likely to uphold version-matched content when published.

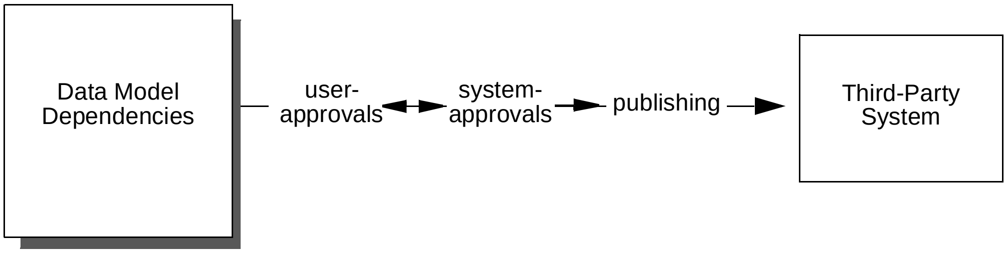
Static Publishing



Dynamic Publishing



Export to XML



A Brief Look at Dependency Analysis

When analyzing approved assets for broken links and inconsistent versions the approval system performs a dependency test:

1. The approval system determines whether dependencies among *user-approved* assets match dependencies that are specified in either the template code or the data model, taking into consideration the publishing method. As shown in Figure 3, on page 206:
   - In static publishing, the approval system tests dependencies (among approved assets) against template code. The dependencies are determined as the code in the template is evaluated. If a default template has been assigned to assets of the given type, the approval system uses it to determine the dependencies. If there is no default template, the approval system uses the template that is specified for the given asset. (For detailed information about approvals and static publishing, see Chapter 13, "Various Topics in Static Publishing and the Approval Process.")
   - In dynamic publishing and Export to XML, the approval system tests dependencies (among approved assets) against the data model.
2. If the approval system finds that dependencies among user-approved assets are satisfied, it allows the assets to be published and releases them to the publishing system for inclusion in the next publishing session.

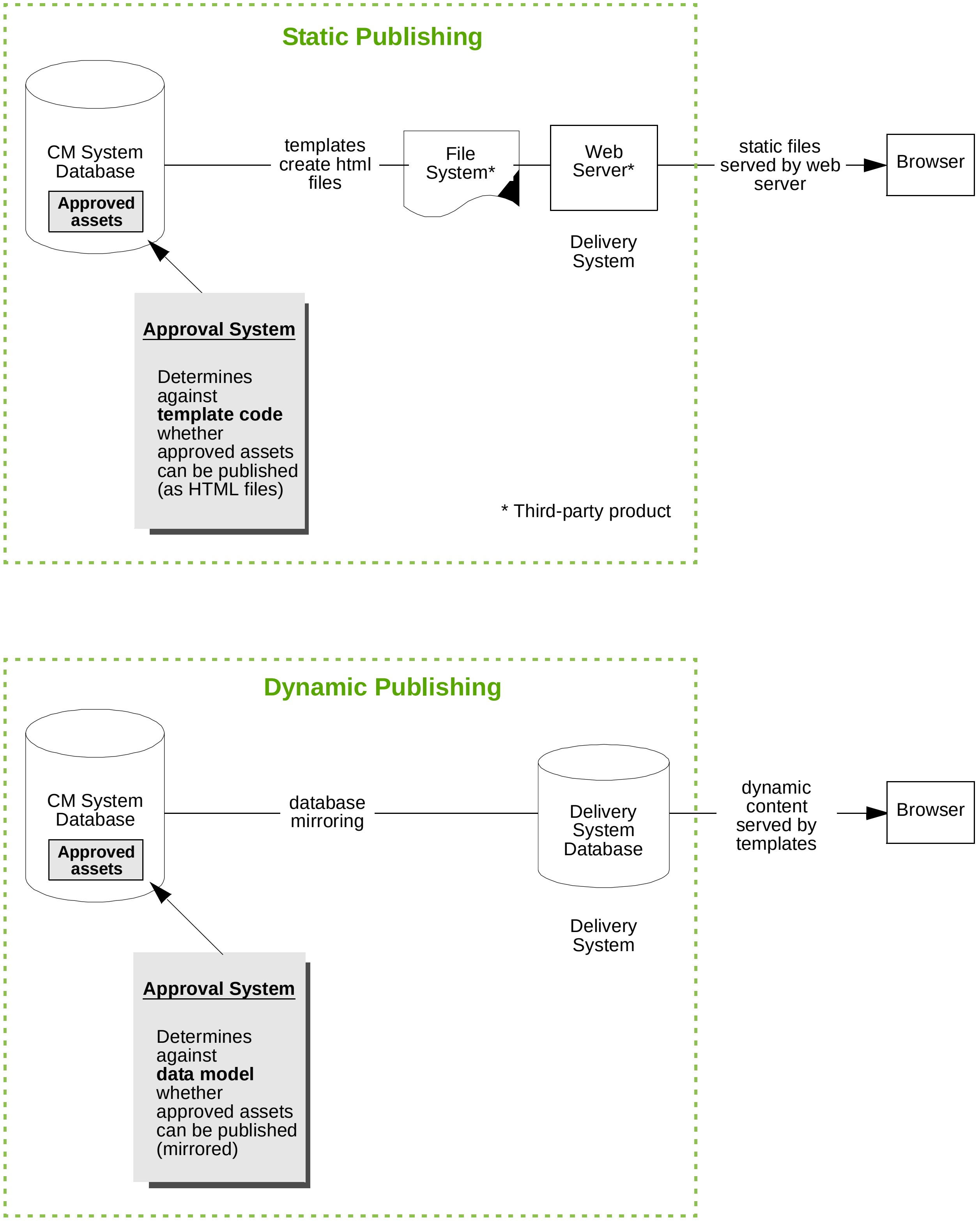
   However, if it finds the dependencies to be breached, the approval system looks for published assets that will satisfy those dependencies at publication time and therefore turns to published content. The approval system scans the `PublishedAssets` table to determine what has been published to the target destination:
   - If an asset requiring approval does not exist on the target destination (and in the correct version, if applicable), the approval system prompts the user to approve the asset.
   - If the asset requiring approval already exists on the target destination (and in the correct version, if applicable), the asset will satisfy the dependency at publication time and therefore does not require approval (and re-publication).

Note

Version matching can be imposed by the CS administrator, in which case user-approved assets must match the versions of published assets in order to satisfy the dependency criteria (which are called `exists` and `exact`).

For dynamic publishing, version matching is specified in the `mwb.conservative.dependencies` property, in `gator.ini`. For static publishing it is specified in the template code. For more information see "Ensuring Version-Matched Content," on page 215.

Figure 2: Approvals and Publishing Methods


Static Publishing
CM System Database
Approved assets
templates create html files
File System*
Web Server*
static files served by web server
Browser
Delivery System
Approval System
Determines against **template code** whether approved assets can be published (as HTML files)
* Third-party product
Dynamic Publishing
CM System Database
Approved assets
database mirroring
Delivery System Database
dynamic content served by templates
Browser
Delivery System
Approval System
Determines against **data model** whether approved assets can be published (mirrored)

3. On completing its analysis of all the user-approved assets, the approval system either approves or disallows the publishing of certain assets:

   The approval system supports fractional publishing rather than "all or none" publishing. Assets that pass the dependency test are released to the publishing system for inclusion in the next publishing session. Assets that fail the dependency test are held back from the publishing system, and the user is informed as to which additional assets also require approval. Dependencies among approved assets are hence the cause of lighter-than-expected publishing sessions.

Dependencies among approved assets have a specific treatment and a name ("approval dependencies"), both of which are critical to the understanding of how the approval system works. Approval dependencies are the main subject of the rest of this chapter. They are described and explained in the section "Terms and Definitions," on page 203, along with the equally important terms, "approvals" and "approval states." (A summary of the approval process can be found in the section "Putting It All Together," on page 221.)

Terms and Definitions

Three especially important terms in the Content Server publishing model are "approval," "approval dependencies," and "approval states." These terms deal with dependencies that are critical to the understanding of how the approval system works. They are also used throughout this guide, as well as the Content Server publishing interface. We recommend that you familiarize yourself with the terms, their definitions, and the supporting examples. For information about these terms, see the following sections:

- Approval: Intent to Publish vs. Permission to Publish
- Approval Dependencies
- Approval States

Approval: Intent to Publish *vs.* Permission to Publish

For an asset to be published, it requires approval first from the user, then from the approval system. To distinguish one type of approval from the other, we use the terms **user-approval** and **system-approval**.

- The difference between user-approval of assets and system-approval of assets is the following:

  Whereas the user's approval of an asset signals the user's *intent to publish* the asset, the system's approval of the asset grants the publishing system *permission to publish* the asset.

  The user's approval of an asset is equivalent to submitting the asset to the approval system for a dependency test. The system's approval of a user-approved asset is equivalent to (1) confirming that the user-approved assets pass the dependency test, and (2) releasing the asset to the publishing system for publication in the next publishing session.

- The approval system never approves assets before the user has had a chance to do so:

  When we say that an asset is system-approved, it is understood that the user has already approved the asset.

- When a user approves an asset, it is always to a given publishing destination.

  When we say that an asset is user-approved, the publishing destination is understood (and explicitly identified, if necessary).

- System-approvals are conditional:

  When approving assets, the user can assign only one state to them, and that is "Approved." The approval system responds by treating the approval as tentative and requiring verification. Only when the approval system completes its dependency test and confirms that the "Approved" state is valid does the system approve (release) the assets for publication.

  If, however, the approval system cannot confirm the "Approved" state, it rejects the state, assigns its own approval state to the assets, informs the user as to which supporting assets require approval, and holds the user-approved assets from publication until the user also approves the supporting assets. The system's approval of a user-approved asset is thus conditional on the user also approving the supporting assets. Conditional approvals are caused by approval dependencies, which are explained in the next section.

- The approval system can unapprove assets:

  The approval system supports an approval queue to handle asset modification events as a way of keeping approval tables up to date. For example, if an asset is approved for publication, but then changed, the approval queue handles this change by unapproving the asset—that is, rejecting it from the publishing queue.

  The approval queue is not accessible to the user. It runs, by default, every 5 minutes. However, when you invoke a feature that uses approval functionality (such as the **Publishing** tab), the approval queue is forced to run before the screen is rendered to keep approval information up to date.

Approval Dependencies

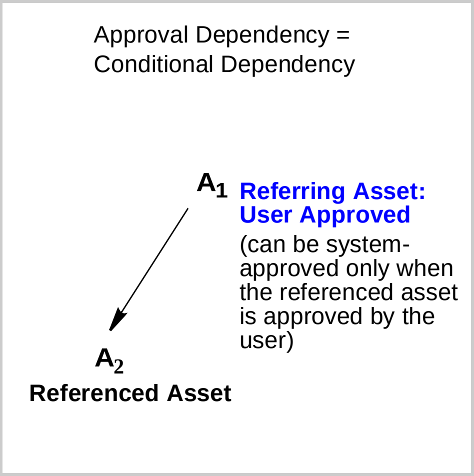
Note

Before reading this section, make sure you understand the distinction between the terms “user-approval” and “system-approval.” For more information, refer to the section “Approval: Intent to Publish vs. Permission to Publish,” on page 203.

An **approval dependency** is a conditional dependency and results in a conditional approval:

Condition 1. An approval dependency is created if the user approves an asset that refers to another asset for content.

Condition 2. When an approval dependency is created, the system’s approval of the referring asset is conditional on the approval of the referenced asset.
In other words, the approval system can approve a referring asset only if the referenced asset is also approved. The referring asset is said to have an approval dependency on the referenced asset.



Note

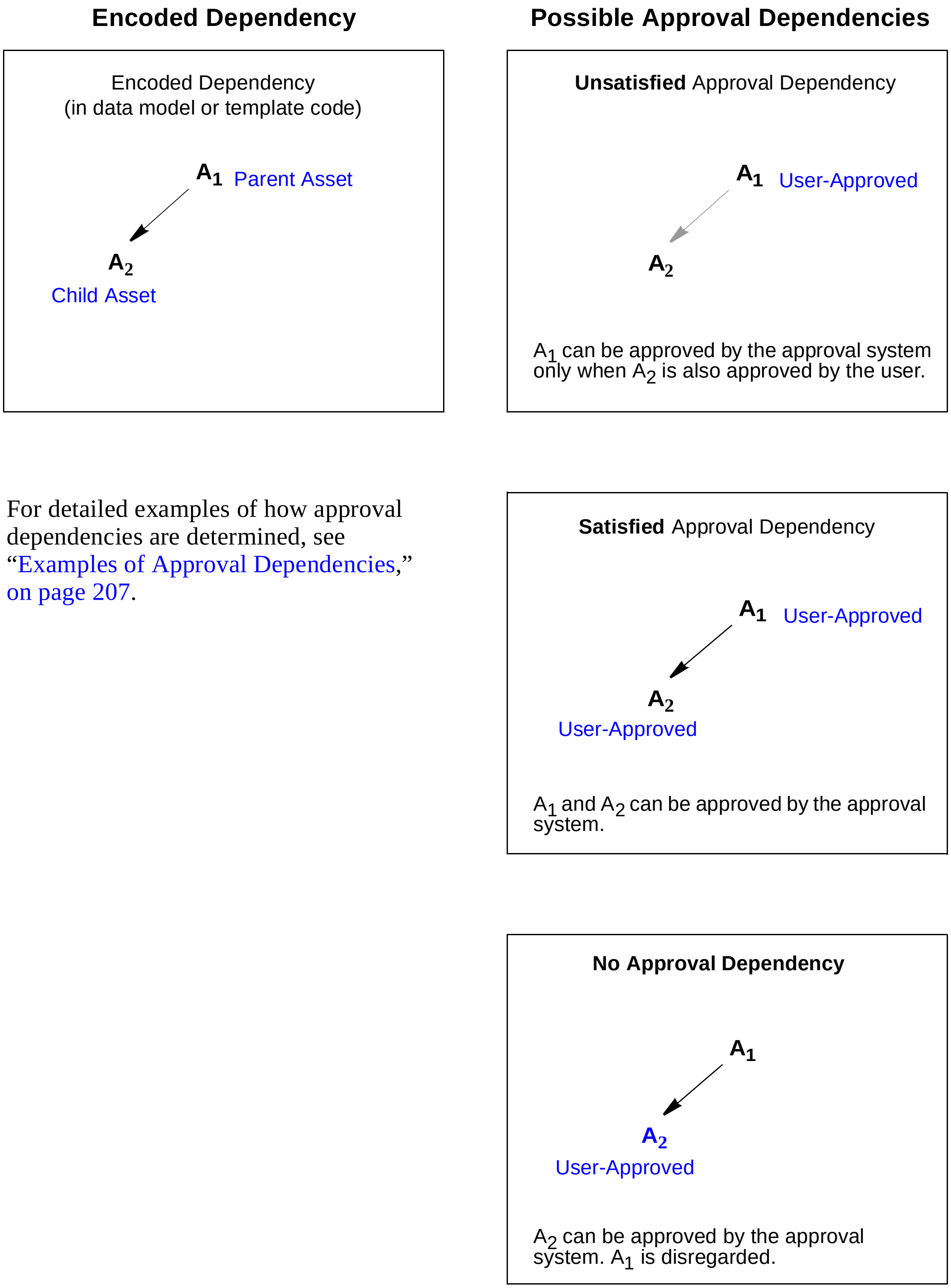
From this point forward, we define “referring asset” to be the parent asset. The “referenced asset” is the child asset. For simplicity, we also assume that parent assets are solely parents, and child assets are solely child assets.

An important implication of approval dependencies is the following:

> While approval dependencies originate in one of two sources (the data model or template code), **the number of approval dependencies is not necessarily equal to the number of dependencies in the source.** As shown in Figure 3, the existence of a dependency in the source can lead to an unsatisfied approval dependency, a satisfied approval dependency, or no approval dependency at approval time. Which outcome is observed depends on two factors:
>
> - Which asset(s) the user has approved.
> - How the approval system treats approval dependencies. The approval system does not simply determine that a dependency exists for a given pair of assets. It explicitly accounts for the direction of the dependency by testing whether the approved asset is a parent or a child. If it determines that a parent asset has been approved, it logs an approval dependency. If it determines that a child asset has been approved, it does not log an approval dependency.

For detailed examples of how approval dependencies are determined, see page 207.

Figure 3: Encoded Dependencies and Approval Dependencies


Encoded Dependency
Possible Approval Dependencies
Encoded Dependency
(in data model or template code)
A_1 Parent Asset
A_2
Child Asset
Unsatisfied Approval Dependency
A_1 User-Approved
A_2
A_1 can be approved by the approval system only when A_2 is also approved by the user.
Satisfied Approval Dependency
A_1 User-Approved
A_2
User-Approved
A_1 and A_2 can be approved by the approval system.
No Approval Dependency
A_1
A_2
User-Approved
A_2 can be approved by the approval system. A_1 is disregarded.

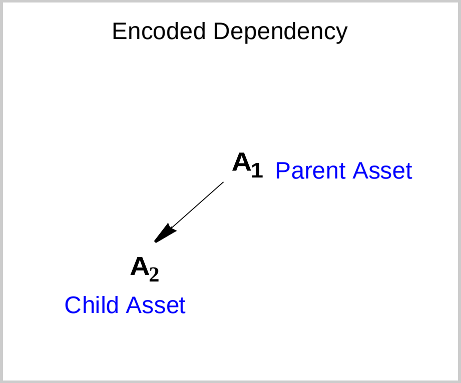
For detailed examples of how approval dependencies are determined, see "Examples of Approval Dependencies," on page 207.

Examples of Approval Dependencies

This example supplements the previous section (especially Figure 3, on page 206), by illustrating in greater depth how approval dependencies are determined.

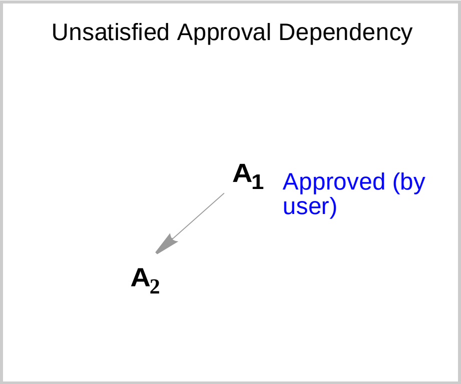
In this example, we have a two-asset data model. Asset A_1 refers to asset A_2 for content. While the data model specifies one dependency ($A_1 \rightarrow A_2$), the user has three different ways to approve the assets and, therefore, create different approval dependencies:

- Unsatisfied approval dependency
- Satisfied approval dependency
- No approval dependency



1. **Unsatisfied approval dependency**

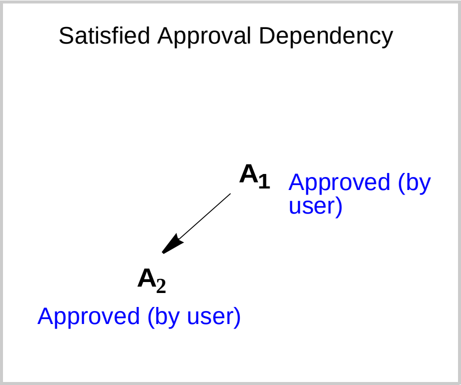
   In this instance, the user approves the parent asset (A_1), but not its child (A_2). The approval system crawls the data model (using the approved asset as a seed), discovers the $A_1 \rightarrow A_2$ dependency, determines the dependency to be unsatisfied (by the user's approval of A_1 alone), classifies the dependency to be an unsatisfied approval dependency, rejects the "Approved" state for A_1, disallows the publication of A_1 until A_2 is also approved, and informs the user that A_2 needs approval.



   In this instance, the system's approval of A_1 is conditional on the user approving A_2. A_1 is said to have an approval dependency on A_2. (Because the approval dependency is unsatisfied, the approval system notifies the user.)

2. **Satisfied approval dependency**

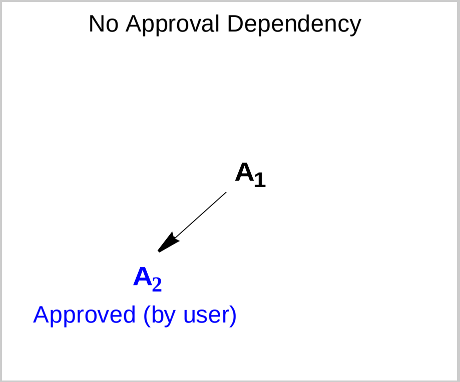
   In this instance, the user approves both the parent asset and its child. The approval system finds the data model dependency to be satisfied (by the user's approval of both A_1 and A_2), classifies the dependency to be a satisfied approval dependency, confirms the "Approved" states, and allows publication of both A_1 and A_2 when the next publishing session begins.



As in the previous instance, the system's approval of A_1 is conditional on the user approving A_2. This time, however, the approval dependency is satisfied. No notice is returned to the user. The asset will be published in the next publishing session.

3. **No approval dependency**

In this instance, the user approves the child asset (A_2), but not its parent (A_1). Here, the approval system does not recognize an approval dependency (because A_2 has no dependency on A_1). The approval system confirms the user's "Approved" state. As a result, A_2 will be published (but A_1 will not).



Summary

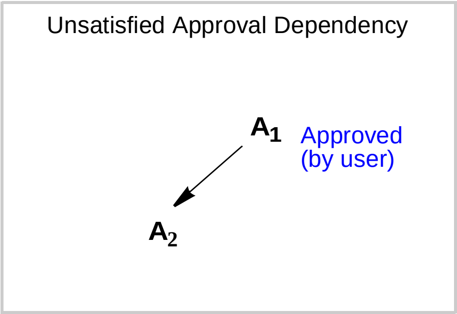
- Whereas data models and template code distinguish between the types of dependencies they specify (for example, named associations, and recommendation associations), the approval system simplifies these dependencies by treating them as parent-child approval dependencies. The parent is the referring asset and depends on its child for content.

  More information about this rule (and other rules regarding dependency analysis) is given later in this chapter. See "Rules for Dependency Analysis," on page 212.
- Whereas data model dependencies and template dependencies are created by the developer to specify how and which assets refer to each other for content, approval dependencies are used by the approval system to flag which user-approved assets can be released for publication and which must be held back from publication; which assets have yet to be approved by the user, which assets have been published, and so on.
- Data model and template code dependencies are fixed for the given data model and template. Approval dependencies can vary, even for a fixed set of assets and a given publishing destination. This is so because the combination of assets approved by the varies from one approval attempt to another.

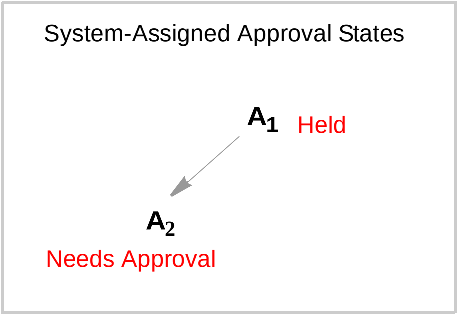
Approval States

When the approval system detects an approval dependency, it must either confirm the user's "Approved" label for each of the assets, or flag the assets with approval states of its own. An **approval state** is a label assigned to user-approved assets by the approval system in order to identify their approval status: why an approved asset cannot be published, whether the asset has been approved and published, whether an asset is eligible for publishing (that is, approved but not yet published because the publishing session has not yet been started), and so on. A complete listing of approval states is given in Table 1, on page 224.

For example, in the figure at the right, the user marked asset A_1 as "Approved." The system, however, treats the approval as tentative. Following a dependency analysis, the system determines that A_1—even though approved— has an unsatisfied approval dependency on A_2 and must be held from publication (until A_2 is also approved).



The system overrides the user's approval of A_1 by assigning A_1 the "Held" approval state. It assigns A_2 the "Needs Approval" state.

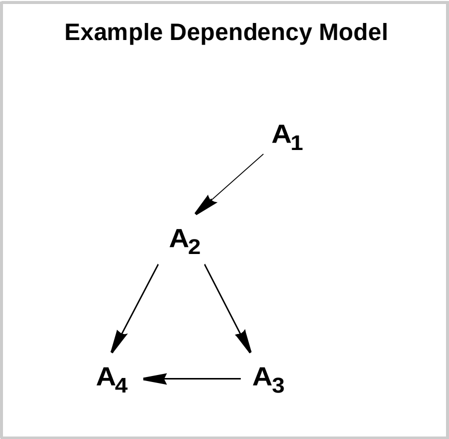


Approval State Scenarios

Even for a fixed set of assets and a given publishing destination, the list of approval states can (and typically does) vary from one approval attempt to another. This is so, because the combination of assets approved by the user tends to vary from one approval attempt to another. To illustrate, we use a simple dependency model and three approval scenarios, all involving one and the same user:

In the figure at the right, each arrow represents an approval dependency. For all four assets to be published, four approval dependencies must be satisfied at publication time:

$A_1 \rightarrow A_2$
$A_2 \rightarrow A_3$
$A_2 \rightarrow A_4$
$A_3 \rightarrow A_4$

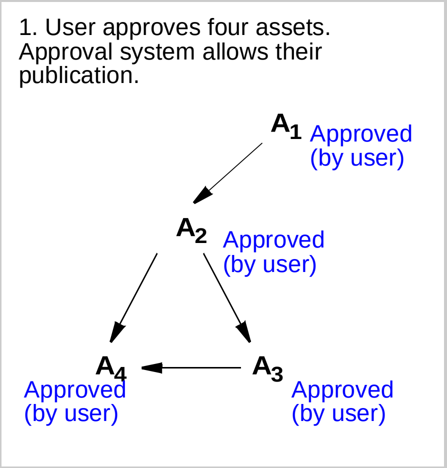


(None of the assets have been previously published.)

Now, consider three scenarios in which the user approves different combinations of assets:

1. In the first (and ideal) attempt, the user approves all four assets, which satisfies all four approval dependencies at once.

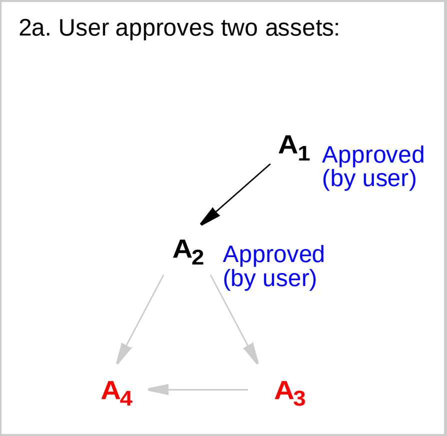
In this scenario, the system logs four approval dependencies, determines they are satisfied, confirms the assets as approved, and allows their publication when the next publishing session begins.



1. User approves four assets. Approval system allows their publication.

2a. In the second scenario, the user approves two assets—A_1 and A_2—which satisfies one of the four approval dependencies. Three approval dependencies remain to be satisfied:

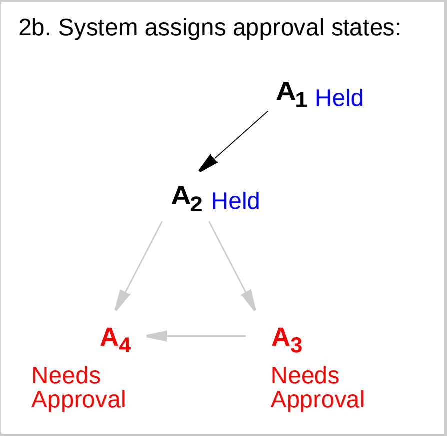
$A_2 \rightarrow A_3$
$A_2 \rightarrow A_4$
$A_3 \rightarrow A_4$



2a. User approves two assets:

2b. The approval system logs three unsatisfied approval dependencies, determines that none of the assets are ready for publication, assigns its own approval states to all the assets, and returns to the user the following list:

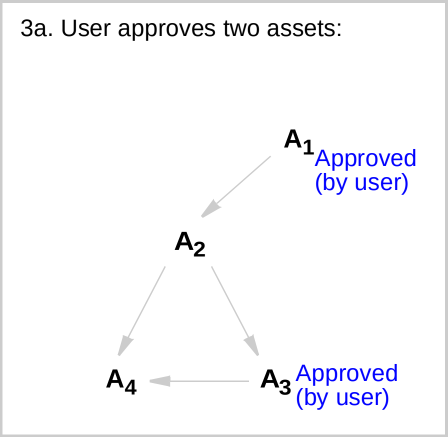
- A_3 and A_4, in the “Needs Approval” state
- A_1 and A_2, in the “Held” state; these assets cannot be published until A_3 and A_4 are approved



2b. System assigns approval states:

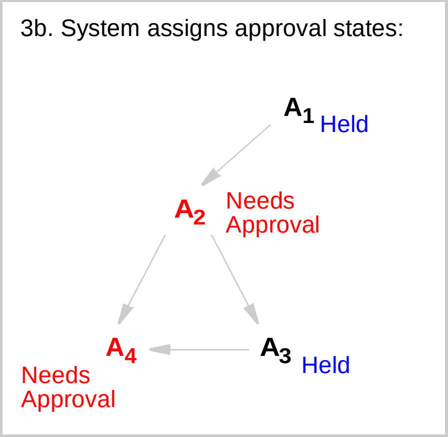
3a. In the third scenario, the user also approves two assets—A_1 and A_3—but this time, satisfies none of the four approval dependencies:

$A_1 \rightarrow A_2$
$A_2 \rightarrow A_3$
$A_2 \rightarrow A_4$
$A_3 \rightarrow A_4$



3b. The approval system logs four unsatisfied approval dependencies, and determines that none of the assets are ready for publication. The approval system returns a list of assets and approval states different from the list in scenario 2. The list identifies:

- A_2 and A_4 in the "Needs Approval" state
- A_1 and A_3 in the "Held" state



Note

If the user had approved a different set of assets, the system would have determined different approval dependencies, and therefore different approval states for the participating assets:

- If the user had approved A_2, the approval system would have determined only three approval dependencies ($A_2 \rightarrow A_3$, $A_2 \rightarrow A_4$, and $A_3 \rightarrow A_4$), even though the data model in our example specifies four dependencies.
- If A_3 were approved, one approval dependency would have been determined ($A_3 \rightarrow A_4$).
- If A_4 were approved, no approval dependencies would have been determined.

For more information as to how approval dependencies are determined, see "Approval Dependencies," on page 205.

Rules for Dependency Analysis

To perform its function, the approval system follows several rules, summarized below and covered in more detail in the sections that follow:

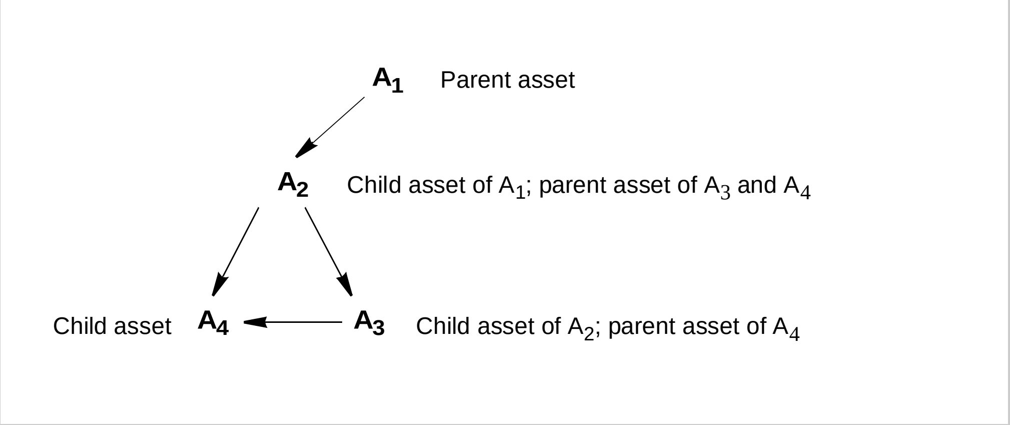
- The approval system simplifies all approval dependencies by treating them as parent-child relationships. For more information, see "Approval Dependencies and Parent-Child Relationships," on page 212.
- In static publishing, the approval system tests approval dependencies against template code. For more information, see "Static Publishing," on page 213.
- In dynamic publishing and Export to XML, the approval system tests approval dependencies against the data model. For more information, see "Dynamic Publishing and Export to XML," on page 215.
- If version matching needs to be controlled, the administrator can set an `exists`, `exact`, or `none` dependency to impose or cancel version matching. For more information, see "Ensuring Version-Matched Content," on page 215.
- If the approval system determines the existence of an unsatisfied approval dependency, it searches for published content to avoid redundant approvals. For more information, see "Evaluating Published Content," on page 220.

Approval Dependencies and Parent-Child Relationships

The approval system simplifies the many types of dependencies that are specified in data models and template code (for example, named associations, unnamed associations, and recommendation associations). Regardless of their origin and type, the approval system classifies all *approval dependencies* as parent-child relationships and adheres to the following rules.

- An asset that refers to another asset is always the parent asset; the referenced asset is the child asset. Figure 4 shows an example of parent-child relationships, in which one asset is solely a parent asset, another is solely a child asset, and the rest are a mixture.

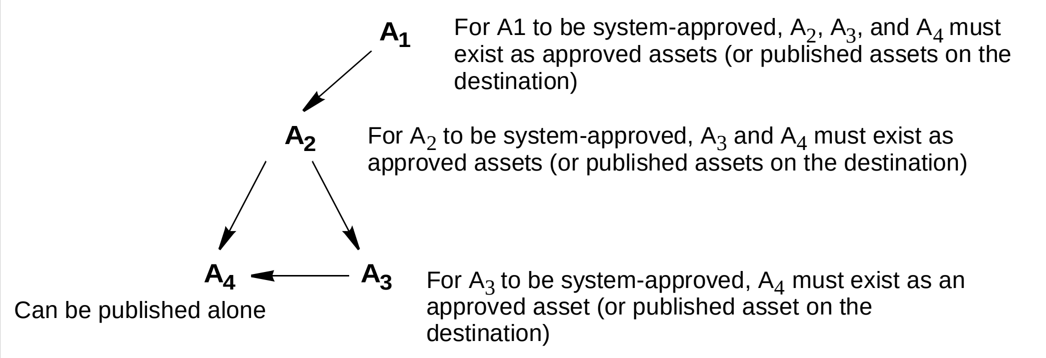
Figure 4: Parent-Child Relationships


A_1
Parent asset
A_2
Child asset of A_1; parent asset of A_3 and A_4
Child asset
A_4
A_3
Child asset of A_2; parent asset of A_4

- A parent asset is dependent on the child asset for content.
- The approval system approves parent assets only if the child assets are either user-approved or they already exist on the publishing destination.
  - If the child assets are also parents, then their child assets must be approved (or must exist on the destination), and so on.
  - If the publishing method is dynamic, version matching can be a requirement (at the CS administrator's discretion), in which case the approval system checks assets for version. For more information about version matching, see "Exists/Exact Dependency Rules," on page 217.
- The approval system approves child assets for publication independently of their parent assets. On the live site, the link from parent to child will not appear to be broken; it simply will not appear. (In this respect, parent data can be inadvertently dropped from the live site.)

Figure 5 displays a hierarchical dependency among four assets to illustrate which assets must be approved concurrently (as a result of approval dependencies), and which can be approved alone.

Figure 5: Approval Dependencies


A_1
For A1 to be system-approved, A_2, A_3, and A_4 must exist as approved assets (or published assets on the destination)
A_2
For A_2 to be system-approved, A_3 and A_4 must exist as approved assets (or published assets on the destination)
A_4
Can be published alone
A_3
For A_3 to be system-approved, A_4 must exist as an approved asset (or published asset on the destination)

Static Publishing

For static publishing, the approval system tests approval dependencies against template code.

> **Note**
>
> Approval templates represent the developer's approval strategy for static publishing. Because approval for static publishing is complex, it is given a separate treatment in Chapter 13, "Various Topics in Static Publishing and the Approval Process." The section below introduces the basic concepts.

Evaluating Approval Dependencies

For both flex and basic assets that are statically published, there are two types of approval dependencies: template and reference.

Template dependencies are strictly independent of the data model, regardless of how (or whether) the assets are associated in the data model. For example, your data model can specify an article-image association. In dynamic publishing, the article depends on the image for content, and cannot be published without the image. In static publishing, however, the template chosen to render the asset defines its own dependencies. Some possibilities are:

- The approval template code re-creates the article-image association as a dependency, thereby creating on the published page a relationship that already exists in the data model.
- The approval template code creates dependencies on yet other assets (such as audio files), thereby creating relationships that do not exist in the data model, but will exist on the published page.
- The approval template code does not create any dependency between the two assets, thereby creating no relationship at all on the published page. The assets are treated as stand-alone content and published independently of each other, even though the data model specifies an association.

Tags that generate template dependencies are:

- `<asset:load>`
- `<asset:loadall>`
- `<assetset:setasset>`
- `<assetset:setlistedassets>`
- `<render:logdep>`

Reference dependencies are generated when a link is created from one page to another. They are registered as reference dependencies between the primary assets of the two pages. For example, if we create a hyperlink from the approval template of asset `A` to a page where asset `B` is the primary asset, the approval system will register this as asset `A`'s reference dependency on `B`. Tags that generate this kind of dependency are:

- `<render:getpageurl>`
- `<render:gettemplateurl>`
- `<render:gettemplateurlparameters>`

Approval dependencies for static publishing are a complex topic. Additional information about approval dependencies can be found in Chapter 13, "Various Topics in Static Publishing and the Approval Process."

Test Rendering

When an asset is user-approved to a static publishing destination, the approval system determines approval dependencies for that asset by evaluating the code in the template that is assigned to the asset. The approval system also performs a "test-render" in which it evaluates the template code for compositional dependencies, which are manifested when the content is published.

Compositional dependencies are the dependencies of a generated page on the assets that were used to generate that page. They are determined by the logic in that page's template.

The same tags that create template and approval dependencies at approval time also create compositional dependencies at publication time.

If a default template has been assigned to assets of that type, the approval system uses it to determine the dependencies. If there is no default template, the approval system uses the template that is specified for the given asset.

Your developers create default templates for asset types for the following reason: when the static publishing method actually publishes an asset, it does not necessarily use the template that is assigned to the asset—the code in another element could determine that a different template renders that asset in certain cases. If this is the case for your online site, it is likely that the developers who created the templates also designed default templates for the approval system to use when determining approval dependencies.

You or your site designers can set default approval templates for each asset type and for each publishing destination. See “Assigning Approval or Preview Templates,” on page 290.

Dynamic Publishing and Export to XML

For dynamic and **Export to XML** publishing methods, the approval system tests approval dependencies against the data model.

> **Note**
>
> Dynamic publishing and Export to XML allow for the possibility of unsatisfied **compositional dependencies** on the live site, because templates that render the page are not evaluated during dynamic publishing.
>
> Compositional dependencies appear when a dynamic page is assembled. They involve a set of assets that are used to generate the page. For more information about compositional dependencies, see “Test Rendering,” on page 214 and Chapter 13, “Various Topics in Static Publishing and the Approval Process.”

Ensuring Version-Matched Content

Any publishing session (such as the one in Figure 5, on page 213) can be made more stringent by the administrator requiring parent assets to match child assets in terms of version, thereby ensuring self-consistent content. Dependency on version is not encoded in either the data model or the template code:

- For dynamic publishing, it is specified in one of Content Server’s properties—`mwb.conservative.dependencies` (in the `gator.ini` file)—by setting the value of the property to `exists` or `exact`.
- For static publishing, it is specified by setting the `deptype` attribute in the relevant tags to `exists`, `exact`, or `none` (the tags are listed in “Evaluating Approval Dependencies,” on page 214).

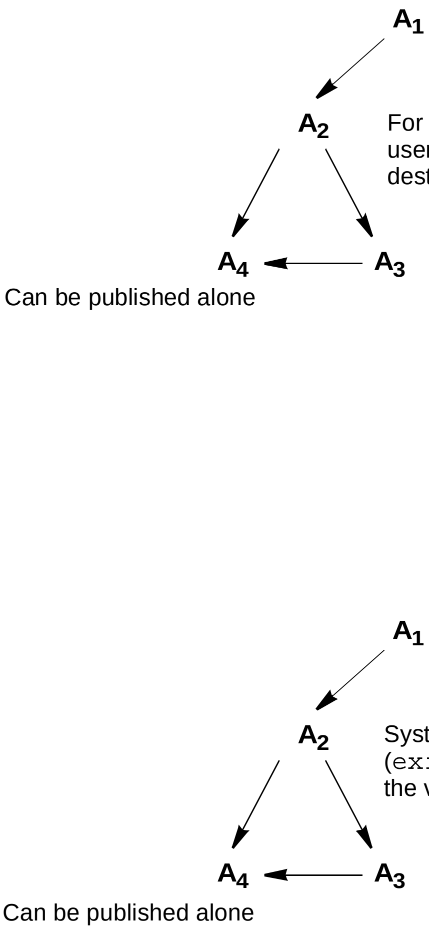
Dependency on version is qualified by the terms `exists`, `exact`, or `none`.

- An **`exists` dependency** does not require version matching. For a parent to be published, its child assets must simply exist either as approved assets or as published assets on the publishing destination. The version of the parent asset *need not* match any versions of its child assets.

- An **`exact` dependency** requires version matching. This dependency is identical to the `exists` dependency, except that the version of the parent must match the versions of its child assets (which means that all assets in the dependency must match each other's version).
- A **`none` dependency** causes the tag in which it is used to specify no *approval* dependency, at all.

Figure 6 summarizes and illustrates `exists` and `exact` dependencies.

Figure 6: Exists/Exact Dependencies



`Exists` dependencies do not require version matching:

For A_1 to be system-approved, A_2, A_3, and A_4 must exist either as user-approved assets (or as published assets on the target destination)

For A_2 to be system-approved, A_3 and A_4 must exist either as user-approved assets (or as published assets on the target destination)

For A_3 to be system-approved, A_4 must exist either as a user-approved asset (or as a published asset on the target destination)

`Exact` dependencies require version matching:

System-approval of A_1 is the same as in Figure 6 (`exists` dependency, *but* the version of A_1 must match the version of its child asset, A_2

System-approval of A_2 is the same as in Figure 6 (`exists` dependency), *but* the version of A_2 must match the version of its child assets, A_3 and A_4

System-approval of A_3 is the same as in Figure 6 (`exists` dependency), *but* the version of A_3 must match the version of its child asset, A_4

Exists-Exact Dependencies in Dynamic Publishing

For dynamic publishing of flex assets, whether the `exists` or `exact` dependency is in effect depends on the value of the property `mwb.conservative.dependencies`, in the `gator.ini` property file.

- The default value (`false`) sets `exists` dependencies.
- A value of `true` sets `exact` dependencies.

> **Note**
>
> When `mwb.conservative.dependencies` is set to `false` (exists) and an attribute is in use, the following fields of the attribute are locked and cannot be changed: Value Type, Storage Style, External ID, External Table, and External Column.
>
> If you change the value of `mwb.conservative.dependencies`, you must re-approve the assets that were affected by the change.

Exists-Exact Dependencies in Static Publishing

For static publishing, the developer sets `exists`, `exact`, or `none` dependencies using the `deptype` attribute in applicable tags, such as `getpageurl` and `assetload`. (For a listing of the tags, see “Test Rendering,” on page 214).

Template dependencies are *by default* `exact` dependencies—if you wish to approve asset A you must approve B if B has been changed. Reference dependencies are *always* `exists` dependencies—if you approved and published B once, you are not required to approve it again in order to re-publish A.

The exception is when you set `deptype="none"` on any of the tags. As a result, no *approval* dependency at all is created by the tag. This means no record is created for the dependency *during approval*. In all other contexts, such as static publishing and live sites, the `deptype` attribute is ignored.

Exists/Exact Dependency Rules

When an asset is approved to a dynamic or **Export Assets to XML** destination, the approval system determines approval dependencies against the data model.

> **Note**
>
> When reading this section, bear in mind that in the context of an approval dependency, the term “dependent asset” is equivalent to the “parent asset.” For rules governing parent-child relationships in approval dependencies, see “Approval Dependencies and Parent-Child Relationships,” on page 212.

Basic Asset Types

For basic assets, the approval system follows these dependency rules:

Relationship to Approved Asset	Dependency	
	Exists	Exact
Named association	✓ by default, unless configured to be `exact`	
Unamed association. The dependent asset is referenced by the approved asset through an unnamed association.	✓	
For page asset only: another page asset at a lower level in site plan	✓	
Embedded link	✓	
Embedded pagelet		✓

- Rules for named associations:
  - Depending on how your asset associations are designed, an approved asset has either an `exists` or `exact` dependency on assets that it references through named asset associations.
- Rules for unnamed associations:
  - An approved asset has an `exists` dependency on assets that it references through unnamed associations.
- Rules for page assets:
  - An approved page asset has an `exists` dependency on page assets that are lower than itself in the hierarchy of the site plan (which is reflected in the `SitePlanTree` table).
- Rules for embedded links:
  - An approved asset has an `exists` dependency on assets that it references by an embedded link. For information about embedded links, see the *Content Server Developer's Guide*.
- Rules for embedded pagelets:
  - An approved asset has an `exact` dependency on assets that it references as embedded pagelets. For information about embedded pagelets, see the *Content Server Developer's Guide*.

CSElement and SiteEntry Assets

The root element for a SiteEntry asset is represented by a CSElement asset. The following rule applies:

> An approved SiteEntry asset has an `exists` dependency on the CSElement that it references.

Flex Families

For flex family members, the approval system follows these dependency rules:

Approved Asset	Approved Asset’s Dependencies						
	Flex Parent Definition	Flex Parent	Flex Definition	Flex Attributes	Flex Filter	Attribute Editor	Template
Flex Parent Definition	exists			exists			
Flex Parent	exists	exists		exists			
Flex Definition	exists			exists			
Flex Asset		exists	exists	exists			exists
Flex Attribute					exists (by default)	exists	

- Rules for flex parent definitions:
    - An approved flex parent definition has an `exists` dependency on its flex parent definition(s) and flex attributes.
- Rules for flex parents:
    - An approved flex parent has an `exists` dependency on its flex parent definition, flex parent, and flex attributes.
- Rules for flex definitions:
    - An approved flex definition has an `exists` dependency on its flex parent definition, flex parent, and flex attributes.
- Rules for flex assets:
    - An approved flex asset has an `exists` dependency on its flex parents, flex asset definitions, flex attributes, and template.
- Rules for flex attributes:
    - An approved flex attribute has either an `exists` or an `exact` dependency on its flex filter, depending on how the flex filter is coded. The default flex filter is coded for an `exists` dependency. If the attribute is of type “asset,” the user can decide whether there is an `exists` or `exact` dependency on that asset.
    - An approved flex attribute has an `exists` dependency on its attribute editor (if one is assigned).

For information regarding `exists` and `exact` dependencies among flex family members, see “Flex Families,” on page 219. For more information about the functions of the flex family members, see chapter 1 of the *Content Server Developer’s Guide*.

Engage Visitor Data Assets

The approval system uses the following rules to determine approval dependencies for the Engage visitor data asset types:

	Approved Asset's Dependencies				
Approved Asset	History Definition	History Attribute	Visitor Attribute	Recommendation	Related Flex Asset
History definition		exact			
Segment	exists		exists		
Recommendation					
Promotion				exists	
Flex Asset					exists

- History definitions have `exact` dependencies on their history attributes.
- Segments have `exists` dependencies on the history definitions and visitor attributes used in them.
- Promotions have `exists` dependencies on the recommendation assets that they override.
- Flex assets have `exists` dependencies on any assets that are designated as related items (for Related Items recommendations)

Evaluating Published Content

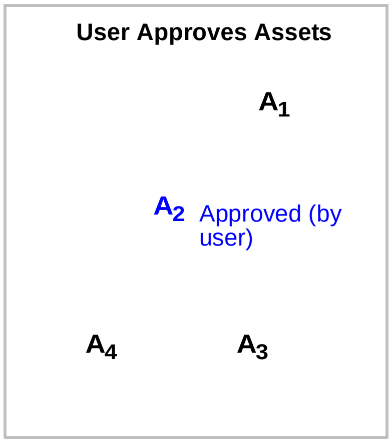
If the approval system determines the existence of an unsatisfied approval dependency, it searches for published content to avoid redundant approvals.

Specifically, the approval system reads the `PublishedAssets` table. If assets currently requiring approval are listed as published to the given destination, the approval system considers it unnecessary to approve (and re-publish) the same assets, assuming version-matching is not a requirement. If an `exact` dependency is specified in the data model or template code, the approval system checks the versions of the published assets. Information about enforcing version-matched content is given on page 215.

Putting It All Together

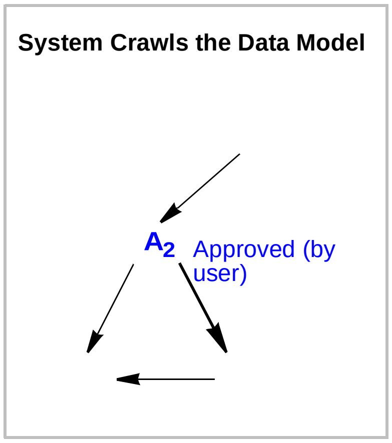
This section contains a scenario that provides a comprehensive illustration as to how the approval system works. The scenario begins with the user approving an asset, continues with the approval system's dependency analysis, and ends with the approval system's response. This scenario touches on all the concepts that were discussed throughout this chapter: approvals, approval dependencies, and approval states, using a dynamic publishing example and an `exists` dependency.

1. In this scenario, a user edits and approves a single asset — A_2 (out of a possible four) — to be published dynamically under an `exists` dependency. Only A_4 has been previously published to the target destination.

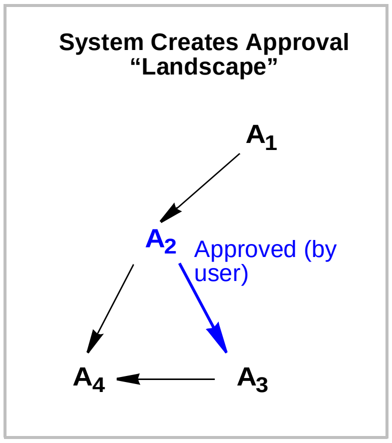


2. When the user attempts to publish, the approval system crawls the data model to determine the encoded dependencies (represented by arrows in the figure to the right).

   Using the approved asset as a seed, the approval system follows the asset's links to and from other assets in the data model, follows the other assets' links to and from yet other assets, and so on until the system determines the boundaries of the seed's dependency "network" and no more dependencies remain to be detected.



3. From the results of step 2, the approval system constructs an approval "landscape," which identifies all the interdependent assets and relationships among them.

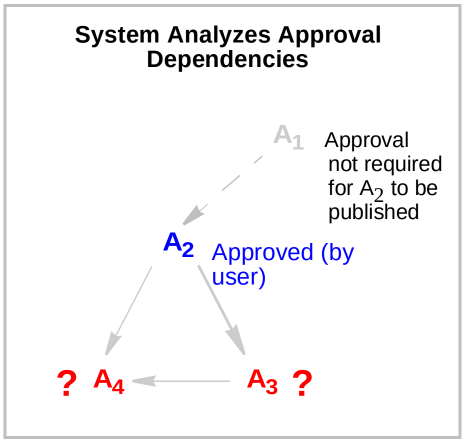


4. The approval system tests for approval dependencies and their impact on the publishing session. In this example, the approval system determines the following:

   - A_1 does not require approval because the approved asset A_2, as its child, can be published independently. However, A_2 is also a parent asset and cannot be published until the status of its child assets (A_3 and A_4) is determined.

     The approval system now searches for previously published assets. Referring to the `PublishedAsset` table, the approval system determines:

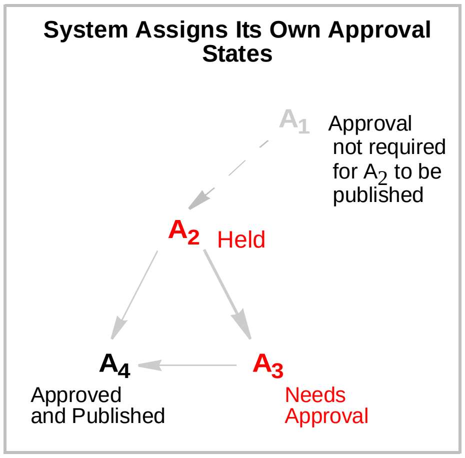
   - A_3 has never been published to the target destination. A_3 requires approval to satisfy the $A_2 \rightarrow A_3$ dependency.
   - Because A_4 has been previously published to the target destination, its re-approval (and re-publication) is unnecessary. The dependency of A_2 and A_3 on A_4 will be satisfied at publication time.


System Analyzes Approval Dependencies
A1
Approval not required for A2 to be published
A2
Approved (by user)
? A4
A3 ?

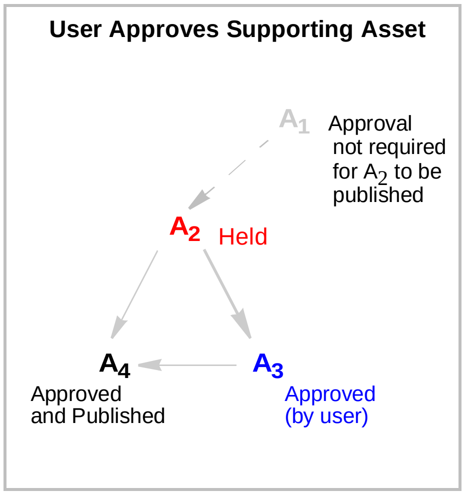
5. From step 4, the approval system concludes that three approval dependencies must be satisfied in order for the approved asset to be published, and, accordingly, assigns its own approval states to the assets:

   - The $A_2 \rightarrow A_3$ approval dependency needs to be satisfied (by the user approving A_3). The system assigns A_3 the **Needs Approval** state and A_2 the **Held** state.
   - $A_3 \rightarrow A_4$ needs to be satisfied (by the user approving A_3, which is already assigned the **Needs Approval** state).
   - $A_2 \rightarrow A_4$ is automatically satisfied by the pre-publication of A_4, which is now assigned the **Approved and Published** state.

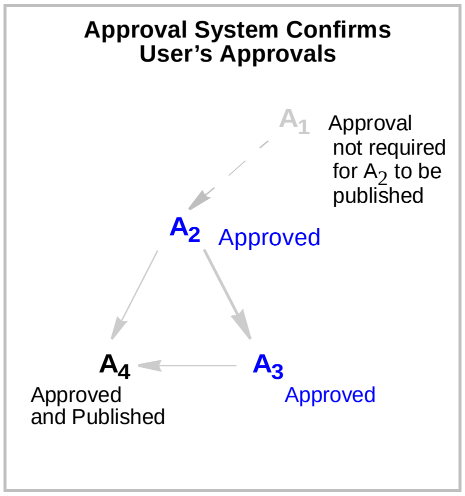
   (A complete list of approval states is available in Table 1, on page 224.)


System Assigns Its Own Approval States
A1
Approval not required for A2 to be published
A2
Held
A4
Approved and Published
A3
Needs Approval

6. Finally, the approval system displays the list of assets and their approval states (shown in the figure above). In response, the user approves the asset that was assigned the **Needs Approval** state (A_3).


User Approves Supporting Asset
A_1
Approval not required for A_2 to be published
A_2
Held
A_4
Approved and Published
A_3
Approved (by user)

7. To complete the approval process, the approval system updates the approval state of A_2 to **Approved** and releases both A_2 and A_3 to the publishing system for publication in the next publishing session.


Approval System Confirms User's Approvals
A_1
Approval not required for A_2 to be published
A_2
Approved
A_4
Approved and Published
A_3
Approved

Reference: Approval States

Table 1 lists the possible approval states that can be assigned by the approval system to assets in an approval dependency and displayed to the user.

Note

When referring to Table 1, bear in mind that in the context of an approval dependency, the term “dependent asset” is equivalent to the “parent asset.” For rules governing parent-child relationships in approval dependencies, see “Approval Dependencies and Parent-Child Relationships,” on page 212.

Table 1: Approval States Assigned by the Approval System

Approval State	Description
Approved. Approved and ready to publish to *destination*.	(Informational) This asset will be published at the next publishing event to this destination, unless the asset is changed, or an `exact` dependency changes.
Approved and published. Asset version is the same as that on *destination*.	(Informational) An asset with an `exact` dependency has been published to this destination.
Currently checked out. Will not be published to *destination*.	(Action may be required) The asset is checked out under revision tracking. Although approved, it cannot be published until revision tracking relinquishes control: • Check in – The asset must be re-approved. • Undo Checkout – The asset remains approved and can be published. • Rollback – The asset must be re-approved.
Approved for inclusion as a link in pages exported to *destination*.	(Informational) This asset is approved for static publishing, if it is linked to from the page that is being exported.
Asset has been modified since approved for publish to *destination*.	(Action required) The asset must be re-approved.
Approved, but approval for publish to *destination* was based on versions of the child assets that no longer exist.	(Action required) The asset must be re-approved so that its version matches the version of its child assets.

Table 1: Approval States Assigned by the Approval System

Approval State	Description
Held. Approved, but child assets have not been approved for publish to *destination*.	(Action required) The asset is held from a publishing session when any one of the following conditions is true: • The asset was approved, but was then re-edited and has not yet been re-approved. • The asset has a child asset with an `exact` dependency, and that child asset is not approved. • The asset has a child asset with an `exists` dependency and that child asset has never been published. • The asset has a child asset with an `exact` dependency. The child asset has been published, but it has since been edited and is not yet approved. • The asset has an `exact` dependency on a child of another asset that was previously approved, but has since been edited and is not yet re-approved. • The asset is checked out (revision tracking).
Needs Approval. Not yet approved for publish to destination.	(Action required) The asset must be approved.
This asset cannot be published until the assets it references have been approved.	(Action required) A referenced asset must be approved before this asset can be published. Related assets that are held are also listed and may require approval.

Chapter 13

Various Topics in Static Publishing and the Approval Process

This chapter presents the main concepts and features of static publishing, including the approval process as its relates to static publishing. When reading this chapter, keep in mind the approval and publishing contexts. While they tend to be discrete—for example, approval templates, pubkeys, and starting points apply to only one context, but not the other—approval and publishing systems depend on each other and therefore interact to some extent.

This chapter contains the following sections:

Static Publishing Terminology

Compared to mirror publishing, in which dependencies in both the approval and publishing stages are determined by the asset model, static publishing is an entirely template-driven process, in both the approval and publishing stages. If the approval template and publishing templates differ with respect to the dependencies they specify—a condition we do not recommend—the likelihood of publishing unapproved content increases.

Coding templates is the responsibility of the developer. However, the Content Server administrator (and any other users who publish) must understand which templates to invoke and what outcome to expect in order to prevent adverse effects on publishing sessions.

This chapter begins by explaining the most commonly used terms in static publishing, then delves into the workings of the approval system. It concludes with a series of examples illustrating how various relationships among assets and templates determine approval and publishing results.

> **Note**
>
> The terminology presented here refers only to static publishing. Terms such as **publish key**, **dependencies**, and **references**, have completely different meanings in mirror publishing and are beyond the scope of this document.

Page

In the static publishing context, a **page** is typically an HTML file that is exported to a disk. (The file is not to be confused with a Page asset type.) Each exported page is represented by a Publish Key.

Publish Key

A **publish key** (or pubkey) is strictly a publish-time term. A pubkey represents an atomic publishing unit, which in the case of static publishing is an exported file (equivalent to a Page, not to be confused with a Page asset type).

A pubkey is defined by two constructs: an asset (called a Primary Asset) and a publish-time template (not to be confused with an Approval Template, because they may be different).

The initial pubkeys in static publishing are the Starting Points, defined by the user who initiates the publishing session. All other pubkeys are discovered and logged during publishing.

Publish Queue

A publish queue is a list of pubkeys awaiting publication. For a pubkey to be added to the publish queue, its Primary Asset must be approved.

Primary Asset

The **primary asset** participates in the definition of a pubkey. In layman's terms, its ID and type become the `cid` and `c` parameters for the template. A primary asset must be approved before its page(s) can be exported.

> **Note**
>
> If it is determined that a page's primary asset is not approved, the page will not be exported. This can lead to the creation of a broken link on the publishing destination, depending on which approval and publishing templates were used.
>
> Ultimately, it is the developer's responsibility to code approval templates such that they can validate the dependencies that are expected to exist on the target destination.

Approvals and Static Publishing

Exporting a page first requires approval of the page's *primary asset.* Approval of the primary asset is contingent on the approval states of its dependent assets, which are dictated by the primary asset's **approval template**. The dependent assets' dependencies are further dictated by their own approval templates.

> **Note**
>
> Because the approval template is not necessarily the one used to render the asset, it is possible to publish assets without passing them through the approval system. This implication and others are discussed and illustrated in "Behavior and Functionality," on page 233 and "Approval and Publish-Time Templates: What Happens When They Differ?," on page 235.
>
> Also note that compositional dependencies (involving non-primary assets of a page, as explained on page 231) need not be approved in order to appear on an exported page (to filter out unapproved assets, see the `<render:filter>` tag in the *Developer's Tag Reference* for more information).

Approval Template

The purpose of approval in static publishing is to allow the developer to define dependencies in a way as to ensure that the correct content is published and it is published intact.

Compared to mirror publishing, in which asset dependencies in both the approval stage and publishing stage are determined by the asset model, static publishing is entirely template-oriented. As a result, approval behavior (as well as publish-time behavior) is very much defined by the developer who writes the approval templates (as well as the publish-time templates). Content Server then uses the approval template to discover an asset's dependencies. If an asset is published against the approval template, its approval dependencies are likely to be the same as its Compositional Dependencies.

Approval templates are assigned by use of the "Set Default Templates" feature in the publishing destination screen.

- If a default template is not explicitly chosen, the approval system will choose the asset's default template.
- If neither the approval nor default template is specified, the asset will be approved without an approval template (i.e., it will have no approval dependencies).

If the approval template is shared among sites, the approval system will choose the site entry corresponding to the current site.

Approval Queue

The approval queue handles asset modification events to keep approval tables up to date. For example, if an asset is approved for publication, but you then change it, the approval queue will handle this by unapproving the asset—rejecting it from the publish queue.

You never work directly with the approval queue. The queue runs every 5 minutes by default. However, if you invoke a feature that uses approval functionality (such as the **Publishing** tab), the approval queue is forced to run before the screen is rendered to keep approval information up to date.

Approval Dependencies

When you approve an asset, its Approval Template is used to determine its approval dependencies. The following tags create approval dependencies:

- `<asset:load>`
- `<asset:loadall>`
- `<assetset:setasset>`
- `<assetset:setlistedassets>`
- `<render:logdep>`
- `<render:getpageurl>`
- `<render:gettemplateurl>`
- `<render:gettemplateurlparameters>`

If you approve asset `A` whose template (or any element called by the template) references asset `B` using one of the tags above, an approval dependency is created from `A` to `B`. This generally means that when you want to approve `A`, you must also approve `B`.

Types of Approval Dependencies

Approval dependencies are created and used at approval-time, not to be confused with publish-time dependencies (even though they can be the same). Approval for static publishing involves two types of approval dependencies:

- Template dependencies
- Reference dependencies

Template dependencies are created when an asset's approval template uses another asset to define the content. For example, if asset `A`'s approval template loads asset `B`, then `A` has a template dependency on `B`. In more practical terms, if you have an approval template for a Page asset that shows an Article asset, the Article asset is used on the Page, so the dependency is of a template kind. The following tags generate template dependencies:

- `<asset:load>`
- `<asset:loadall>`
- `<assetset:setasset>`

- `<assetset:setlistedassets>`
- `<render:logdep>`

Reference dependencies are generated when a link is created from one page to another. They are registered as reference dependencies between the primary assets of the two pages. For example, if we create a hyperlink from the approval template of asset `A` to a page where asset `B` is the primary asset, the approval system will register this as asset `A`'s reference dependency on `B`. Tags that generate this kind of dependency are:

- `<render:getpageurl>`
- `<render:gettemplateurl>`
- `<render:gettemplateurlparameters>`

Template dependencies (in static publishing) are *by default* `exact` dependencies — if you want to approve `A` you must approve `B` if `B` has been changed. Reference dependencies are *always* `exists` dependencies—if you approved and published `B` once, you are not required to approve it again in order to re-publish `A`.

The exception is when you set `deptype="none"` on any of the tags. In that case, no *approval* dependency at all is created by that tag. This means no record is created for it *during approval;* in all other contexts, such as static publishing and live sites, the `deptype` attribute is ignored.

Static Publishing

Three main components play a role in the static publishing process:

- Publish-Time Template
- Starting Points
- Compositional Dependencies

Publish-Time Template

The purpose of the publish-time template is to render content as files. Typically, publish-time templates are identical to the approval templates. However, when they differ, content can be published without first being approved. Bypassing the approval system requires the developer to provide a means by which publishable content can be validated.

Starting Points

The **starting point** is the pubkey (or pubkeys) where export begins, which is at publish-time. Typically, starting points are selected to link to most, if not all the pages on your site. (The page may not necessarily be the home page; for example, it could be a sidebar with many links. The choice is yours.) The static publishing system will crawl your site, beginning with the starting points, and will keep logging new pubkeys as it discovers them.

Compositional Dependencies

Compositional dependencies are the dependencies of a generated page on the assets that were used to generate that page. They are determined by the logic in that page's Publish-Time Template. Compositional dependencies dictate what is rendered on the exported page as prescribed by the template, completely ignoring any `deptype` attributes. The same tags that created template and reference dependencies in the approval template also create compositional dependencies at publish-time. The tags are listed in "Approval Dependencies," on page 230.

Frequently Asked Questions

How Do I Select an Approval Template?

The approval template (if specified) is the only template ever used by the approval system to validate a given asset. This template may or may not be used at publish-time, depending on what the starting point is set to. The best advice is to set approval templates that most closely represent the asset's intended dependencies.

What if the template that contains the most representative set of dependencies is not the template that you want to publish the asset with? Set it as the approval template for assets of that type, and use any desired template(s) as starting point(s).

Are Data Model Dependencies Accounted For in Any Way?

Named/unnamed associations, attributes, and so on, are not used in static publishing. The only dependencies that matter are those established by templates.

Why Do We Track Publish-Time Compositional Dependencies?

After you export a page, its content is frozen. However, the assets used to generate that page may evolve, making the affected pages obsolete.

Because it records dependencies, Content Server is able to remind you which pages need updating, assuming you will want to republish those pages. Content Server gives you two republishing options: automatic refresh and manual refresh.

- With automatic refresh, pages are queued automatically for the next publishing session, without your having to approving compositional dependencies that have changed.

  If you choose this option, you will not have to approve the modified assets in order for the page to be queued for publishing; in other words, the article page will be republished as soon as you change the image asset. To set up automatic refresh, specify `exists` or no dependencies between the article and the image; this disables the approval system from prompting you to approve the assets.

  > **Note**
  >
  > In version 5.5 (and all versions prior to 2004), Content Server was silent until the assets involved in compositional dependencies were approved. While this provided more control, it made it harder to determine which pages were affected.

- With manual refresh, you review the updated pages before publishing them.

  If you choose this option, the affected pages will be `Held` until approved; in other words, you must reapprove the article page every time the image changes. For this to work, you need to set up `exact` dependencies.

Behavior and Functionality

Following is a compilation of simple scenarios which demonstrate static publishing behavior:

- Example 1: Template Dependencies
- Example 2: Reference Dependencies
- Approval and Publish-Time Templates: What Happens When They Differ?

The participating asset types are Page and HelloArticle (from HelloAssetWorld). The templates are `Ttemplate`, `RTemplate`, and `dummyTemplate`.

For clarity, the asset IDs have been hardcoded into the templates. Normally, assets would be loaded somehow instead of being hardcoded, so developers would need to leverage the `deptype='none'` attribute in their tags if they do not wish to set up additional dependencies. See “Sample Templates,” on page 237 for the template code.

Example 1: Template Dependencies

Setup: Page `P` asset uses `Ttemplate` for both approval and publishing. `Ttemplate` loads HelloArticle `A`, therefore establishing a template dependency.

Starting Point: `P+Ttemplate`

Pubkeys (exported files): `P+Ttemplate`

Approval Dependencies: Page `P` has template dependencies on `Ttemplate` because of `<render:logdep>`, and on HelloArticle `A` because of `<asset:load>`. Because `Ttemplate` does not create any links, there are no reference dependencies.

Publish Dependencies: Starting Point is the only pubkey; therefore only one file will be exported.

Note that the exported page does not have any publish-time compositional dependencies because it is never actually used by the template. There is only reason why `P` had to be approved—`P` is the exported page’s primary asset. So, the only compositional dependencies determined at publish-time are the exported page’s dependencies on `A` and `Ttemplate`.

Action	Approval Status	Publish Status	Comments
Initial	Approved: `P`, `A`, `Ttemplate`	Published: `A` and `Ttemplate`	`P` was **not** published because it was not referenced by the template.
Edit `P`	`P` status becomes `Changed`	Not affected by change.	Since there are no compositional dependencies on `P`, the change had no effect.

Action	Approval Status	Publish Status	Comments
Edit A	P status becomes Held. A status becomes Changed – need to approve P again.	After approval, page is published again	Template dependency on A causes P to be held, and subsequent approval makes the page publishable again.

Example 2: Reference Dependencies

Setup: Page P asset uses Rtemplate for both approval and publishing. Rtemplate references HelloArticle A, which is in turn rendered by dummyTemplate.

Starting Point: P+Rtemplate

Pubkeys (exported files): P+Rtemplate, A+dummyTemplate

Approval Dependencies: Page P has the routine <render:logdep> template dependency on its Rtemplate. In addition, it establishes a reference dependency on A because of <render:getpageurl> (but **not** on dummyTemplate, because dependencies between pages are logged only between their primary assets). Note that dummyTemplate does not participate in an approval dependency because it is not referenced by any tags.

Publish Dependencies: The relevant tags are <render:logdep> and <render:getpageurl>. These dictate the publish-time compositional dependencies as Rtemplate and A. As before, note that Page P does not have a compositional dependency because it is not used by any of the templates. When export encounters <render:getpageurl>, it detects that another page is involved, so it creates the A+dummyTemplate pubkey on-the-fly, and runs dummyTemplate to generate its contents. Note that dummyTemplate does not officially have a compositional dependency because it is not used by any tags—it participates only as a pubkey member; so, if dummyTemplate is changed, the approval/publishing cycle will not recognize the change. For this reason, we advise not altering <render:logdep>, which appears on every newly created template.

Action	Approval Status	Publish Status	Comments
Initial	Approved: P, A, Rtemplate	Published: A, Rtemplate	P was **not** published because it was not referenced by any template. The result is two HTML files: P+Ttemplate and A+dummyTemplate
Edit P	P status becomes Changed – approve it to go back to status quo	Not affected by change.	Since there are no compositional dependencies on P, the change had no effect.

Action	Approval Status	Publish Status	Comments
Edit A	A status becomes Changed; P is still Approved, **not** Held because it has only an exists dependency on A (no reason to hold P).	P+Rtemplate was automatically queued for re-publishing because <render:getpageurl> generated a compositional dependency. (If you are not sure as to why, keep reading); A+dummyTemplate was not affected because it does not use A.	A+dummyTemplate was not affected because it does not have tags that use A.

Approval and Publish-Time Templates: What Happens When They Differ?

Generally, using the same template for approval and for publishing does not create problems, because approval dependencies match compositional dependencies. Problems start to occur when you use a different template for approval than for publishing. This can happen in several situations. (For template code, see "Sample Templates," on page 237.)

Example 1

Asset A uses T1 for approval. T1 does not reference any assets.
Asset A uses T2 for publishing. T2 references asset B using <asset:load>.
T2 loads asset B, which was never approved, but B will nevertheless be on the exported A+T2 page!

Does this mean you can publish assets that were never approved?

Yes! Some users prefer this, only to avoid approving assets before publishing. Approval can be a taxing operation because exact dependencies cause primary assets to be given Held status if the dependent assets change. As a result, pages of those primary assets cannot be exported. However, a better way to handle such situations is to make good use of the deptype attribute with exists or none values to avoid unwanted approval dependencies.

Now, what happens after asset B is published and then we change B? In this case, there is no approval dependency between asset A and asset B, but there is a compositional dependency. This means that whenever asset B is changed, the affected pages will be automatically re-published. (Because there are no dependencies, the automatic refresh option is chosen). So, A+T2 will be placed in the publish queue automatically as soon as asset B is saved—not approved. (For information about republication options, see "Why Do We Track Publish-Time Compositional Dependencies?," on page 232.)

Example 2

Asset A uses T1 for approval. T1 references asset B using <asset:load> under an exact dependency. Asset A uses T2 for publishing. T2 does not reference asset B.

In this case, there is an approval dependency between asset `A` and asset `B`, but there is no compositional dependency. This means that whenever asset `B` is changed, asset `B` has to be approved before asset `A` can be approved, even though the published page does not make any use of asset `B`.

Example 3

Asset `A` uses `T1` for approval. `T1` does not reference any assets.
Asset `A` uses `T2` for publishing. `T2` references asset `B` using `<asset:getpageurl>`.

In this case, there are no approval dependencies, so `A` is the only approved asset. During publishing of `A+T2`, we realize there is another page to be created, with `B` as primary asset. Now, remember that **primary assets must be approved before their page is exported**. However, `B` is not approved, so its page will not be exported. `A+T2` is exported, but has a broken link.

Unknown dependencies and static publishing. Situations become complicated when you start using `<render:unknowndeps>`. This tag removes compositional dependencies, and it records unknown dependencies to the pubkey. Assets thus remain in the publish queue permanently (meaning they are refreshed with every publish cycle). This behavior is expected, because if unknown dependencies exist (that is, dependencies may change at any moment, such as query results), the system cannot determine which assets a given asset depends on. To be on the safe side, the system always republishes the assets.

Many `exact` dependencies. While this is not always avoidable, keep in mind that when a primary asset has `exact` dependencies on other assets, a change to any of the dependent assets will cause the approval system to change the primary asset's status to `Held`. The primary asset's page is not publishable unless approved again. If the modified asset is used by many pages, they will all need re-approval. You may consider setting the `deptype` to `exists`, and make the dependent pages automatically publishable—but the modified asset must be ready when the publishing session begins.

Note

The preferred way to customize export URLs (without using `simplename`) is to use `PREFERREDFILE` and `PREFERREDDIR` as parameters to `<render:getbloburl>` and `<render:getpageurl>` to specify arbitrary names. For example:

```
 <render:getbloburl
         blobtable='MungoBlobs' blobcol='urldata'
         blobkey='id' blobwhere='1088466917821'
         outstr='pagelogoURL' csblobid='1088466917821'>
   <render:argument name='PREFERREDFILE'
         value='myBlob.out'/>
   <render:argument name='PREFERREDDIR' value='myDir'/>
</render:getbloburl>
```

This code exports the blob whose id is `1088466917821` to `myDir/myBlob.out`.

Sample Templates

This section lists the template implementations used in the examples on pages 233 and 235. Taglib and import definitions have been skipped.

Ttemplate:

```
<%-- Record dependencies for the Template --%>
<ics:if
   condition='<%=ics.GetVar("tid")!=null%>'><ics:then><render:l
   ogdep cid='<%=ics.GetVar("tid")%>' c="Template"/></
   ics:then></ics:if>
<asset:load name='myArticle' type='HelloArticle'
   objectid='1156878442427'/>
```

Rtemplate:

```
<%-- Record dependencies for the Template --%>
<ics:if
   condition='<%=ics.GetVar("tid")!=null%>'><ics:then><render:l
   ogdep cid='<%=ics.GetVar("tid")%>' c="Template"/></
   ics:then></ics:if>
<render:getpageurl outstr="myURL" pagename='HelloAssetWorld/
   dummyTemplate' cid='1156878442427' c='HelloArticle'/>
Got URL: <a href='<%=ics.GetVar("myURL")%>'> Click here</a><br/
   >
```

dummyTemplate:

completely blank, no `logDep`.

Chapter 14

The Static Publishing Process

This chapter explains how static publishing works and how it is configured.

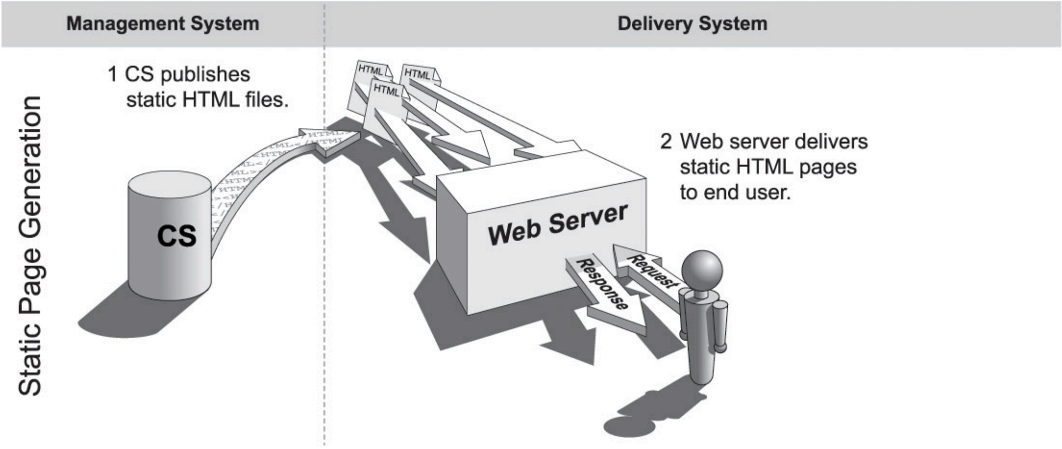
This chapter contains the following sections:

Rendering Static Pages

In extremely simple terms, Content Server renders a static page as follows:

1. A page name is submitted by the client browser to Content Server (through HTTP or another protocol).
2. Content Server locates the page in the `SiteCatalog` table, invokes the root element, and renders the page to an `html` file.
3. You publish the file (either through FTP or another file transfer protocol) to the web server that is hosting your online site.

Typically, administrative users of the static publishing method set up a quality assurance process to test the rendered files before moving them to the web server of the delivery system.



How Static Publishing Works

When content is statically published, the approval system, the publishing schedule, and your destination configurations all contribute to the process of exporting your approved assets into static files.

When a static publishing session runs, this is what happens:

1. The publishing system notifies the CacheManager servlet that a session is about to begin for a specific destination. CacheManager clears all the pages that were previously exported to this destination from the page cache on the source system.

   During an export publishing session, exported files are cached to the page cache. If publishing sessions occur more frequently than the cache is configured to be cleared (expiration time), there could be exported pages in the cache that have the same names as the newly exported pages. To ensure that old files are not used, the CacheManager servlet clears the page cache of any pages that were exported to during previous publishing sessions before a new publishing session starts.

2. Content Server creates an export queue and adds all references that can be published to it. A reference that can be published is one of the following:

- A pubkey that has been designated as an **export starting point**. (See "Export Starting Points," on page 250.)

  There must be a starting point for the site and its asset must be approved, or the export session cannot begin. Every site that is exported must have at least one export starting point designated for it.
- Any previously exported page or pagelet (reference) whose assets have changed and been approved again since the last publishing session to this destination. (After a site has been published, the publishing system knows which references have been published. It reads the data in the `PublishedAssets` table and then adds any references whose assets have been changed and approved again for this destination to the export queue.)
- Any asset whose template uses `RENDER.UNKNOWNDEPS` tag.

  This tag alerts the approval system that the dependencies for the asset being rendered by the template cannot be calculated because they are unknown. It is typically used for queries. When the dependencies are unknown, the system cannot determine whether changed dependents would require the asset to be published. Therefore, the presence of this tag means that the asset must be published during every publishing session.
- Any page or pagelet (reference) that has not yet been published and is connected to another page or pagelet through a `RENDER.GETPAGEURL` tag. (That is, it is referenced from another page and has not yet been published.)
- Any page or pagelet (reference) identified by a `RENDER.SATELLITEPAGE` or `satellite.page` tag.

3. For the first export starting point, Content Server determines whether its asset has been approved:
   - If the asset has not been approved, the publishing session ends.
   - If the asset is approved, Content Server determines the page name of the template assigned to the asset.
4. Content Server renders the export starting point by passing the following information to Content Server:
   - The page name of the template for the starting point.
   - The `rendermode` variable set to `export[`*`destinationID`*`]`.
   - The name and location of the export directory where the rendered files should be saved.
   - Values for publishing arguments that are explained later in this section ("Static Publishing Arguments," on page 242.)
5. Content Server invokes the root element of the page name, and begins rendering every approved asset that is connected to that export starting point as a reference. Content Server writes the resulting rendered pages to files rather than posting them to the browser.
6. After each file is rendered, Content Server writes a message about that reference to the publish log for the session.
7. If there is more than one export starting point, the process cycles back tostep 3 in this description.

8. When the publishing session is successful, Content Server concludes the session by writing information about the published references to the `PubKey` and `PublishedAssets` tables.
9. Content Server also writes information about the assets that were published to the `ApprovedAssets` and `ApprovedAssetDeps` tables. It logs the date from the assets' updated field at the time that the assets were published so the approval system can calculate dependencies correctly the next time an asset is approved.

Static Publishing Arguments

When you configure publishing destinations for static publishing, you must provide publishing arguments that Content Server needs in order to publish assets correctly. You provide these arguments from the Content Server interface. Some of the publishing arguments are used to name the export files, while others are used to generate links (HREFs) within the files.

Table 2 lists the publishing arguments for static publishing. The arguments are listed as they are displayed in the "Publishing Destination" form. The name of the corresponding argument is listed in parentheses:

Table 2: Static Publishing Arguments

Publishing Argument	Description
URL Prefix `(URLPREFIX)`	Optional. A prefix that the Export process adds to the beginning of the URLs that are used as links (HREFs) within the exported files. Use this argument to specify the web server alias of your delivery system in the URLs so that the links in the HTML files can be resolved when the files are moved to that system. If you do not specify a value for this argument, URLs are relative. The names of the generated files do not include this prefix—only the links within the files use it. Because you should always test the HTML files before you move them to the delivery system, be sure that the name of the web alias on the testing system is the same as the name of the web alias on the delivery system.

Table 2: Static Publishing Arguments *(continued)*

Publishing Argument	Description
Base Directory (DIR)	Optional. A subdirectory name for the files published to this export destination. It is created as a child directory of the base directory specified by the `cs.pgexportfolder` property. When you use this argument, the directory path for the publishing destination is as follows: `<cs.pgexportfolder>/< DIR>` Note that URLs that are contained in exported files do not include this destination directory path in the URL. Exported files have URLs that begin below this level, inside the destination directory. Therefore, if you use the `DIR` argument and you are using the `URLPREFIX` argument, too, be sure that the web root or the web alias represented by the `URLPREFIX` argument points to this directory. This argument is typically used to organize the contents of the base export directory when there are multiple export destinations for your sites.
Suffix (SUFFIX)	Optional. The file suffix to use for the generated files when the **Filename** field for the asset is not used or it does not specify a suffix. The default is **html**.
Use Simple File Naming (SIMPLENAME)	Optional. If set to `true`, this argument overrides the normal file-naming conventions. (For a description of the file-naming conventions, see "Export Path and File Naming Conventions," on page 245.) When set to `true`, this argument instructs Content Server to ignore the `SiteCatalog` page entry of the template that is used to render the asset and to use only the value entered in the asset's **Filename** field for the name of the file. If the asset does not have a filename, Content Server uses the asset's ID, instead. **Note:** There is one exception to this rule: assets with upload fields—that is, an asset that is a blob—will always receive a standard (long) file name, even when this value is set to true. **Caution.** This argument must be used with caution, and, if your site design is such that individual assets are rendered by more than one template, you cannot use this argument. Without the `SiteCatalog` page entries used in the file name, file names cannot be guaranteed to be unique when an asset is rendered by more than one template.

Table 2: Static Publishing Arguments *(continued)*

Publishing Argument	Description
Use simple directory naming (SIMPLEDIR)	Optional. If set to `true`, this argument overrides the normal directory-naming convention. (For a description of the file-naming conventions, see "Export Path and File Naming Conventions," on page 245.) When set to `true`, this argument instructs Content Server to put the rendered HTML file directly into the default export directory when there is no path information available from the parent asset. (Normally, Content Server uses the page asset's ID when there is no path information.) Note that if there is a value for path, this argument is ignored.
Old Template (OLDTEMPLATE)	Deprecated. This argument instructs Content Server to use the rendering methodology that was standard in the 3.5 and earlier versions of the product. If the online site that you present on your delivery system was designed for the rendering model used in versions of the product before the 3.6 version and it has not yet been redesigned for the 3.6 rendering model, you must set this argument to `true`. Note that if you set this argument to true, you do not set an export starting point for your site. The export starting point is determined by the templates themselves.
Verbose Output (VERBOSE)	Optional. Activates error logging during the publishing process. When set to true, additional messages are written to the `PubMessage` table, rather than just errors. Be sure to use the `VERBOSE` publishing argument only for troubleshooting because it causes the publishing process to take longer than normal.

Export Path and File Naming Conventions

Content Server names exported files and saves them to directories that it also names according to a specific convention. This section explains the directory and file naming conventions, and provides examples that you can use to determine how your exported files and their directories will be named.

Path Naming Conventions

Content Server names paths to exported files by using a convention that depends on the type of asset and whether the asset contains path information of its own. The export path can be as simple as a single directory or it can be a path of several directories, limited only by the number of characters that are supported by the operating system.

The export path naming convention is:

```
<cs.pgexportfolder>/<DIR>/<file_dir>                [1]
```

where

- `<cs.pgexportfolder>` sets the base directory for all exported files. It is a required argument and applies to all static publishing destinations.

  `<cs.pgexportfolder>` is the value of the `cs.pgexportfolder` property in the `futuretense.ini` file. Files are exported either directly to `<cs.pgexportfolder>` or to its subdirectories if the `DIR` argument is used (for more information about `DIR`, see "Static Publishing Arguments," on page 242).

  The directory specified by `<cs.pgexportfolder>` or by its combination with the `DIR` argument (i.e., `<cs.pgexportfolder>/<DIR>/)` is the **destination directory**.

  Typically, `<cs.pgexportfolder>` names a testing location (a file system where the exported files are verified) rather than a location directly on the delivery system.

- `<DIR>` is used to create subdirectories of the base directory (specified by `cs.pageportfolder`).

- `<file_dir>` is the directory to which the HTML file is exported. It takes one of the values shown in Table 3, on page 246, depending on whether the asset is a page asset and whether it specifies path information of its own.

  In general, the directory name is based on the value set in the **Path** field of the asset's parent page asset. If there is no value set for path, Content Server uses the ID of the parent page asset instead. However, note the following about path information:

  - Path information entered for a page asset is used only for its children assets. When a page asset is exported, it receives the path set for its parent page, not the one set for that page asset.
  - For an asset other than a page asset, if path information is entered in the **Path** field for the asset, that information is used instead of the information provided by the parent page asset.
  - Path information set for an asset for a specific destination on the "Specify Path/Filename *for destination*" form is used in place of any path information provided in the **Path** field on the asset's "New" or "Edit" form or provided from the parent page asset when the asset is exported to that destination.

Table 3: Directory Names

<file_dir>	Description
`<parent_page’s_path>`	When a page asset is exported, it receives the path set for its parent page, not the one set for that page asset.
`<asset’s_path>`	For an asset other than a page asset, if path information is entered in the **Path** field for the asset, that information is used instead of `<parent_page’s_path>`.
`<ID_of_parent_page>`	If no path information is available from the parent page or the asset itself, (i.e., `<parent_page’s_path> and` `<asset’s_path>` are not available), then `<ID_of_parent_page>` is used.
`<file_dir>` is omitted	If `<parent_page’s_path>` and `<asset’s_path>` are both unavailable **and** `SIMPLEDIR` is set to `true`, then `<file_dir>` is omitted from the export path. The export path becomes: `<cs.pgexportfolder>/<DIR>` For more information about `SIMPLEDIR`, see page 244.

Table 4, on page 247 provides examples of how export paths are named during static publishing.

Table 4: Sample Export Paths

Page Asset	<cs.pg exportfolder>	DIR	parent_page's_path	asset's_path	ID_of_parent_page	Export path
Home	/export					/export
Home	/export	/Japan				/export /Japan
World News	/export				998877665	/export/998877665
World News	/export				998877665	/export/news
World News	/export	/Japan	/news/		998877665	export/Japan/news
Asset Other Than Page						
Oil Price	/export				997766554	/export/997766554
Oil Price	/export	/Japan			997766554	/export/Japan/ 997766554
Oil Price	/export	/Japan	/news/		997766554	/export/Japan/ news
Oil Price	/export		/news/	/energy	997766554	/export/energy
Oil Price	/export	/Japan		/energy	997766554	/export/Japan/ energy
Oil Price	/export	/Japan	/news/	/energy	997766554	/export/Japan/ energy

Paths for Links Within Exported Files

URLs in links for HREFs within the exported files do not include the values that specify the destination directory (<cs.pgexportfolder>/<DIR>). Instead, they begin within the destination directory and are relative to that directory.

The URLPREFIX argument is used to resolve URLs by specifying the location of those files on the delivery system. That is, URLs for links within the exported files are created like this:

```
<URLPREFIX>/<path_of_parent_page>
```

This means that the value for the URLPREFIX argument must match the value of the web alias that identifies the directory where the files are found. If you are using only cs.pgexportfolder to create the destination directory, the web alias that the URLPREFIX represents must point to that location. And if you are using the DIR argument to add a subdirectory to the destination directory specified by cs.pgexportfolder, be sure that the web root or the web alias represented by the URLPREFIX argument points to this directory.

File Naming Conventions

How Content Server names an exported file depends on whether a file name is provided by the asset (in the “Inspect” form) and whether the SIMPLENAME parameter (page 243) is set to true or false. Possible file names are shown in Table 5. The naming convention is explained below the table.

Table 5: File Naming Conventions

File Name	Description
<pagename>_ <packedargs(if any)>_ <filename>.<SUFFIX>	Used for assets with a **populated** “Filename” field and SIMPLENAME=false
<packedargs(if any)>_ <filename>.<SUFFIX>	Used for assets with a **populated** “Filename” field and SIMPLENAME=true
<pagename>_ <packedargs(if any)>_ <assetID>.<SUFFIX>	Used for assets with an **empty** “Filename” field and SIMPLENAME=false
<packedargs(if any)>_ <assetID>.<SUFFIX>	Used for assets with an **empty** “Filename” field and SIMPLENAME=true

Note

When export files are named, the following syntax changes are made:

- Slash characters (`/`) in page names are converted to hyphens (`-`) in file names.
- Equal signs (`=`) in packed arguments are converted to hyphens (`-`) in file names.
- Ampersand characters (`&`) in packed arguments are converted to underscores (_) in file names.

- `<pagename>` is the name of the SiteCatalog page entry of the template that is used to render the asset. (All template assets have SiteCatalog page entries.)

Note

`<pagename>` is dropped when the `SIMPLENAME` publishing argument is set to `true`. Be sure to use `SIMPLENAME` carefully. If your assets are rendered by more than one template, do not use `SIMPLENAME`. For more information about `SIMPLENAME`, see page 243.

- `<packedargs(if any)>` are passed in from the page entry's root element. If any packed arguments are passed in from the root element of the asset's rendering template, the values of those arguments are also included in the name of the file generated for the asset.

  For information about packed arguments and how they relate to URLs, see the *Content Server Developer's Guide*.
- `<filename>` is the value of the asset's "Filename" field. If the "Filename" field is empty, Content Server uses the `asset_ID` specified in the asset.

Note

A file name set for an asset for a specific destination in the "Specify Path/ Filename *for destination*" form supersedes the file name provided in the "Filename" field on the asset's "New" or "Edit" form when the asset is exported to that destination.

- `<asset_ID>` is the ID that is specified in the asset. `<asset_ID>` is used if the asset's "Filename" field is empty or if the following note applies.

Note

Content Server needs both the asset's ID and type to determine whether a file name is provided for the asset. If the identity of the asset's type is not provided, the file name cannot be used. In such a case, Content Server uses the asset's object ID in place of its file name.

- `<SUFFIX>` is a publishing argument that specifies the extension of the file name. For more information about `SUFFIX` see "Static Publishing Arguments," on page 242.

Table 6 provides examples of how file names are created during static publishing. In the examples `SIMPLENAME` is set to `false`.

Table 6: Directory Naming Conventions

Asset Name	SiteCatalog Pagename	packedargs	filename	asset_ ID	SUFFIX	Exported File Name
Home	BF/Page/Home	cid=123 c=Page		123		BF-Page-Home_123.html
Home	BF/Page/Home	cid=123 c=Page		123	.htm	BF-Page-Home_123.htm
Home	BF/Page/Home	cid=123 c=Page	home.html	123		BF-Page-Home_home.html
Home	BF/Page/Home	cid=123 c=Page	home.html	123	.htm	BF-Page-Home_home.html
Home	BF/Page/Home	cid=123 c=Page PACKEDARGS= "topicword=oil"	home.htm	123		BF-Page-Home_topicword_oil_home.htm

Export Starting Points

The static publishing method cannot begin rendering files unless it knows where to start. You tell it where to start by designating at least one starting point: that is, a page asset and the template that should be used to render it. Content Server invokes the root element of the template's page name, and begins rendering every approved asset that is connected to that export starting point—assets connected through an association, a hyperlink, a navigation bar, a query, and so on.

Note

There must be at least one export starting point for an exported site. Typically it is your home page. However, depending on how your online site is designed, you may need to designate more than one export starting point. Why? Because if there is a section on your site that isn't connected to the rest of the site, it won't be rendered. For example:

- If your online site has more than one top-level page asset, you need an export starting point for each one.
- If your site uses a combination of static pages and dynamic pages and there is a hard-coded static URL in an HREF on a dynamically-generated page, the asset that the URL points to should be designated as an export starting point.

Configuring Your System for Static Publishing

The main steps when configuring a system for static publishing are these:

- Step 1. Create the Batch User Account (If One Does Not Exist)
- Step 2: Specify the Base Export Directory
- Step 3: Configure an Export Destination
- Step 4: Map a URL Prefix for Your Web Server
- Step 5: Create the Export Starting Points
- Step 6: Approve Your Assets
- Step 7: Publish and Test the Results
- Step 8: Set Up the Schedule

Note

Configuring a system for static publishing requires you to provide information that is specific to your Content Server installation and your business practices. Before starting the procedures in this section, it is best to read them in order to gather and confirm the information they prompt you to provide.

Step 1. Create the Batch User Account (If One Does Not Exist)

Note

The steps in this section must be completed regardless of the number and types of publishing methods you are setting up. If a batch user account already exists for your installation, skip to "Step 2: Specify the Base Export Directory," on page 253.

The purpose of creating a batch user account is two-fold:

- Configure a **batch user** account for the publishing system on the source to use.
- Specify where the publish **logs** should be stored.

The procedures in this section require you to set several properties in the `futuretense.ini`, `batch.ini`, and `futuretense_xcel.ini` files. Additional properties are also available to you for fine-tuning your system configuration. For a complete list of properties, see the *Property Files Reference*.

To create the batch user account

1. Log in to the Advanced interface on your source system.
2. In the **Admin** tab, expand **Content Server Management Tools** and double-click **User**.
3. Create a user account for the publishing system on the source system to use (that is, the batch user) and assign it the following ACLs:
    - Browser
    - ElementEditor

- PageEditor
- TableEditor
- Visitor (if your installation includes Engage)
- VisitorAdmin (if your installation includes Engage)
- xceleditor
- xceladmin

If you need help with this step, see “Creating a New User,” on page 78.

4. Start the Property Editor and open the `futuretense_xcel.ini` file for the source system. (If you need help with this step, see the *Property Files Reference*.)
5. In the **Publishing** tab, set values for the following properties:
   - `xcelerate.batchhost`
     Set this property to the host name of the application server (preferable to the web server) that is hosting the **source** system. The source system is the batch host. If the port number of the web server is anything other than 80, you must also include the port number. For example, `myserver:7001`
   - `xcelerate.batchuser`
     Set this property to the name of the user that you created in step 3 of this procedure.
   - `xcelerate.batchpass`
     Set this property to the password of the user that you created in step 3 of this procedure.
6. Save and close the property file.
7. Open the `batch.ini` file for the source system.
8. In the **Results** tab, set the `request.folder` property to point to the directory that you want to hold your publish log files. By default, this property is set to the `dispatcher` subdirectory in the application server installation directory.

Note

Be sure to set this property to a directory that can be written to. Otherwise, there will be no Publish History summary information displayed in the Publish Console for your publishing sessions.

9. Save the property file and close the Property Editor.
10. Restart the application server.
11. Continue with the configuration procedure for the delivery type you are planning to use:
    - For static publishing, go to “Configuring Your System for Static Publishing,” on page 251 the next procedure in this section.
    - For dynamic publishing, go to “Configuring Your System for Dynamic Publishing,” on page 264.

- For Export Assets to XML, go to “Configuring Your System for Export Assets to XML,” on page 278.

Step 2: Specify the Base Export Directory

As mentioned previously in this chapter, the directory that Content Server exports files to is determined by the `cs.pgexportfolder` property in the `futuretense.ini` file. If you want to add subdirectories to this directory, you use the `DIR` publishing argument in the definition of the destination.

To set the base directory for the exported files

1. Start the Property Editor on the source system. If you need help with this step, see the *Property Files Reference*.
2. Open the `futuretense.ini` file.
3. In the **Export/Mirror** tab, select the `cs.pgexportfolder` property and set the value to the file directory (location) to which all files should be exported.

   This is a global setting for the system. If you have multiple destinations and want the files published to those destinations to be stored in separate subdirectories within this directory, use the `DIR` publishing argument when you configure those destinations.

   For more information about export directories, see “Export Path and File Naming Conventions,” on page 245.
4. Save the property file and close the Property Editor.
5. Restart the application server.

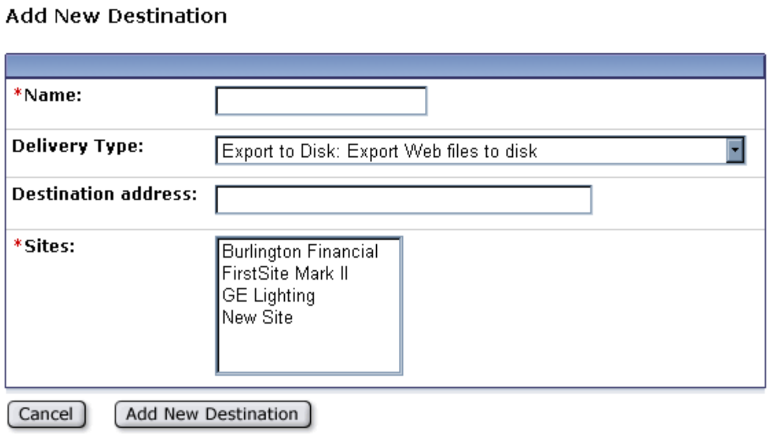
> **Note**
>
> Before you run an export publishing session, make sure that this base export directory exists and that it has enough space for the HTML pages that will be created there.

Step 3: Configure an Export Destination

To create an export destination

1. Log in to the Advanced interface on the source CS system.
2. In the **Admin** tab, expand **Publishing**, then **Destinations**, and double-click **Add New**.

Content Server displays the “Add New Destination” form.



Add New Destination

*Name:	
Delivery Type:	Export to Disk: Export Web files to disk
Destination address:	
*Sites:	Burlington Financial FirstSite Mark II GE Lighting New Site

3. In the **Name** field, enter a unique name for the destination.
4. In the **Delivery Type** drop-down list, select **Export to Disk: Export Web Files to Disk**.
5. Leave the **Destination Address** field empty. (This field is for Mirror to Server only.)
6. In the **Sites** list, select the sites whose assets can be approved for and published to this destination.
7. Click **Add New Destination**.

   Content Server writes the information to the `PubTarget` table and displays the new destination in the “Inspect” form.
8. Edit the destination and specify the appropriate configuration options.
   a. In the action bar, click **Edit**.
   b. Configure your destination by filling in fields and selecting options in the form, as appropriate for your destination.

      For descriptions of the options shown in the form, see “Static Publishing Arguments,” on page 242.
9. Repeat steps 2–8 for each additional export destination you need to configure.

Step 4: Map a URL Prefix for Your Web Server

To make your exported content available to the public at its final destination (your delivery system), you must configure your web server to map a URL path prefix to the place where the files will reside so that URLs can be resolved.

Refer to the vendor documentation for your web server for information about the appropriate procedure for mapping URL prefixes on your web server. Remember that the name you specify for this prefix must match the name that you specified with the `URLPREFIX` argument for the destination. If you use the `DIR` argument, be sure that the web root or alias includes the `DIR` directory.

Step 5: Create the Export Starting Points

To create an export starting point

1. Using the Advanced interface on the source system, find the asset which you want to specify as an export starting point.
2. Open the asset in its "Status" form and scroll to the **Publish Destination** section.
3. Do one of the following:
   - If the asset is not yet approved for the destination you are currently configuring, select **Approve for Publish** from the drop-down list in the action bar and go to step 4.
   - If this asset is already approved for the destination you are currently configuring, go to step 6.
4. In the publishing approval form, select the destination for which you are approving the asset and click **Approve**.

   Content Server calculates dependencies for all the assets linked to this asset and displays the results.
5. When Content Server displays the approval results, select **Status** from the drop-down list in the action bar.
6. Scroll down to the "Publishing Destination" section of the "Status" form.
7. In the **File/Path** field, click **Specify Path/Filename, Start points**.
8. Configure the export starting point:

   a. Select **Yes** for **Is this asset an export starting point?**

   b. Select a template.

   > **Note**
   >
   > If the publishing argument `OLDTEMPLATE` is set to `true` for this destination, no templates will be displayed for you to choose from. With the old, pre-version 3.6 style of rendering, the template is determined by the asset itself and you cannot override the template assigned to the asset in an export starting point.

   c. (Optional) If you want to specify path information that is different for this destination than any path information provided elsewhere for this asset, enter the path in the **Path** field. For more information about path names, see "Export Path and File Naming Conventions," on page 245.

   d. (Optional) If you want to specify file name information that is different for this destination, enter it in the **Filename** field. A file name entered in this form for this destination overrides any file name information provided elsewhere for the asset.

   e. (Optional) If you want the directory name for the path created according to the simple naming convention described in the definition of the `SIMPLEDIR` publishing argument, select **Force specified path**. Selecting this check box is like setting the `SIMPLEDIR` publishing argument to `true`, but only for this asset rather than for all assets exported to this destination.

f. (Optional) If you want the file name created according to the simple naming convention described in the definition of the `SIMPLENAME` publishing argument, select **Force specified filename**. Selecting this check box is like setting the `SIMPLENAME` publishing argument to `true`, but only for this asset rather than for all assets exported to this destination.

9. Click **Save**.
10. Repeat steps 1–9 for each top-level page asset in the site.

Step 6: Approve Your Assets

If you want to perform a simple test of your configuration, select the export starting point with the fewest number of dependent assets and approve each of those assets.

If you want to complete a test publish of your entire site, approve all the assets for the site. See "Approving Multiple Assets," on page 283 for help with approving many assets at once.

Step 7: Publish and Test the Results

Once the assets you want to publish to the selected destination are approved, run a test publishing session:

1. In the Advanced interface on the source system, click **Publishing** in the button bar.
2. In the **Publish Console**, select your destination from the drop-down list and click **Select Destination**.

   Content Server displays information about the assets that are ready to be published to this destination. (If you have not yet created an export starting point for this destination, the form states this fact. You must create at least one export starting point before the publishing session can begin.)
3. Click **Publish**.

   The publishing system exports all the approved assets for this destination into files.
4. Click the link to the **Publish Console**.
5. In the **Publish Console**, scroll down to **Publishing History** and click the **Inspect** icon next to the publishing session.

   A session summary displays a list of all the files that were created during the export process.
6. Click the file representing the top-level page in your site.

   Content Server opens the page in a new browser window.
7. Scan the page for errors and click all of the links it contains to verify that they work. Make sure to test links to all of the other files that were generated.
8. Test the results by directly entering the URL of the rendered home page into your browser and navigating the entire site. To do this, you must first set up a web server root on the CS management system that points to the export directory.
9. If you decide that the files are ready for delivery, copy them to your delivery system.

Step 8: Set Up the Schedule

After you have ensured that

- your destination is configured correctly,
- the file names and directory names are created according to your needs, and
- your web server is delivering the files correctly

you can finish configuring your publishing system by completing the following steps on the source system:

- Create scheduled publish events for the destination. For help with this step, see "Scheduling Publish Events," on page 287.
- Plan how you will copy the generated files to your web servers, for both the testing area and the delivery system. For example, you can set up a script that automatically copies the files via FTP to a testing area after a publishing session completes.

Chapter 15

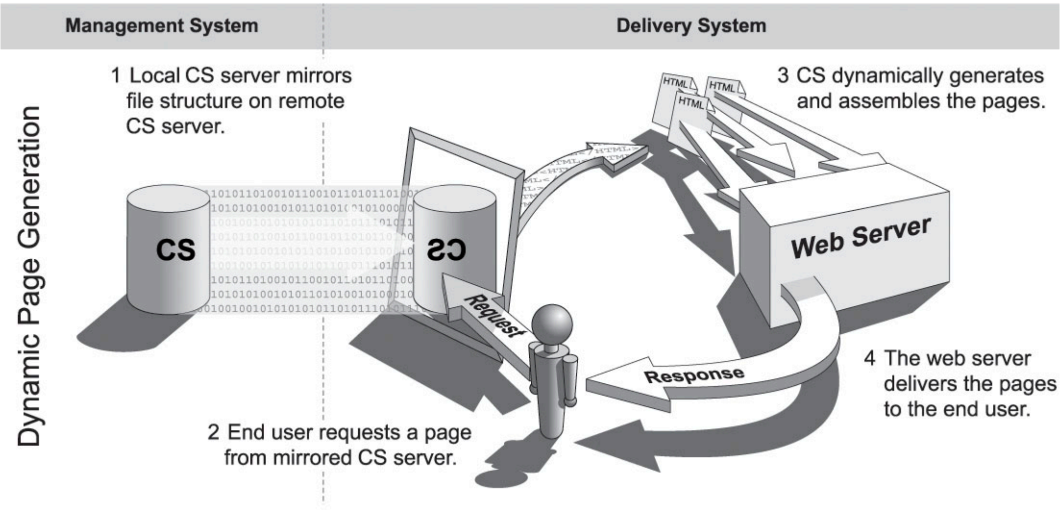
The Dynamic Publishing Process

This chapter explains how dynamic publishing works and how it is configured.

This chapter contains the following sections:

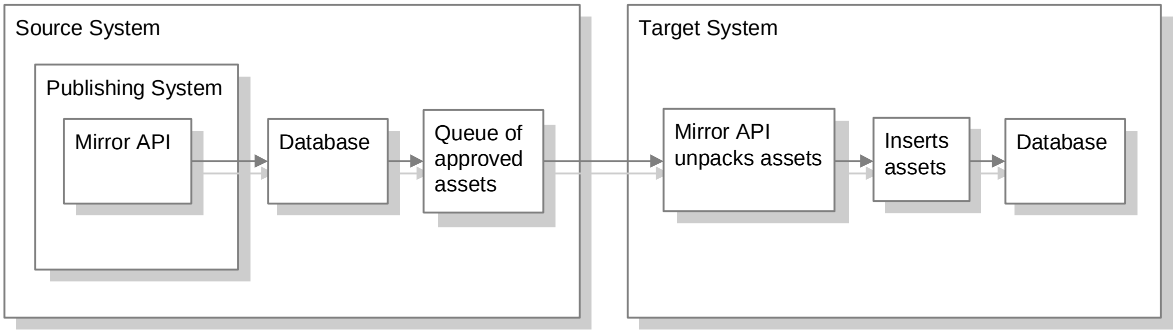
How Dynamic Publishing Works

The dynamic publishing method gathers information from the approval system, the publishing schedule, and the destination configurations, copies data to the destination, unpacks that data on the destination system, and then invokes the CacheManager servlet to refresh any pages that should be regenerated to take advantage of the new content.


Management System
Delivery System
Dynamic Page Generation
1 Local CS server mirrors file structure on remote CS server.
CS
CS
HTML
HTML
HTML
3 CS dynamically generates and assembles the pages.
Web Server
Request
Response
2 End user requests a page from mirrored CS server.
4 The web server delivers the pages to the end user.

Whether or not your CS delivery system is dynamic, you probably use the dynamic publishing method to move data from your development systems to your management system. For information about that process, see “Troubleshooting,” on page 295.

When a dynamic publishing session runs, this is what occurs:



1. On the source system, dynamic publishing uses the list of approved assets that the publishing system passed to it to create a mirror queue for the destination.

   Mirror Queue for Basic Assets

   For basic assets, the following information is added to the queue:

   - The asset’s main table row. For example, for page assets, the asset’s row in the `Page` table.
   - The appropriate rows in the `AssetPublication` table. These rows list sites and which assets belong to them.

- Rows in the AssetRelationTree table that refer to any assets that are associated with the asset being published.
- The associated assets referenced by those rows in the AssetRelationTree table. The asset table rows, AssetPublication rows, and associated assets of any dependent assets are also mirrored, if they are approved and have not yet been published.

Mirror Queue for Flex and Complex Assets

For flex asset types and the other multi-table asset types like template and CSElement, the information from the appropriate rows from all of their tables are serialized into an object and stored in the _Publish table for that asset type.

For example, when a template is to be published, the appropriate rows from the Template, SiteCatalog, and ElementCatalog tables are serialized into an object and stored in the Template_Publish table.

Each item in a _Publish table is added to the mirror queue.

2. Content Server uses the AssetPublishList table to create a list of all the assets that are in the mirror queue.
3. The mirror operation starts.

   First, the AssetPublishList is mirrored from the source to the destination.
4. The mirror queue is delivered and the publishing system unpacks the assets in the queue.

   For flex assets, dynamic publishing de-serializes the objects in the _Publish tables, and inserts the results in the appropriate tables.

   For basic assets, each row in the queue is copied.
5. When the items in the mirror queue are unpacked, the publishing system sends messages that the mirror publish concluded successfully. The destination system responds as follows:

   - The newly-published assets are marked as changed on the destination system, which means that before they can be published from that system to another destination, they must be approved. Note that this feature is enabled by default. You can turn it off if you need to. See “Step 12: Turn Off Asset Invalidation on the Delivery System,” on page 273 for details.
   - The CacheManager servlet on the destination regenerates the appropriate pages in the cache so that all pages that refer to the assets that were just published are updated. It also rebuilds any pages that have unknown compositional dependencies.
   - CacheManager then communicates a message about which pages must be refreshed to each Satellite Server identified by the satellite.ini file on the destination system. It communicates with the co-resident version and any remote instances that are identified as belonging to this CS system. The Satellite Server applications then use that information to refresh the Satellite Server page caches.
   - The AssetPublishList table is cleared, making it ready for the next publishing session.

6. On the source system, the publishing system updates the publish log file. Unlike the static publishing method, which writes to the publish log after each asset is exported, dynamic publishing waits until the entire mirror queue has been successfully mirrored before writing the results to the publish log.
7. When the publishing session is successful, Content Server concludes the session by writing information about the published references to the `PubKey` and `PublishedAssets` tables and by clearing the `AssetPublishList` table on the source system.
8. Content Server also writes information about the assets that were published to the `ApprovedAssets` and `ApprovedAssetDeps` tables so the approval system can calculate dependencies correctly the next time an asset is approved.

Before Configuring Dynamic Publishing

Before you configure the dynamic publishing process, take into the consideration the topics that are presented in this section. Prior knowledge of the information you will be providing will help to ensure a smooth configuration process:

- Users and Dynamic Publishing
- Dynamic Publishing Arguments
- CacheManager
- Configuring a Mirror Destination

Users and Dynamic Publishing

In addition to the batch user account on the source system that a publish event uses to process a publishing session, the dynamic publishing method requires another user account: the **mirror user account**, which is located on the destination server. This is the user account that the publishing system uses to unpack the mirror queue on the destination system.

When you set up a dynamic publishing destination, you must create the mirror user account on the destination system that the destination represents and you must specify the identity of that user to the source system with the `cs.mirroruser` and `cs.mirrorpassword` properties in the `futuretense.ini` file.

Dynamic Publishing Arguments

There are three publishing arguments for dynamic publishing:

- `REMOTEUSER` and `REMOTEPASS` are used to create multiple mirror user accounts with unique user names.

  Whereas properties in the `futuretense.ini` file apply globally (to all publishing destinations), the publishing arguments provide for specificity. Both `REMOTEUSER` and `REMOTEPASS` allow you to create destination-specific accounts in cases when you publish to multiple destinations from the same source and require unique account names on each destination.

  - `REMOTEUSER` specifies the mirror user account on the destination system when it differs from the one specified by the `cs.mirroruser` property in the `futuretense.ini` file on the source.

  - REMOTEPASS specifies the password for the user account that is designated by the REMOTEUSER argument. Note that the value of REMOTEPASS is not stored in an encrypted form, which means that anyone with access to the database on the source can see the password.

- Verbose is used for error logging and applies to both dynamic and static publishing.

  VERBOSE activates detailed error logging during the publishing process. When VERBOSE=true, additional messages are written to the PubMessage table, rather than just error messages. Be sure to use the VERBOSE publishing argument only for troubleshooting, as it lengthens the publishing process.

CacheManager

The CacheManager is a Content Server servlet that maintains the page cache on any dynamic CS system, including the management system.

CacheManager is important to the publishing system because it locks the appropriate assets on the destination during a mirror publish and ensures the integrity of the page cache both before and after a mirror publishing session.

For more information about the CacheManager, see the *Content Server Developer's Guide*.

Configuring a Mirror Destination

The dynamic publishing method copies information for approved assets from one CS database to another. This information must be configured for the mirror destination. Configuring a mirror destination involves two steps: initializing the destination, and configuring the data for the destination.

Whenever you create a new mirror destination, you must initialize it before you can publish to it. To initialize a mirror destination, you specify the following information:

- The sites. Based on the sites that you select, the appropriate rows from the Publication, SitePlanTree, and PublicationTree tables are mirrored to the destination.
- Any custom table that supports or is directly related to your asset types. In other words, a table for assets that was not created by either AssetMaker or Flex Family Maker. Examples of these are Source and Mimetype. These tables are called "auxiliary tables."

Mirror destination configuration is the process of indicating which data (e.g., assets types, associations, start menu items, etc.) will be mirrored to the target destination. Mirror destination configuration moves configuration data and rows from auxiliary tables—which are non-asset database tables that are used for the dynamic display of the assets—from one CS database to another.

When to Use Mirror Destination Configuration

You use the Mirror Destination Configuration feature in several situations:

- To set up any new mirror destination.
- To move configuration items that support asset types (start menu items, associations, and so on) from a development system to a management system or to a delivery system.

- To move workflow configuration data from a development system to a management system.
- If you or your developers add any additional sites, asset types, or auxiliary tables to your system.
- If you or your developers add any additional categories or subtypes for existing asset types.
- When you need to troubleshoot your configuration. If you can successfully initialize a mirror destination, that means that the source and the destination systems are communicating.

Configuring Your System for Dynamic Publishing

The main steps when configuring a system for dynamic publishing are as follows:

- Step 1: Create the Batch User Account (If One Does Not Exist)
- Step 2: Set Up the Destination System
- Step 3: Identify the Mirror User to the Source System
- Step 4: Identify the Proxy Server (If There Is a Firewall)
- Step 5: Create a Mirror Destination
- Step 6: Initialize the Destination
- Step 7: Configure the Mirror Destination
- Step 8: Approve Your Assets
- Step 9: Publish the Assets
- Step 10: Test the Results
- Step 11: Set Up the Schedule
- Step 12: Turn Off Asset Invalidation on the Delivery System

Note

Configuring a system for dynamic publishing requires you to provide information that is specific to your Content Server installation and your business practices. Before starting the procedures in this section, it is best to read them in order to gather and confirm the information they prompt you to provide.

Step 1: Create the Batch User Account (If One Does Not Exist)

Creating a batch user account is required before any type of publishing can take place. This procedure needs to be performed only once, regardless of the number and types of publishing destinations you are creating.

- If you have not already done so, create the batch user account. For instructions, see "Step 1. Create the Batch User Account (If One Does Not Exist)," on page 251.
- If a batch user account already exists for your installation, skip to "Step 2: Set Up the Destination System."

Step 2: Set Up the Destination System

To mirror publish assets from one CS system to another, you must ensure that the sites and asset types are the same on both the source and the destination system. You must also create an additional user on the destination system, called the **mirror user** (as opposed to the batch user, which exists on the source). This user completes the mirror publish database transactions on the destination system.

> **Note**
>
> The database properties on the CS system and the destination CS system, if not an exact match, must be compatible. Otherwise, your assets cannot be mirrored.

To set up your destination system

1. If you have any custom support tables—a lookup table for a field in an asset type, for example—create those tables on the destination system.
2. On the destination system, create the mirror user and note the following:
    - This user must have the following ACLs:
        - Browser
        - ElementEditor
        - PageEditor
        - TableEditor
        - Visitor (if Engage is installed)
        - VisitorAdmin (if Engage is installed)
        - xceladmin
        - xceleditor
    - Because the mirror user account is used by the CacheManager to regenerate the page cache after a publishing session, the mirror user must have sufficient privileges to regenerate all the pages in the cache. Therefore, the mirror user account must be given to all ACLs which are assigned to page entries in the `SiteCatalog` table or database tables that are holding data to be rendered.
    - If any of the sample sites are installed on the destination system, a user named “mirroruser” already exists. For security, if you decide to use this user as your mirror user, be sure to **change the password** for this user; or if you decide to create a different mirror user, be sure to delete the sample site mirroruser.
    - If you plan to mirror publish from any source system to multiple destination systems, use one of the following options for the username and password of the mirror users that you create on each destination:
        - Make all username/password match exactly (because you can specify only one mirror user in the `futuretense.ini` file on your source system).
        - Create a mirror user on the second destination with a different name from the one specified in `futuretense.ini` and then provide that information with the `REMOTEUSER` and `REMOTEPASS` publishing arguments. For information

about the publishing arguments, see "Dynamic Publishing Arguments," on page 262.

- If you need help with creating the mirror user, see "Creating a New User," on page 78.

Step 3: Identify the Mirror User to the Source System

Next, you identify the name and password of the mirror user on the destination system to the source system by setting property values in the `futuretense.ini` file on the source system.

To identify the mirror user to the source system

1. Start the Property Editor and open the `futuretense.ini` file on the source CS system. If you need help with this step, see the *Property Files Reference*.
2. In the **Export/Mirror** tab specify values for the following properties:
   - `cs.mirroruser`

     Set this property to the name of the user that you created on the destination system in the preceding procedure.
   - `cs.mirrorpassword`

     Set this property to the password of the user that you created on the destination system in the preceding procedure.
3. Save the property file.
4. Do one of the following:
   - If there is a firewall separating your source system from the destination system, go to "Step 4: Identify the Proxy Server (If There Is a Firewall)," on page 266.
   - If there is no firewall separating your source and destination systems, close the Property Editor. Stop and restart the application server. Then go to "Step 5: Create a Mirror Destination," on page 267.

Step 4: Identify the Proxy Server (If There Is a Firewall)

If you have a firewall between your source and your destination, you must identify the proxy server to the source system. With the `futuretense.ini` file still open in the Property Editor, complete the following steps:

To identify the proxy server

1. In the **Export/Mirror** tab, specify values for these properties:
   - `cs.mirrorproxyserver`

     Set this property to either the name or the IP address of the firewall's proxy server.
   - `cs.mirrorproxyserverport`

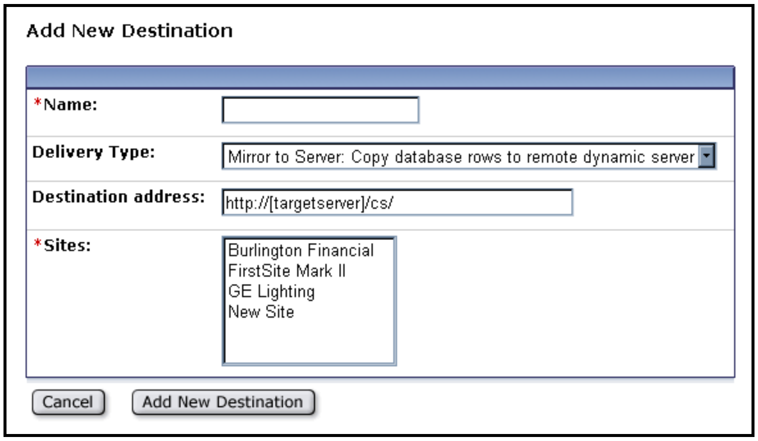
     Set this property to the port number of the firewall's proxy server.
2. Save the property file and close the Property Editor.
3. Restart the application server.

Step 5: Create a Mirror Destination

To create a mirror destination

1. Log in to the Advanced interface on the source CS system.
2. In the **Admin** tab, expand **Publishing**, then **Destinations**.
3. Under **Destinations**, double-click **Add New**.

   The “Add New Destination” form appears.



Add New Destination

*Name:	
Delivery Type:	Mirror to Server: Copy database rows to remote dynamic server
Destination address:	http://[targetserver]/cs/
*Sites:	Burlington Financial FirstSite Mark II GE Lighting New Site

4. In the **Name** field, enter a unique name for the destination.
5. In the **Delivery Type** field, select **Mirror to Server: Copy database rows to remote dynamic server**.
6. In the **Destination Address** field enter the URL containing the web and application server information for the destination.

   ```
   http://<server>:<port>/<context>/
   ```

   Note

   A slash is required at the end of the URLs because these URLs are appended dynamically.

7. In the **Sites** list, select the sites whose assets can be approved for and published to this destination.
8. Click **Add New Destination**.

   Content Server writes the information to the `Pubdestination` table and displays the destination in the “Inspect” form.

9. Edit the destination and specify the appropriate configuration options.
   a. In the action bar, click **Edit**.
   b. Configure your destination by filling in fields and selecting options in the form, as appropriate for your destination.

      For descriptions of the options shown in the form, see “Dynamic Publishing Arguments,” on page 262.

10. If you are ready to initialize this destination system (you must initialize the destination system before you can publish to it), click the **Initialize Mirror Destination** button in the "Publish Destination" form, then proceed to Step 6: Initialize the Destination" for detailed instructions (begin at step step 4 of that procedure).

    If you need to create more mirror destinations, repeat steps 2 through 9 for each additional mirror destination that you need to create for your source system.

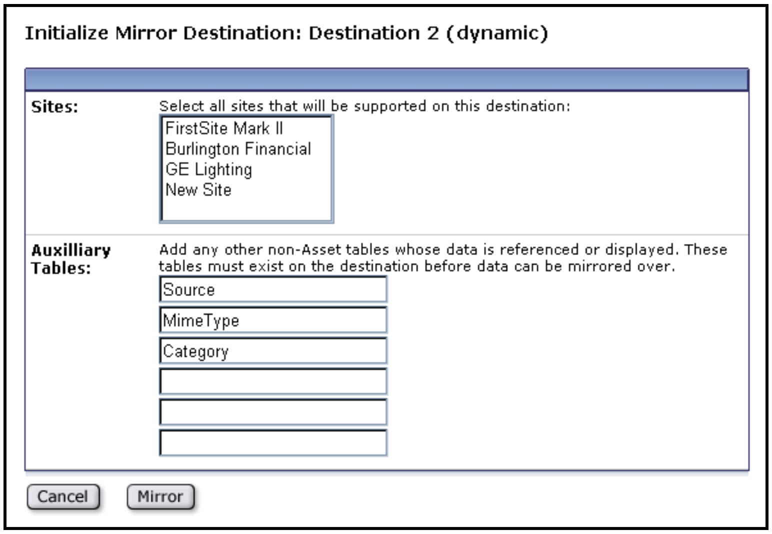
Step 6: Initialize the Destination

You must initialize the destination system before you can publish to it. This creates the basic site information on the destination system. Specifically, the `Publication` and `PublicationTree` tables are updated with the site names and asset types published to the target.

To initialize the destination system

1. In the **Admin** tab, expand **Publishing**, then **Destinations**.
2. Under **Destinations**, double-click the destination you want to initialize.
3. In the "Publish Destination" form, click **Initialize Mirror Destination**.

   The **Initialize Mirror Destination** screen appears



Initialize Mirror Destination: Destination 2 (dynamic)

Sites: Select all sites that will be supported on this destination:
FirstSite Mark II
Burlington Financial
GE Lighting
New Site

Auxilliary Tables: Add any other non-Asset tables whose data is referenced or displayed. These tables must exist on the destination before data can be mirrored over.
Source
MimeType
Category

4. Select the sites whose assets you want to publish to this destination.
5. In the **Auxiliary tables** fields, enter the names of the following tables:
   - `Source`, if you are using the source feature for any of your asset types
   - `Mimetype`, if you are using CS-DocLink, flex filter assets, the ImageFile asset type, or if you are using this table for any of your custom asset types.
   - Any other auxiliary tables (such as lookup tables) for your asset types.

   There are only five fields for tables on this form. If you have to enter more than five table names, complete this procedure for the first five tables, then repeat steps 3, 4, and 5 for the remaining tables.
6. Click **Mirror**.

If the initialization is successful, Content Server displays a confirmation message; otherwise, an error message is displayed.

Based on the sites that you selected in step 4, the corresponding rows from the `Publication`, `SitePlanTree`, and `AssetType` tables are copied to the destination.

Additionally, all the rows in the tables that you specified as auxiliary tables are copied to the destination.

Note

You can also initialize the target destination from the **Publish Console**:

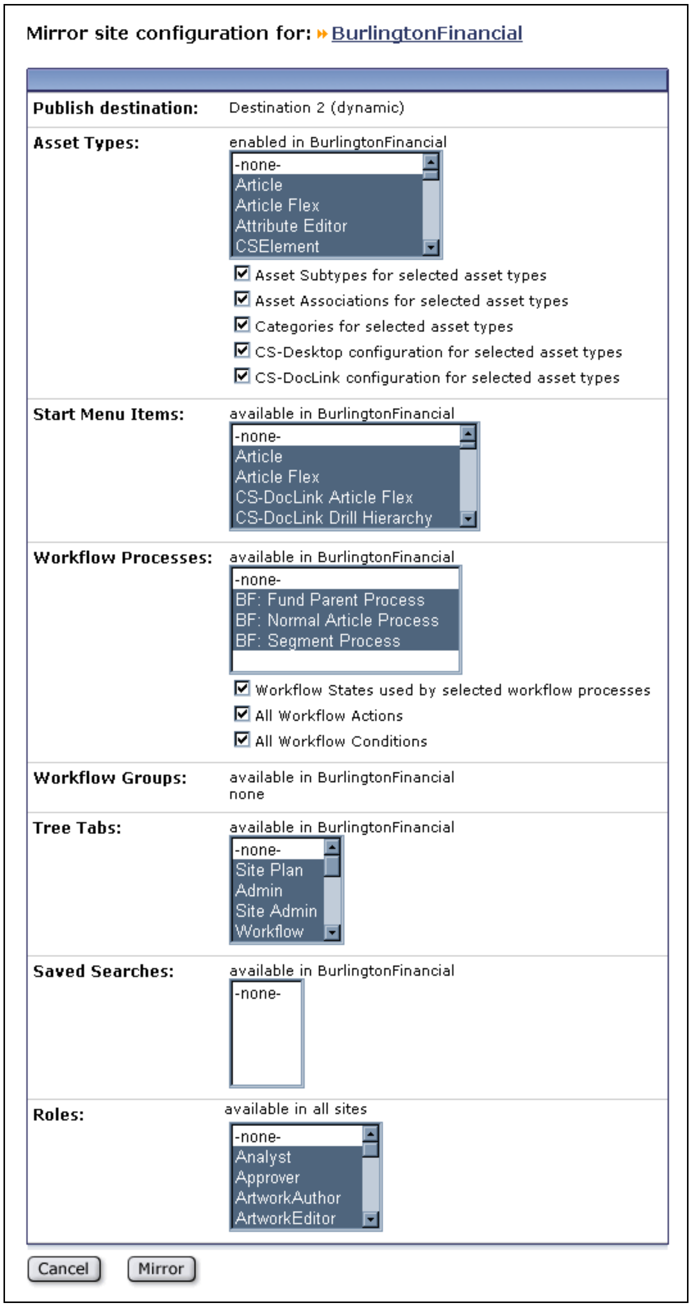
1. In the button bar, click **Publishing**.
2. In the **Publish Console**, select your mirror destination from the drop-down list and then click **Select Destination**.
3. Click the **Create Site on Target** button. This creates the basic site information on the destination system.

Step 7: Configure the Mirror Destination

Now you need to configure the data that will be on your target destination. This configuration will vary depending on the purpose of the target destination. For example, if the target destination is strictly a delivery machine, it makes sense to only include asset types when mirroring. Before proceeding with configuring the destination, you may find it helpful to refer to "Migrating a Site from One System to Another," on page 282.

1. In the in the **Admin** tab on the source system, expand **Sites** and double-click the site whose data you want to publish on the target destination.
2. In **Publish Destinations** (near the bottom of the screen), click **Mirror site configuration for *destination***, where ***destination*** is the target destination.

Content Server displays the “Mirror site configuration” form.



Mirror site configuration for: BurlingtonFinancial

Publish destination:	Destination 2 (dynamic)
Asset Types:	enabled in BurlingtonFinancial -none- Article Article Flex Attribute Editor CSElement ☑ Asset Subtypes for selected asset types ☑ Asset Associations for selected asset types ☑ Categories for selected asset types ☑ CS-Desktop configuration for selected asset types ☑ CS-DocLink configuration for selected asset types
Start Menu Items:	available in BurlingtonFinancial -none- Article Article Flex CS-DocLink Article Flex CS-DocLink Drill Hierarchy
Workflow Processes:	available in BurlingtonFinancial -none- BF: Fund Parent Process BF: Normal Article Process BF: Segment Process ☑ Workflow States used by selected workflow processes ☑ All Workflow Actions ☑ All Workflow Conditions
Workflow Groups:	available in BurlingtonFinancial none
Tree Tabs:	available in BurlingtonFinancial -none- Site Plan Admin Site Admin Workflow
Saved Searches:	available in BurlingtonFinancial -none-
Roles:	available in all sites -none- Analyst Approver ArtworkAuthor ArtworkEditor

Select the data to make available on the target destination:

- **Asset Types**: Select the asset types that you want to make available on the target destination.

  If your asset types have subtypes, categories, or associations, select the appropriate options from the list in the **Asset Types** section. Select the CS-Desktop / CS-DocLink configuration check boxes to configure the selected asset types for CS-Desktop and CS-DocLink.

- **Start Menu Items**: Select any start menu items that you or the developers designed for your content providers to use.
- **Workflow Processes**: If there are workflow processes for your asset types, select these processes and the appropriate workflow items.
- **Workflow Groups**: Select any workflow groups that you or the developers designed for your content providers to use.
- **Tree Tabs**: Select the tree tabs that you want to make available on the target destination.
- **Saved Searches**: Select any saved searches that you or the developers designed for your content providers to use.
- **Roles**: Select the roles you want to exist on the target destination.

3. Click **Mirror**.

   Based on which asset type configuration options you selected in step , appropriate rows from the `AssocNamed`, `AssocNamed_Subtypes`, and `Category` tables are copied to the destination.

   If you selected **Start Menu Items** or **Saved Searches**, appropriate rows from the tables that implement those features are copied to the destination.

   Based on which workflow configuration options you selected in step , appropriate rows from the workflow tables are copied to the destination.

4. Repeat steps 1–3 for each site whose assets you want to publish on the target destination.

Step 8: Approve Your Assets

To truly test your published site, you must approve and publish all the assets for the site. See “Approving Multiple Assets,” on page 283 for help with approving many assets at once.

If you want to perform a simple test of your configuration, temporarily create a home page asset with just a few dependents. For information about approving individual assets, see the *Content Server Advanced Interface User’s Guide*.

Step 9: Publish the Assets

After you have approved assets that can be published to your destination, you can run a test publishing session:

1. On the source system, click **Publishing** in the button bar.
2. In the **Publish Console**, select your mirror destination from the drop-down list and click **Select Destination**.

   Content Server displays information about the assets that are ready to be published to this destination.

3. If for some reason the target destination has not yet been initialized, it must be initialized now or the publishing session will fail. To initialize the destination, click **Create Site on Target** in the **Publish Console**. This creates the basic site information on the destination system.
4. Click **Publish**.

The publishing system mirrors all the approved assets for this destination to the Content Server database on the destination system.

Step 10: Test the Results

To test the results, point your browser to the home page asset on the destination system and examine the site.

If you have not already done so, you must determine the asset's URL. A Content Server URL is constructed by concatenating the following values:

- The hostname or IP address (and sometimes the port number) of the destination system.
- The CGI path, which the system obtains from the `ft.cgipath` property in the `futuretense.ini` file. For example, for WebLogic and other application servers with servlet architectures, this path is `/servlet/`.
- The string `ContentServer?pagename=`
- A page name from a `SiteCatalog` entry.
- Additional information that is passed in with the Content Server page criteria variables, `c`, `cid`, `tid`, and `p` (see the *Content Server Developer's Guide* for information about these variables).

For example, the URL for the Burlington Financial home page looks like this when it is rendered by the JumpStart Kit:

```
http://localhost:7001/servlet/
   ContentServer?pagename=BurlingtonFinancial/Page/Home
```

Complete the following steps to determine the URL of your home page and test the site:

1. Start the Property Editor and open the `futuretense.ini` file for the destination system. (If you need help with this step, see the *Property Files Reference*.)
2. In the **App Server** tab, locate the `ft.cgipath` property and write down its value.
3. In the **Compatibility** tab, locate the `ft.approot` property and write down its value.
4. Save the property file and close the Property Editor.
5. Open a text editor. Enter the server name, a slash (/), and then the cgipath. Precede the server name with the proper protocol—`http://` or `https://`—as in the following example:

   WebLogic, WebSphere, and Sun JES Application Server:

   ```
   http://bigfatsun.fatwire.com:8080/servlet/
   ```
6. At the end of the string, type a slash, and then add the following text:

   `ContentServer?pagename=`*`your_home_page`*

   Now the URL should look similar to the following examples:

   Example for WebLogic, WebSphere, and Sun JES Application Server:

   ```
   http://bigfatsun.fatwire.com:8080/servlet/
      ContentServer?pagename=ExampleSite/Page/Home
   ```
7. Point your browser to the URL you assembled.
8. Scan the page for errors and test all links to ensure they work.

Note

FatWire recommends that you conduct a complete test of the system under peak load conditions after you have mirrored the entire site for the first time, and at regular points thereafter.

Step 11: Set Up the Schedule

After you have ensured that your destination is configured correctly, you can finish configuring your publishing system by completing the following steps on the source system:

- Create scheduled publish events for the destination. For help with this step, see "Scheduling Publish Events," on page 287.
- If you are using images that are not assets in the design of your site—that is, your site designers want to store all images on the web server rather than manage them as assets—plan how you will move the image files from the web server for the management system to the web server for the delivery system. For example, you can set up a regular FTP transfer.
- If you are using elements and SiteCatalog page entries that are not CSElement and SiteEntry assets, you must use the CatalogMover tool to mirror them to the destination system. For help with CatalogMover, see the *Content Server Developer's Guide*.
- If you are using the eWebEditPro HTML editor (from Ektron) and your content providers will use the upload feature on that toolbar to upload image files, you must set up an FTP server or another means of transmitting these files from the management system to the delivery system. Because these files are not assets, they are not mirrored to the destination.

Step 12: Turn Off Asset Invalidation on the Delivery System

By default, the publishing system is configured to mark an asset as changed on the destination system when you publish an asset from one system to another (source to destination). Then, the newly-published asset must be approved on the destination system before it can be published to another destination.

The default configuration is appropriate for development and management systems. However, when you are publishing to the delivery system, there is no need for the publishing system to take the time to mark the change—assets are published to the delivery system but not from it.

Therefore, on the delivery system, turn off this publishing feature by completing the following steps:

1. Start the Property Editor and open the `futuretense_xcel.ini` file. If you need help with this step, see the *Property Files Reference.*
2. In the **Publishing** tab, locate the `xcelerate.publishinvalidate` property and set its value to `false`.
3. Save the file and close the Property Editor.

Retrieving Logs From Delivery CS Systems

Content Server provides a way to retrieve publishing-related logging information from the delivery system and display it in the Publishing console. To achieve this, Content Server inserts the publishing session ID into each relevant log entry. The management interface has the ability to retrieve the relevant entries for a publishing session and display them in the “Info” screen for that session.

To enable remote publishing logging

1. Activate the logger by setting the following property in `commons-logging.properties` on both source and target:

```
org.apache.commons.logging.LogFactory=com.fatwire.cs.core.
   logging.ContextAwareLogFactory
```

2. Add the following lines to your `web.xml` on the target system:

```
<filter>
      <filter-name>ContextHeaderFilter</filter-name>
      <filter-class>com.fatwire.cs.core.logging.context.
         filter.ContextHeaderFilter</filter-class>
</filter>
<filter-mapping>
      <filter-name>ContextHeaderFilter</filter-name>
      <servlet-name>ContentServer</servlet-name>
</filter-mapping>
```

3. On the target system, set the following `futuretense.ini property`:

```
log.file.location=<path to CS log file>
```

Chapter 16

The Export Assets to XML Publishing Process

This chapter explains how Export Assets to XML works and how it must be configured.

This chapter contains the following sections:

- The Export Assets to XML Publishing Method
- Configuring Your System for Export Assets to XML

The Export Assets to XML Publishing Method

The **Export Assets to XML** publishing method is a hybrid between the static and dynamic publishing methods. Assets are rendered into files, but this publishing method uses the dynamic dependency calculation method rather than the method of static publishing.

The Export Assets to XML publishing method differs from the other two in that it is really a data transformation method. While static publishing creates one HTML file per page, which could mean that several assets are rendered into one file, Export Assets to XML creates **one XML file** for **each asset** that is approved to a destination configured to use this publishing method.

Static publishing creates static files that are to be delivered from a static CS delivery system. In contrast, Export Assets to XML creates **XML files** for delivery to a database or non-CS system.

When an Export Assets to XML publishing session runs, this is what happens:

1. On the source system, the approval system provides a list of the assets approved for this destination to Export Assets to XML.
2. Export Assets to XML renders each asset in the list to an XML file. The output file describes the asset, stating all the values for all of the asset's fields.

When you use the Export Assets to XML publishing method, typically you set up a quality assurance process to test the generated files before you move them to the external (non-CS) system.

The XML Output

The output from Export Assets to XML is well-formed XML that describes an asset object in terms of its field values. Fields are described as attributes.

For each of the asset's fields, there is a name/value pair. The statement of the value includes the data type of the field. For example, the **Name** field holds characters. So the name/value pair for an asset's **Name** field would appear as follows in the output:

```
<attribute name="Name">
      <string value="nameOfAsset"/>
</attribute>
```

Flex attributes serve as the fields for flex assets. In the output XML for a flex asset, the flex attribute values are prepended with `Attribute_`. Here's an example of an attribute value for the price of a GE Lighting sample site product asset:

```
<attribute name="Attribute_price">
      <decimal value="5.9"/>
</attribute>
```

Note that the XML output uses the column names rather than the field name of each field/attribute and that column names and field names can be different.

The following is a description of the XML file created by this publishing method, presented as a pseudo-dtd file

```
<!-- ASSET:
-- ASSET defines an asset object
-- an asset is made up of attributes
-->
```

```
<!ELEMENT ASSET (ATTRIBUTE)*>
<!ATTLIST ASSET TYPE CDATA #REQUIRED>
<!ATTLIST ASSET ID CDATA #IMPLIED>
<!ATTLIST ASSET SUBTYPE CDATA #IMPLIED>

<!ELEMENT ATTRIBUTE (STRING | DATE | INTEGER | DECIMAL | BINARY |
FILE | ASSETREFERENCE | ARRAY | STRUCT | LIST)>
<!ATTLIST ATTRIBUTE NAME CDATA #REQUIRED>

<!ELEMENT STRING>
<!ATTLIST STRING VALUE CDATA #REQUIRED>

<!ELEMENT DATE>
<!ATTLIST DATE VALUE CDATA #REQUIRED>

<!ELEMENT INTEGER>
<!ATTLIST INTEGER VALUE CDATA #REQUIRED>

<!ELEMENT DECIMAL>
<!ATTLIST DECIMAL VALUE CDATA #REQUIRED>

<!ELEMENT BINARY>
<!ATTLIST BINARY VALUE CDATA #REQUIRED>

<!ELEMENT FILE (CDATA) >
<!ATTLIST FILE NAME CDATA #REQUIRED>

<!ELEMENT FILE>
<!ATTLIST FILE NAME CDATA #REQUIRED>

<!ELEMENT ASSETREFERENCE>
<!ATTLIST ASSETREFERENCE TYPE CDATA #REQUIRED>
<!ATTLIST ASSETREFERENCE VALUE CDATA #REQUIRED>

<!ELEMENT ARRAY (STRING | DATE | INTEGER | DECIMAL | BINARY | FILE
| ASSETREFERENCE | ARRAY | STRUCT | LIST)+>

<!ELEMENT STRUCT (FIELD)+>

<!ELEMENT FIELD (STRING | DATE | INTEGER | DECIMAL | BINARY | FILE
| ASSETREFERENCE | ARRAY | STRUCT | LIST)+>
<!ATTLIST FIELD NAME CDATA #REQUIRED>

<!ELEMENT LIST (ROW)+>

<!ELEMENT ROW (COLUMN)+>

<!ELEMENT COLUMN ( STRING | INTEGER | DECIMAL | DATE |
ASSETREFERNCE)>
<!ATTLIST COLUMN NAME CDATA #REQUIRED>
```

Publishing Arguments for Export Assets to XML

There is one publishing argument for Export Assets to XML: `DIR`. It provides a subdirectory name for the files published to the destination. It is created as a child directory of the base directory specified by the `cs.pgexportfolder` property.

When you use this argument, the directory path for the publishing destination is as follows:

`<cs.pgexportfolder>`/<**DIR**>

File Naming Conventions for Export Assets to XML

Exported XML files are named according to the following convention:

`assetID``.xml`

For example, if the asset ID is 3344556677, the file name for the asset's XML file is: `3344556677.xml`

The Export Assets to XML publishing method does not use any information that is entered in an asset's **Path** or **Filename** field.

Configuring Your System for Export Assets to XML

The main steps when configuring for the Export Assets to XML delivery type are as follows:

- Step 2: Specify the Base Export Directory
- Step 3: Configure an Export Assets to XML Destination
- Step 4: Approve Your Assets
- Step 5: Publish and Test the Results
- Step 6: Set Up the Schedule

Step 1: Create the Batch User Account (If One Does Not Exist)

Creating a batch user account is required before any type of publishing can take place. This procedure needs to be performed only once, regardless of the number and types of publishing destinations you are creating.

- If you have not already done so, create the batch user account. For instructions, see "Step 1. Create the Batch User Account (If One Does Not Exist)," on page 251.
- If a batch user account already exists for your installation, skip to "Step 2: Specify the Base Export Directory."

Step 2: Specify the Base Export Directory

As mentioned earlier in this chapter, the directory that Content Server exports files to is determined by the `cs.pgexportfolder` property in the `futuretense.ini` file.

To set the base directory for exported files, complete the following steps:

1. Start the Property Editor. If you need help with this step, see the *Property Files Reference*.

2. Open the `futuretense.ini` file on the source CS system.
3. On the **Export/Mirror** tab select the `cs.pgexportfolder` property and set its value to the full path of the directory to which files should be exported.

   This is a global setting for the system. If you have multiple destinations or you are publishing both statically and with the Export Assets to XML delivery type and you want the files stored in separate places, use the `DIR` publishing argument to override the value of `cs.pgexportfolder` when you configure your Export Assets to XML destinations.
4. Save the file and close the Property Editor.
5. Restart the application server.

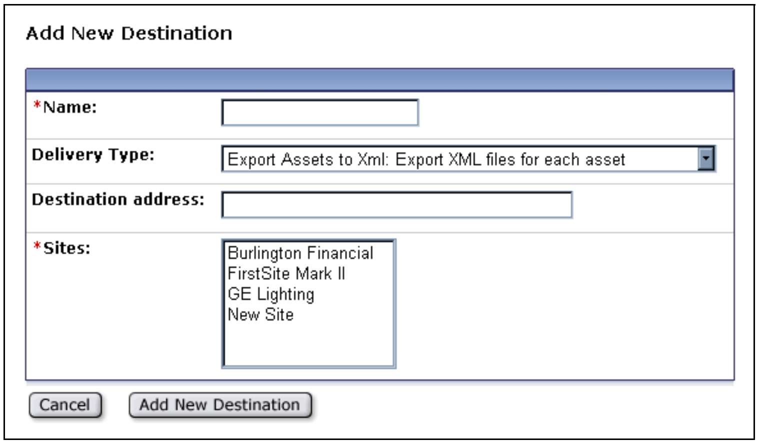
> **Note**
>
> Before you run an Export to XML publish, make sure that this destination directory exists and that is has enough space for the XML files that will be created there.

Step 3: Configure an Export Assets to XML Destination

Next, create the Export to XML destination by completing the following steps:

1. In the **Admin** tab on the source CS system, expand **Publishing,** then **Destinations**.
2. Double-click **Add New**.

   The “Add New Destination” form appears.



Add New Destination

*Name:	
Delivery Type:	Export Assets to Xml: Export XML files for each asset
Destination address:	
*Sites:	Burlington Financial FirstSite Mark II GE Lighting New Site

3. In the **Name** field, enter a unique name for the destination.
4. In the “Delivery Type” drop-down list, select **Export Assets to XML**.
5. Leave the **Destination Address** field empty. This field is for Mirror to Server only.
6. In the **Sites** list, select the sites whose assets can be approved for and published to this destination.
7. Click **Add New Destination**.

   The information about this destination is written to the `PubTarget` table.

8. Repeat steps 2–7 for each additional export destination that you need to configure for your source system.

Step 4: Approve Your Assets

If you want to perform a simple test of your configuration, select an asset and approve it and any of its dependent assets.

If you want to complete a test publish of your entire site, approve all the assets for the site. See "Approving Multiple Assets," on page 283 for help with approving many assets at once.

Step 5: Publish and Test the Results

After you have approved assets that can be published to your destination, you can run a test publishing session:

1. On the source CS system, click **Publishing** in the button bar.
2. In the **Publish Console**, select your destination from the drop-down list and then click **Select Destination**.

   Content Server displays information about the assets that are ready to be published to this destination.
3. Click **Publish**.

   Content Server exports all assets approved for this destination to files. The files are stored in the directory you specified earlier.
4. In the **Publish Console**, click the inspect icon next to the session summary for this session.
5. In the session summary, which displays a list of all the resulting XML files, verify that the files are correct.

Step 6: Set Up the Schedule

After you have ensured that your destination is configured correctly, that the directory names are created according to your needs, you can finish configuring your publishing system by completing the following steps on the source system:

- Create scheduled publish events for the destination. For help with this step, see "Scheduling Publish Events," on page 287.
- Plan how you will move the generated files to your external, non-CS systems. For example, you can set up a regular FTP transfer that automatically moves files to a testing area after a publishing session completes.

Chapter 17

Additional Publishing Procedures

In addition to the configuration steps described in the preceding sections of this chapter, administrators and developers also perform, whenever necessary, the publishing procedures described in this chapter.

This chapter contains the following sections:

Migrating a Site from One System to Another

During the development phase for your sites, your site designers and developers develop sites and asset types with all their supporting configuration (subtypes, associations, start menu items, and so on), code templates, and assist the administrators in customizing the user interface, when necessary, by writing elements for workflow and so on. Your site designers and developers do this work on the development system.

When the site is ready for the content providers, you can use the **Mirror Destination Configuration** feature to migrate it to the management system and the delivery system.

Note

You can use this feature to move your configuration data only when **none** of the **sample sites** exist on the **destination** system.

Moving a Site from a Development System to a Management System

Complete the following procedure **once**:

1. Log in to the Content Server interface on the management (destination) system as the **fwadmin** user.
2. If you have any custom support tables—a lookup table for a field in an asset type, for example—create those tables on the management (destination) system.
3. On the management system, create all the users.
4. On the source system, log in to the Content Server interface.
5. Create a **Publishing Destination** for the destination system that uses the Mirror to Server publishing method.

   For help with this step, see "Step 5: Create a Mirror Destination," on page 267.
6. Initialize the mirror destination.

   For help with this step, see "Step 6: Initialize the Destination," on page 268.
7. Configure the mirror destination.

   For help with this step, see "Step 7: Configure the Mirror Destination," on page 269.
8. On the destination (management) system, configure the users for the site. For help with this step, see "Granting Users Access to a Site (Assigning Roles to Users)," on page 98.
9. If you created a flex family, log back in to the source system. Approve and then publish all of the data structure flex assets to the management (destination) system.

   You can either approve each asset individually or use the **Approve Multiple Assets** feature on the **Admin** tab. For information about this feature, see "Approving Multiple Assets," on page 283.
10. Approve and then publish all of the rest of the appropriate assets for the destination system: template, page, collection, query, CSElement, and SiteEntry assets, and so on.

11. If there are any elements or page entries in the `SiteCatalog` table that are not CSElement or SiteEntry assets, use CatalogMover to mirror those items to the management system. For help with CatalogMover, see the *Content Server Developer's Guide*.

Moving a Site to a Delivery System

When you move a site to a delivery system, there is no need to mirror start menu items, workflow, saved searches, and so on.

Complete the following steps:

1. Log in to the Content Server interface on the delivery (destination) system as the **fwadmin** user.
2. If you have any custom support tables—a lookup table for a field in an asset type, for example—create those tables on the delivery (destination) system.
3. On the source system, log in to the Content Server interface.
4. Create a **Publishing Destination** for the destination system that uses the Mirror to Server publishing method.

   For help with this step, see "Step 5: Create a Mirror Destination," on page 267.
5. Initialize the mirror destination.

   For help with this step, see "Step 6: Initialize the Destination," on page 268.
6. Configure the mirror destination.

   For help with this step, see "Step 7: Configure the Mirror Destination," on page 269.

   Do **not** select start menu items, saved searches, or any of the workflow options.
7. If you created a flex family, log back in to the source system. Approve and then publish all of the data structure flex assets to the management (destination) system.

   You can either approve each asset individually or use the **Approve Multiple Assets** feature on the **Admin** tab. For information about this feature, see the "Approving Multiple Assets," on page 283.
8. Approve and then publish all of the rest of the appropriate assets to the destination (delivery) system: template, page, collection, query, CSElement, and SiteEntry assets, and so on.
9. If there are any elements or page entries in the `SiteCatalog` table that are not CSElement or SiteEntry assets, use CatalogMover to mirror those items to the management system. For help with CatalogMover, see the *Content Server Developer's Guide*.
10. If there are any images in the online site that are not assets, be sure to copy them to the delivery system.

Approving Multiple Assets

Each publishing destination has an option to **Approve Multiple Assets**. It is especially useful for upgrades and for the first publishing session to a destination.

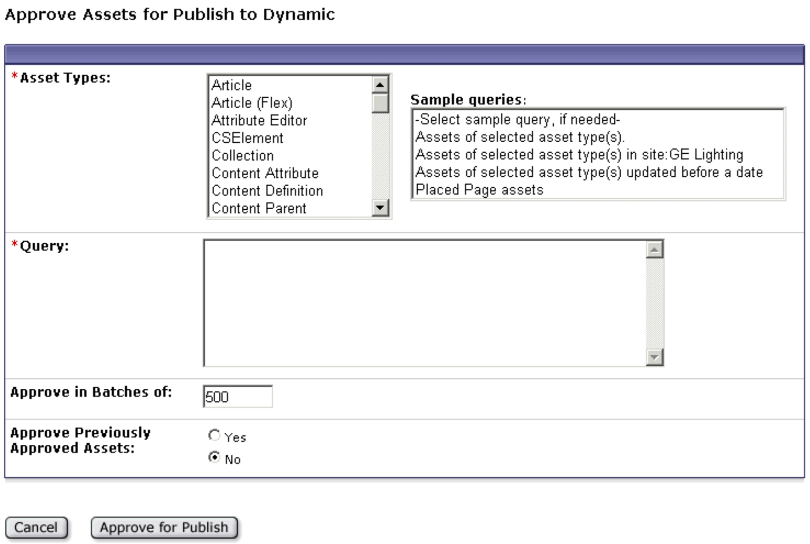
Note that the Approve Multiple Assets feature is **not** the BulkApprover utility. The BulkApprover utility approves only those assets that have been imported into the Content

Server database with the BulkLoader utility. Both BulkLoader and BulkApprover are described in the *Content Server Developer's Guide*.

To approve a group of assets, follow these steps:

1. In the Content Server interface, select **Admin** > **Publishing** > **Destinations** and then expand the destination that you want to approve assets for.
2. Under that destination, select the **Approve Multiple Assets** option.

   The "Approve Assets" form appears:



Approve Assets for Publish to Dynamic

*Asset Types:	Article Article (Flex) Attribute Editor CSElement Collection Content Attribute Content Definition Content Parent	Sample queries: -Select sample query, if needed- Assets of selected asset type(s). Assets of selected asset type(s) in site:GE Lighting Assets of selected asset type(s) updated before a date Placed Page assets
*Query:		
Approve in Batches of:	500	
Approve Previously Approved Assets:	○ Yes ● No	

3. In the **Asset Types** section, from the list on the left, select the asset types that you want to approve.
4. From the **Sample Queries** list on the right, select a query. If the exact query that you need is not present, select the query that is most like the one that you want.

   Content Server creates a SQL query based on the items that you selected and displays it in the **Query** box.
5. (Optional) If necessary, edit the SQL query.
6. Click in the **Approve in Batches of** field and enter a numeric value. The default is set to 500.

Note

When you click the **Approve for Publish** button, the approval system approves batches of assets. The number of assets in each batch is determined by value in the **Approve in Batches of** field. Because the approval process is not a background process, it is possible that the browser session could time out while the batch is being approved.

When using the **Approve Multiple Assets** feature, to ensure that the session does not time out, adjust the following settings:

- The `cs.timeout` property in the `futuretense.ini` file, which sets the browser session timeout value. Start by setting this value to 1800 (which means 30 minutes). (See the *Property Files Reference*.)
- The value you specify in the **Approve in Batches of** field. Start by using the default of 500 and lower it if necessary.

7. Specify the **Approve Previously Approved Assets** option as follows:
   - If you want Content Server to ignore all previous approvals and re-approve all selected assets, select **Yes**. This is the option to use if previous approvals may have become invalid for some change such as a change in default templates for that destination.
   - If you want Content Server to approve only the assets that have not yet been approved, select **No**. This is the correct choice when you are sure that previously approved assets are still valid.
8. Click **Approve for Publish**.

Creating Destinations

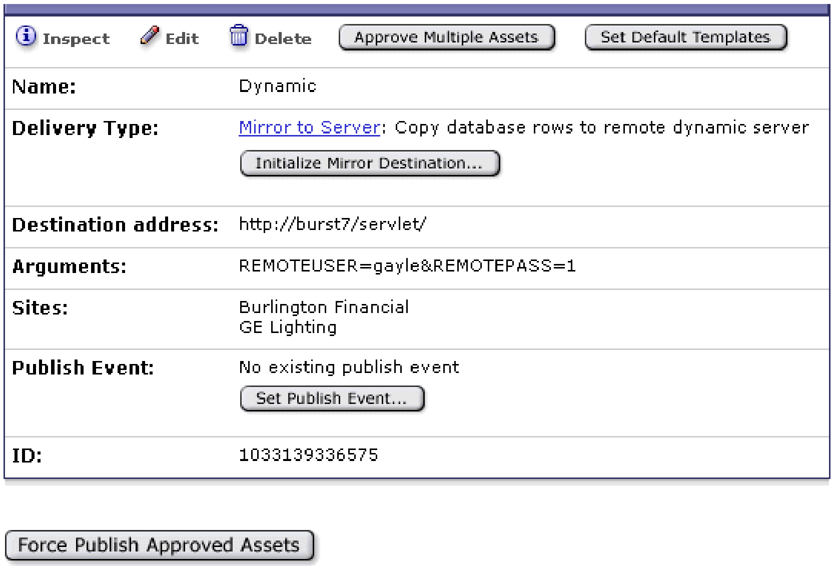
The procedures for creating destinations are described in the previous section. For information about creating new destinations, use one of the procedures in the following list, as appropriate:

- To create an export destination, see "Step 3: Configure an Export Destination," on page 253.
- To create a mirror destination, see "Step 5: Create a Mirror Destination," on page 267.
- To create an export assets destination, see "Step 3: Configure an Export Assets to XML Destination," on page 279.

Editing Destinations

To edit a destination:

1. Select **Admin** > **Publishing** > **Destinations**,
2. Double-click on the destination that you want to edit.



Name:	Dynamic
Delivery Type:	Mirror to Server: Copy database rows to remote dynamic server
Destination address:	http://burst7/servlet/
Arguments:	REMOTEUSER=gayle&REMOTEPASS=1
Sites:	Burlington Financial GE Lighting
Publish Event:	No existing publish event
ID:	1033139336575

3. Click the **Edit** icon, and then make the appropriate changes.

   For information about the values that can be entered in the fields in this form, see the delivery-type specific procedures in the previous sections:

- For an Export to Disk destination, see “Step 3: Configure an Export Destination,” on page 253.
- For a Mirror to Server destination, see “Step 5: Create a Mirror Destination,” on page 267.
- For an Export Assets to XML destination, see “Step 3: Configure an Export Assets to XML Destination,” on page 279.

Deleting Destinations

To delete a destination, complete the following steps:

1. In the Content Server interface, select **Admin** > **Publishing** > **Destinations**.
2. Double-click on the destination that you want to delete.
3. Click the **Delete** icon.

   Content Server displays a verification message.

Creating Export Starting Points

Export starting points are defined in “Export Starting Points,” on page 250. For information about creating them, see “Step 5: Create the Export Starting Points,” on page 255.

Scheduling Publish Events

You set up publishing events for all of your publishing destinations (destinations). Because publish events run as background processes and there is no publish queue, you can set up publish events for different destinations that run at the same time, if necessary.

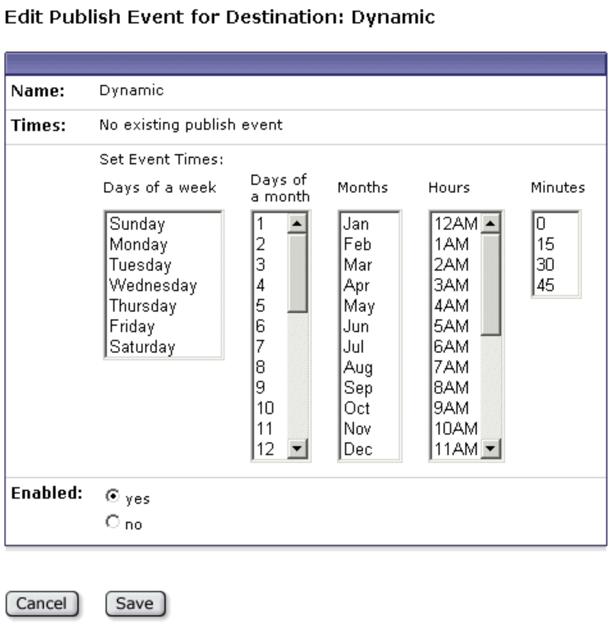
Note the following:

- There can be only **one** publish event for **each** destination.
- If any of your systems serve as both source and destination, be sure that incoming and outgoing publishing sessions **do not overlap**. For example, you publish from the development system to the management system and you publish from the management system to the delivery system. Never schedule a publish event from the development system to the management system for a time when the management system is publishing to the delivery system.

To schedule a publish event, complete the following steps:

1. In the Content Server interface, select **Admin** > **Publishing** > **Destinations**.
2. Double-click on the destination that you want to schedule a publish event for.
3. In the “Publish Destination” form, click the **Set Publish Event** link.

   Content Server displays the “Publish Event” form:



Edit Publish Event for Destination: Dynamic

Name: Dynamic

Times: No existing publish event

Set Event Times:

Enabled: ● yes ○ no

Now you select days, months, hours, and minutes to set a schedule.

Example Schedule

For example, if you want to set a schedule so that Content Server publishes the assets approved for this destination at 7 a.m., 4 p.m., and 8 p.m. every day of the week except Sunday, you would complete these steps:

1. In the **Days of a week** field, click on **Monday** and shift-click on **Saturday** to select all days of the week except Sunday.
2. Skip the **Days of a month** field, since you have already selected the days you want.

3. In the **Months** field, click on **January** and shift-click on **December** to select all months.
4. In the **Hours** field, click on **7am**, scroll down the list, control-click on **4pm,** and control-click again on **8pm**.
5. In the **Minutes** field, click on 0.
6. Under Enable, select **yes**.
7. Click the **Save** button.

   A summary of the schedule now appears in the **Publish Event** section of the “Publish Destination” form.

Reading the Schedule Abbreviations

The abbreviations for scheduled events are displayed in several places in the Content Server interface. Here is the key:

```
hours:minutes:seconds weekdays/daysOf Month/months
```

This information is displayed as follows:

- Hours are displayed as numbers from 0 (midnight) through 23 (11 p.m.) and are separated from the minutes with a colon.

  If the schedule describes more than one hour, they are displayed with commas separating them.

  The previous example (see “Example Schedule,” on page 287), was for 7 a.m., 4 p.m., and 8 p.m. as the times to publish, which is displayed as **7,16,20:**
- Because you can set minutes in increments of 15, minutes are displayed as numbers 0, 15, 30, or 45 and are separated from the seconds with a colon.

  If the schedule describes more than one minutes increment, they are listed with commas separating them.

  Minutes are displayed as a **0**, which means the event runs on the hour.

  The hours and minutes for the example are displayed like this: **7,16,20:0:**
- Seconds are always set to zero. Therefore, the complete expression of the time in the display of the example is: **7,16,20:0:0**
- The time and the days are separated with a space.
  - Days of the week are expressed as numbers 0 (Sunday) through 6 (Saturday) and ending with the slash (/) character.
  - If no days were selected, it means to run every day and that is displayed with the asterisk (*) character.
  - If more than one day is scheduled, they are displayed with commas separating them.

  The example schedule includes all the days of the week except Sunday, which means days of the week are appended to the display as follows: **7,16,20:0:0 1,2,3,4,5,6/**
- Days of the month are also displayed as numbers (1-31) and end with the slash (/) character.
  - The list of days is separated with a comma.

- If no days of the month were selected, it means that they are all selected and it is displayed as the asterisk (*) character.
- If you specify both days of the week and days of the month, the event runs when either setting matches the current day.

The example schedule does not specify days of the month, which means a value for this item is appended to the display as an asterisk:

7,16,20:0:0 1,2,3,4,5,6/*/

- At the end of the string is the months, displayed as numbers from 1 (January) through 12 (December).
  - The list of months is separated with a comma.
  - If all the months are selected, it is displayed as the asterisk (*) character.

  The example schedule specified all the months. Therefore, the final result is:

  7,16,20:0:0 1,2,3,4,5,6/*/*

Editing Publish Events

To edit a publish event, complete the following steps:

1. In the Content Server interface, select **Admin > Publishing > Destinations**.
2. Double-click on the destination that you want to schedule a publish event for.
3. In the “Publish Destination” form, click the **Set Publish Event** link.

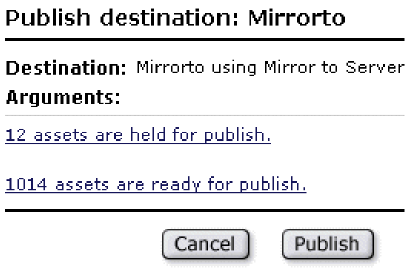
   Content Server displays the “Publish Event” form. The text at the top of the form describes how the event is currently configured.
4. Edit the event as needed and click **Schedule**. To clear your selections, click **Cancel**.

Overriding the Schedule

If you want to start a publishing session immediately, complete the following steps:

1. In the Content Server interface, click the **Publishing** button in the toolbar.
2. In the **Publishing Console**, from the drop-down list, select the destination that you want to publish to.
3. Click the **Select Destination** button.

   Content Server determines whether there are any assets that can be published to the destination and displays a summary form like this one:



Publish destination: Mirrorto

Destination: Mirrorto using Mirror to Server
Arguments:

12 assets are held for publish.

1014 assets are ready for publish.

If there are assets that can be published to the destination, the **Publish** button is displayed. If there are no assets to publish, there is no **Publish** button.

4. Click **Publish**.

   Content Server starts the publishing session.

   Note that if a publishing session for this destination is already underway, the publish event that you just tried to run fails and Content Server displays a status message describing that another session for the same destination is in progress. If you still want to run this event, wait until the current session completes and then repeat this procedure.

Assigning Approval or Preview Templates

When assets are approved for a static publishing destination, the approval system examines the template assigned to the asset to determine its dependencies.

However, when the system publishes the asset, it does not necessarily use the template that is assigned to the asset. Why? Because the code in another element could determine that a different template is used for that asset in certain cases.

Consider the Burlington Financial sample site. An article asset from this sample site can be rendered by several different templates, depending on the context.

So when someone approves an article asset for the Burlington Financial sample site, which template should the approval process use to determine the dependencies for the article? The one that contains the most representative set of dependencies for all of the templates.

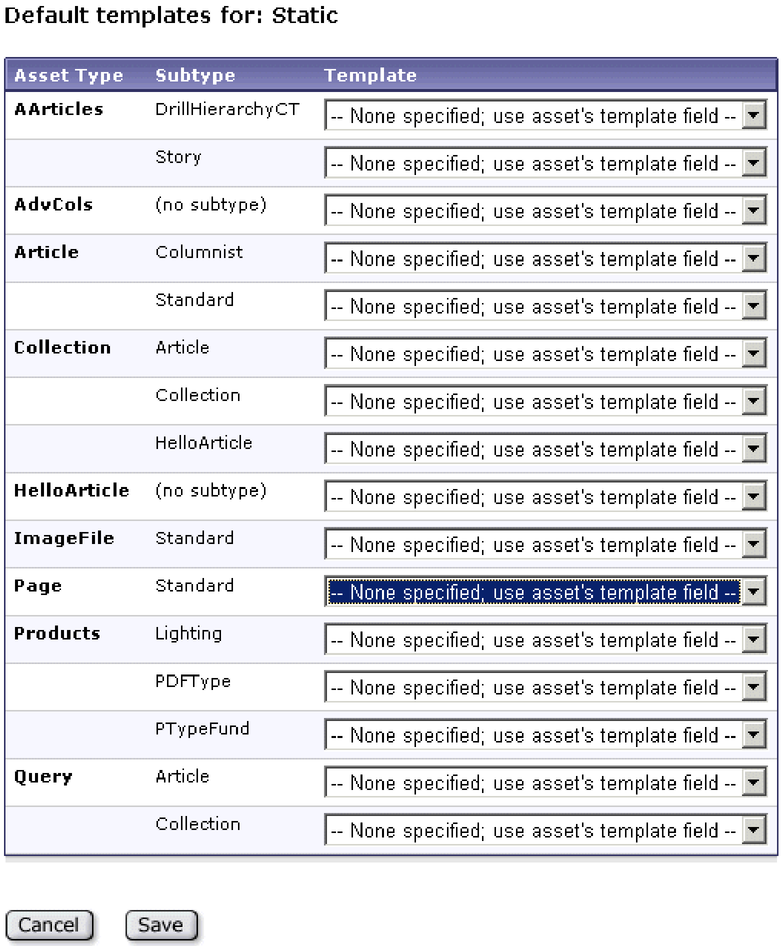
What if the template that contains the most representative set of dependencies is not the template that you want to assign to the asset? Set it as the Default Approval Template for assets of that type.

Note that when you assign default templates to assets that are published to mirror destinations, those templates are not used to calculate dependencies, but they are used when someone previews the asset from the “Status” form.

To set a default template, complete the following steps

1. In the Content Server interface, select **Admin** > **Publishing** > **Destinations.**
2. Expand the destination that you want to configure default templates for and then select the Set Default Templates item.
3. Content Server displays a “Default Templates” form.
4. Click **Edit**.

   The system displays a form like this one:



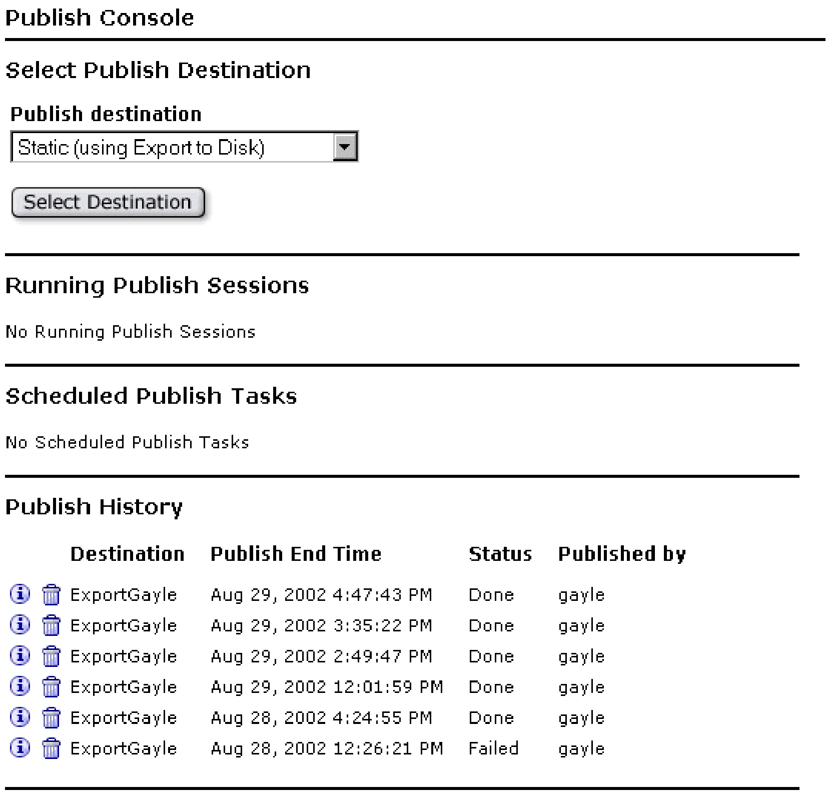
Default templates for: Static

Asset Type	Subtype	Template
AArticles	DrillHierarchyCT	-- None specified; use asset's template field --
	Story	-- None specified; use asset's template field --
AdvCols	(no subtype)	-- None specified; use asset's template field --
Article	Columnist	-- None specified; use asset's template field --
	Standard	-- None specified; use asset's template field --
Collection	Article	-- None specified; use asset's template field --
	Collection	-- None specified; use asset's template field --
	HelloArticle	-- None specified; use asset's template field --
HelloArticle	(no subtype)	-- None specified; use asset's template field --
ImageFile	Standard	-- None specified; use asset's template field --
Page	Standard	-- None specified; use asset's template field --
Products	Lighting	-- None specified; use asset's template field --
	PDFType	-- None specified; use asset's template field --
	PTypeFund	-- None specified; use asset's template field --
Query	Article	-- None specified; use asset's template field --
	Collection	-- None specified; use asset's template field --

5. For each asset type that you need to configure, select the template that you want Content Server to use as the approval or preview template for assets of that type. If there are subtypes configured for an asset type, you can specify a default template for each subtype of that asset type.
6. Click the **Save** button.

Monitoring Sessions in the Publishing Console

You use the **Publish Console** to monitor publishing sessions. To open the **Publish Console**, click the **Publishing** button at the top of the Content Server interface:



Publish Console

Select Publish Destination

Publish destination

Static (using Export to Disk) ▼

Select Destination

Running Publish Sessions

No Running Publish Sessions

Scheduled Publish Tasks

No Scheduled Publish Tasks

Publish History

	Destination	Publish End Time	Status	Published by
ⓘ 🗑	ExportGayle	Aug 29, 2002 4:47:43 PM	Done	gayle
ⓘ 🗑	ExportGayle	Aug 29, 2002 3:35:22 PM	Done	gayle
ⓘ 🗑	ExportGayle	Aug 29, 2002 2:49:47 PM	Done	gayle
ⓘ 🗑	ExportGayle	Aug 29, 2002 12:01:59 PM	Done	gayle
ⓘ 🗑	ExportGayle	Aug 28, 2002 4:24:55 PM	Done	gayle
ⓘ 🗑	ExportGayle	Aug 28, 2002 12:26:21 PM	Failed	gayle

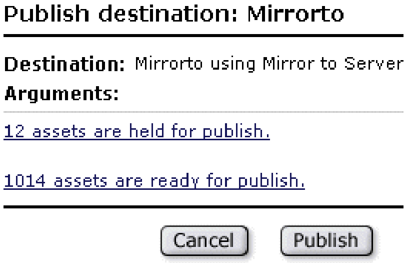
The console displays a summary of all the publishing activity, including any sessions that are currently running, scheduled, or completed.

Verifying Publishing Readiness

To determine whether a publishing session is ready or whether there are any problems with unapproved assets, follow these steps:

1. In the Publish Console, select the destination from the drop-down list.
2. Click the **Select Destination** button.

   A summary form appears. It lists any assets that need to be published but are not ready (they are held) and assets that can be published. For example:



Publish destination: Mirrorto

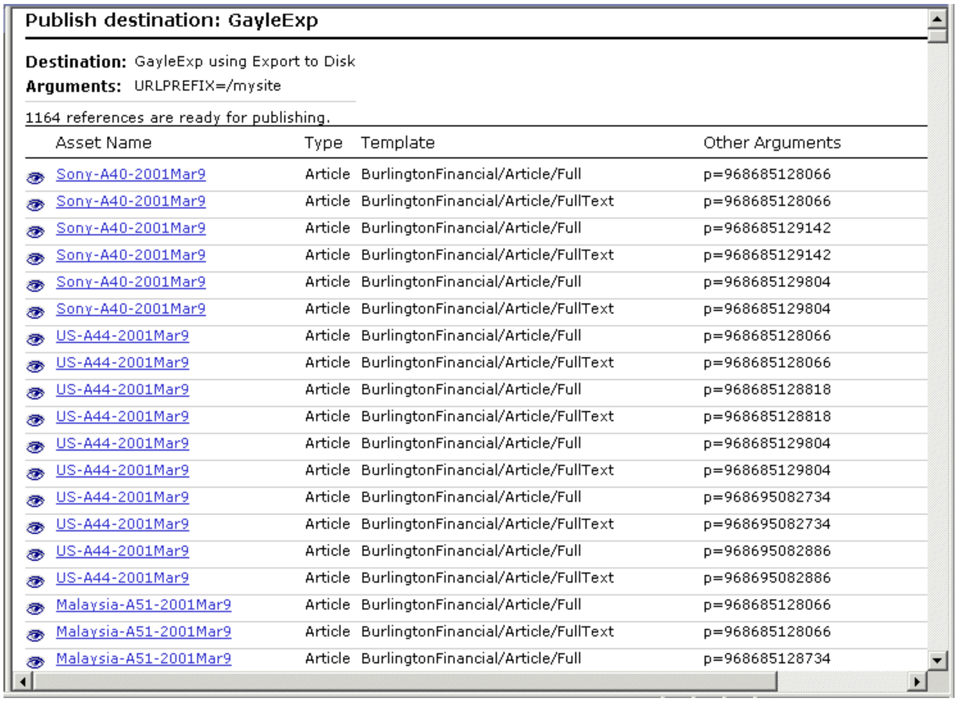
Destination: Mirrorto using Mirror to Server
Arguments:

12 assets are held for publish.

1014 assets are ready for publish.

3. To see the details, click on the link.

   Content Server displays a list of the assets that are either approved or approved and still held for dependency reasons. For example:



Publish destination: GayleExp

Destination: GayleExp using Export to Disk
Arguments: URLPREFIX=/mysite

1164 references are ready for publishing.

Asset Name	Type	Template	Other Arguments
Sony-A40-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685128066
Sony-A40-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968685128066
Sony-A40-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685129142
Sony-A40-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968685129142
Sony-A40-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685129804
Sony-A40-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968685129804
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685128066
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968685128066
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685128818
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968685128818
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685129804
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968685129804
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968695082734
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968695082734
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968695082886
US-A44-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968695082886
Malaysia-A51-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685128066
Malaysia-A51-2001Mar9	Article	BurlingtonFinancial/Article/FullText	p=968685128066
Malaysia-A51-2001Mar9	Article	BurlingtonFinancial/Article/Full	p=968685128734

   If you need to resolve an issue, click the link to the asset and make any changes that are necessary (and perhaps approve it):

- For information about why assets can be held, see page 225.
- For procedures that describe how to work with assets, see the *Content Server User's Guide*.

Managing Publishing History Information

Publishing sessions that are completed are listed in the **Publish History** section of the **Publish Console**, which is displayed when you click the **Publishing** button. Each publish session in the list has a status; Running, Done, or Failed.

Each item in the list has two icons:

- The inspect icon (the circled letter "i"), which displays a summary of that session. Examining the summary is the first step in troubleshooting a failed session.

- The delete icon (the trash can), deletes the session. When you click this icon, Content Server deletes the row from the `PubSession` table, the publishing log file for the session, and any messages for the session from the `PubMessage` table.

  Note that the greater the number of publishing sessions listed in this section, the longer it takes Content Server to open the Publish Console.

If a session shows its status to be Failed, click the inspect icon, note any error messages, and then consult "Troubleshooting," on page 295.

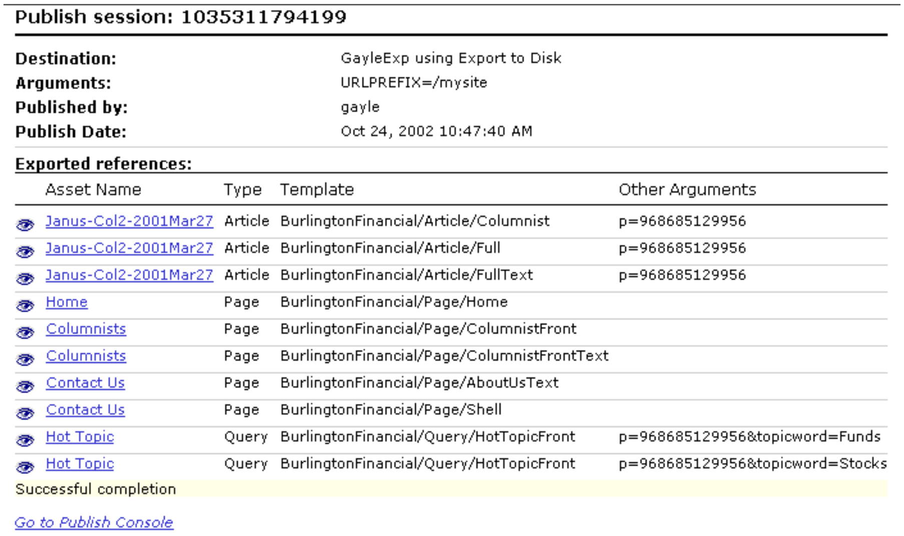
About Session History

The publishing history log files are displayed in the Publishing Console as summaries of publishing sessions.

The text in a history file is a list of all the **references** that were published during that session. The term reference means something different for each publishing method:

- For dynamic publishing, a reference is an **asset**. Each asset that is published is listed by its asset type and its ID.
- For static publishing and Export Assets to XML, a reference is a **generated file**. Each file that is created is listed by its file name.

Here is an example of a session history for a static publishing session:



Publish session: 1035311794199

Destination:	GayleExp using Export to Disk
Arguments:	URLPREFIX=/mysite
Published by:	gayle
Publish Date:	Oct 24, 2002 10:47:40 AM

Exported references:

Asset Name	Type	Template	Other Arguments
Janus-Col2-2001Mar27	Article	BurlingtonFinancial/Article/Columnist	p=968685129956
Janus-Col2-2001Mar27	Article	BurlingtonFinancial/Article/Full	p=968685129956
Janus-Col2-2001Mar27	Article	BurlingtonFinancial/Article/FullText	p=968685129956
Home	Page	BurlingtonFinancial/Page/Home	
Columnists	Page	BurlingtonFinancial/Page/ColumnistFront	
Columnists	Page	BurlingtonFinancial/Page/ColumnistFrontText	
Contact Us	Page	BurlingtonFinancial/Page/AboutUsText	
Contact Us	Page	BurlingtonFinancial/Page/Shell	
Hot Topic	Query	BurlingtonFinancial/Query/HotTopicFront	p=968685129956&topicword=Funds
Hot Topic	Query	BurlingtonFinancial/Query/HotTopicFront	p=968685129956&topicword=Stocks

Successful completion

Go to Publish Console

If there was a problem with the session, an error message is written to the `PubMessage` table and it is displayed as the history instead. If you see an error message in the history for a publishing session, see the section called "Troubleshooting," on page 295 and attempt to resolve the problem.

Publishing All Approved Assets

The publishing system conducts incremental publishing sessions. That is, during any session, only those assets that have been approved since the last session for this destination are published.

However, you may find a situation in which you need to publish all the approved assets in your entire site. (For example, after data repair on the delivery system.) When this is the case, use the **Force Publish All Items Next Time** button in the “Edit Destination” form:

When you click this button, it is the equivalent of changing the status of all approved assets to that of “never been published.” That means that the next publishing session to the destination will publish all approved assets, regardless of whether they have already been published and have not changed since they were published.

Troubleshooting

This section describes error messages and other system indicators that reveal configuration, system, or data errors and suggests corrective actions for each. It contains the following topics:

- About Publishing System Error Messages
- Numeric Messages
- Other Indicators of System or Configuration Issues

About Publishing System Error Messages

The publishing system reports information about publishing sessions in the following ways:

- Displays a status for each session in the **Publish History** list in the **Publish Console**. If you see a status of “Failed,” “Not Found,” or “false,” there was a problem with the publishing session.
- Writes information about the assets that were published to the log file for the publishing session. When you click the inspect icon for a publishing session in the Publish Console, the history log file for that session is displayed.

  The `request.folder` property in the in the `batch.ini` file defines where the publishing history log files are located. Each log file is named as follows:

  ```
  PubSession ID + Output.html
  ```

  For example, if the PubSession ID for the session you are interested in is 9876544, then the log file is named `9876544Output.html`, and it is located in the directory specified by the `request.folder` property.
- Writes error messages about the publishing session to the `PubMessage` table. These messages are displayed in the **Messages** text area when you examine the publishing history in the Publishing Console for a specific publishing session.

  You can use the PubSession ID to look up messages in the `PubMessage` table about a specific publishing session. Note that when you set the `VERBOSE` publishing argument to true, status messages are also written to the `PubMessage` table.
- Writes other information about the session to the `PubSession` table. For example, to determine what time a publishing session started, look up the session by its PubSession ID in the `PubSession` table.

- Writes error messages to the `futuretense.txt` file on both the **source** and the **destination** (target) systems.

You can also find error messages about failed publishing sessions in the following locations:

- The application server log files on both the source and the destination systems.
- The `stdout` and `stderr` logs on both the source and the destination system.

Numeric Messages

This section contains descriptions and suggested corrective actions for the numeric error messages that the publishing system might report in any of the log files mentioned in the list above.

Note

The following list is not a complete list of all the error messages that your Content Server system can report. For a complete list of all error messages, see the error conditions appendix in the *Developer's Tag Reference*.

-2

This error indicates that either the mirror user name or password is not identified correctly. The publishing system on the source began the publishing process, but the destination system couldn't authenticate the user that is identified as the mirror user.

Corrective Action

Correct the mirror user information. See the procedure "Step 3: Identify the Mirror User to the Source System," on page 266 for help.

-3

This error indicates that the mirror user does not have the correct permissions to save the assets to the Content Server database on the destination system. In other words, the destination system user identified to the source as the mirror user does not have all the ACLs it needs.

Corrective Action

Edit the mirror user's account and assign it the appropriate ACLs.

See the procedure "Step 2: Set Up the Destination System," on page 265 for a list of the ACLs that the mirror user needs.

If you still cannot determine which ACL is missing from the mirror user's account, examine the futuretense.txt file on the destination system. There should be an entry that describes which table the mirror process failed to update and the name of the table may help you determine which ACL is missing.

For example, if the mirror session is failing on a visitor table, it is likely that your mirror user does not have the Visitor or VisitorAdmin ACL assigned to it.

-103

This error indicates that one of the tables whose data is being mirror published does not exist on the destination system. This error typically occurs when there is an asset type on the source that does not yet exist on the destination. It is also possible that someone created a custom table to support a function that was custom designed for your site but that table has not yet been created on the destination system.

Corrective Action

Examine the `futuretense.txt` file on the **destination** system and look for the lines that list the -103 error. The text in the message should mention which table is missing.

Then use the appropriate tool to create the table on the destination system. That is, if it is a basic asset type, use AssetMaker. If it is a flex asset type, use Flex Family Maker. If it is a custom table, use CS-Explorer.

-611

This error message is one of the two generic mirror publishing error messages (the other is -12011). Typically -611 means that there was a problem when the publishing system tried to access the destination system and there should be other error messages reported, too.

Corrective Action

Examine the `futuretense.txt` file and application server logs on the **destination** system to look for additional messages.

-612

This error message indicates that the definition of a mirror destination is incorrect in some way. Perhaps the syntax of the destination address is incorrect or there is a typographical error of some kind.

Corrective Action

Log into the Content Server interface on the **source**, examine the definition of the destination, and determine that it is correct. For help, see "Step 5: Create a Mirror Destination," on page 267.

-12011

This error message is one of the two generic messages (along with -611) which mean that the mirror process failed. It is unlikely that this message will appear without other error messages being reported. Typically the other errors will give more information about what went wrong.

Corrective Action

Examine the messages in the `PubMessage` table and the `futuretense.txt` file on the **source** to look for additional error messages.

-12044

This error indicates that the publishing system could not begin the publishing session because it could not contact the destination to start the session. That is, there was no response from the destination; some part of the destination system is offline.

Corrective Action

Start by verifying that the web server and application server are running on the **destination** system. If they are not, start them. Verify that the two systems can connect to each other through the network (perhaps the network went off-line, for example).

-12045

This error indicates that the publishing system was able to start the session, but it then failed in some way.

Corrective Action

Examine the `futuretense.txt` file on the **destination** system and look for error and exception messages in the log.

-12046

This error indicates that there was a problem at the end of a publishing session when the publishing system began to clean up the temporary tables that it creates for each session. When the cleanup process began, the connection with the destination failed. That is, the source system could get no response from the destination when the publishing process tried to clean up the temporary tables.

Corrective Action

Determine whether the web server and application server on the **destination** system are running. If they are not, start them. Check for network connection problems, as well.

-12047

This error indicates that the publishing system was able to begin the cleanup process but it then failed in some way during the operation.

Corrective Action

Examine the `futuretense.txt` file on the destination and look for error exception messages and exceptions that describe the problem.

-13054

This error occurs for complex and flex assets only. In this case, the publishing system began publishing a complex or flex asset but the data did not reach the destination. It can mean that the data itself was corrupt or that the HTTP request connection failed.

Corrective Action

Check the `futuretense.txt` file on the source system. If the problem was the data itself, there will be additional messages in this log file that can help you determine which asset caused the problem. However, if the HTTP request was dropped before the destination system could respond, there will **not** be an additional message in the log.

-13055

This error occurs for complex and flex assets only. In this case, the data for the flex or complex asset was delivered, but the destination system could not save the asset to its Content Server database.

This message indicates that there is a problem with your data. For example, if you are attempting to publish from more than one source to the same destination, there can be ID collisions and other problems with data integrity that will cause this error.

Corrective Action

Examine the `futuretense.txt` and application server log files on the **destination** system. There is often a stack trace message in the `stderr` log that describes what went wrong.

Other Indicators of System or Configuration Issues

This section describes symptoms your CS system may exhibit when the publishing system discovers configuration errors or system problems.

Publishing Session Does Not End and Displays an Odd Status

When a publishing session cannot begin or end, typically the batch user has not been configured correctly. Examine the **Status** listed for the publishing session. If the **Status** is “Not Found” or “false,” it is likely that your batch host is not configured correctly.

Note that the Initialize Mirror Destination feature does not use the batch user. If you can initialize a destination correctly, but the publishing session has errors, it is likely that the batch user settings are incorrect.

Corrective Action

See “Step 1. Create the Batch User Account (If One Does Not Exist),” on page 251 and verify that you have configured the batch user correctly:

- The batch user identified by the `xcelerate.batchuser` property must have the appropriate read/write privileges. These are listed in that procedure.
- The name of the batch user identified by the `xcelerate.batchuser` property must be spelled correctly and the password identified by the `xcelerate.batchpass` property must be correct.
- The server identified by the `xcelerate.batchhost` property must identify the correct server. This property should identify the **web server** that hosts the source system, **not** the destination, **not** the application server if the application server and the web server are on different boxes, and **not** the load balancer if you are using a load balancer. Note that if the port number is something other than 80, you must also specify the port number. For example: `myserver:7001`

  If you see the message “Cannot retrieve output for publish session” in the publishing history, it is likely that the server name is incorrect.
- The port number for the batch host server must also be identified correctly. If you see the message “Cannot retrieve output for publish session” in the publishing history, it is likely that either the port number is incorrect, or if a port number is not specified, that the default port (80) is incorrect.

The System Stops Altogether

If your system simply stops, this behavior indicates that there may not be enough disk space available for the publishing system to function.

Corrective Action

Check the `stdout` and `stderr` files. If there are any Java write errors, you are out of disk space.

- For static publishing and Export Assets to XML, you need enough disk space in your file system to store all the files that are being generated.
- For dynamic publishing, you need space equal to four times the size of the data that is being mirrored available on both the source and the destination system or the mirror operation will fail.

Note that the publishing process will also fail if the Java temp directory is not large enough.

Pages Are Not Refreshed After the Publishing Session Ends

If you are using Satellite Server—either the co-resident Satellite on a management or development system or a remote Satellite Server for a delivery system—and you notice that cached pages are not being regenerated after a publishing session, it is likely that you have configured Satellite Server incorrectly.

Corrective Action

Examine the `futuretense.txt` file on the **destination** system and look for messages like these:

- "Number of satellite servers must match number of usernames and passwords."

  This message indicates that there is something wrong with the values specified for the `cs.satellitehosts`, `cs.satelliteuser`, and `cs.satellitepassword` properties in the `futuretense.ini` file.
- "-100.FormPoster failed flushing URL."

  This message indicates that either a Satellite servlet wasn't running or that the destination system couldn't reach a Satellite servlet that it tried to reach:

Open the `futuretense.ini` file, examine the properties on the **Satellite** tab, and note the values set for the user names, passwords, and host names. Open the satellite.ini files for the co-resident Satellite Server and your remote Satellite Server applications. Compare the values specified for the `cs.satellitehosts`, `cs.satelliteuser`, and `cs.satellitepassword` properties in the `futuretense.ini` file to the values set for the `username` and `password` properties in the `satellite.ini` files. **They must match**.

Remember that the order in which you specify host names in the `futuretense.ini` file must match the order in which you identify user names and passwords.

DB2 Systems, Troubles When Publishing Assets with Associations

When you notice that there are problems with publishing assets that have associations and your system uses a DB2 database, it is likely that the `LOCKLIST` parameter is not set correctly.

Corrective Action

The Content Server installation guide that describes DB2 installations recommends that this property be set to at least 1000. If you find that you are having difficulties publishing assets with associations, increase this value.

Part 4
System Configuration Procedures

This section describes the configuration options that affect all the sites on your CS system. It contains the following chapters:

- Chapter 18, “Configuring the User Interfaces”
- Chapter 19, “Configuring the Lucene Search Engine”
- Chapter 20, “Revision Tracking”

Chapter 18

Configuring the User Interfaces

There are several CS user interface features that must be configured before content providers can use them.

Some of these features are alternate interfaces to the Content Server interfaces: Content Server Desktop, Content Server DocLink, and the InSite interface. Others change the Content Server interfaces (locale settings) and the behavior of the preview function.

This chapter contains the following sections:

Setting the Locale for the Content Server Interfaces

If your organization purchased and installed one of the CS language packs, you must configure a default locale for your CS system to use.

There are several options and properties that you use to set the locale:

- The **Locale** item in the **Admin** tab sets the default locale for the Content Server interfaces. Individual users can override the default and use a different locale (if one is present) by setting a preference.
- The `cs.emailcharset` and `cs.emailcontenttype` properties in the `futuretense.ini` file determine the character set that is used in the e-mail messages that the workflow system sends. These properties—which must be set to a character set that can appropriately represent characters for all of the locales that your content providers can select—are usually set during installation.
- If your system is using the UTF-8 character set and you are using the Mirror to Server publishing delivery type, you must set the value for the `cs.mirrorowsperpost` property in the `futuretense.ini` file to 4 or lower.
- If your system is using the UTF-8 character set and you are using the Burlington Financial sample site's article asset type, you must set the value of `xcelerate.body.length` property in `futuretense_xcel.ini` to 500.

System Default Locale for the CS Users

When a language pack is installed, the installer writes the language and country code for the locale of the language pack to the `LocaleMap` database table. For example, Canadian French is `fr_ca`. When there is more than one locale listed in the `LocaleMap` table, you, as the administrator, need to specify which should be the default locale.

You can override the default locale for individual users by setting it in their user profile. See "Creating and Editing a User Profile," on page 82 for information about how to set a user's locale preference.

Content Server determines which language to use for an individual user through its universal session variable named `locale`. This variable contains the language and country code associated with the user who is logged in.

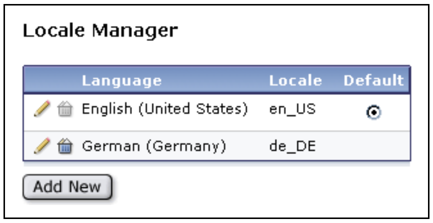
When a user logs in to Content Server, Content Server obtains the value for the `locale` variable as follows:

- If the user's user profile has a value for the locale user attribute, Content Server obtains and uses it.
- If the user profile does not specify a preference, Content Server attempts to match the locale setting of that user's browser to a locale from the `LocaleMap` table.
- If the user profile does not set a preference and Content Server cannot match that user's browser locale, Content Server uses the default locale that is set for the system.

To set the system default locale

1. In the **Admin** tab, double-click **Locale**.

   Content Server displays the “Locale Manager” form.



Locale Manager

Language	Locale	Default
English (United States)	en_US	●
German (Germany)	de_DE	

2. In the “Locale Manager” form, do the following:
   a. Click the **Edit** icon next to the desired locale.
   b. Select the **Default** radio button next to the desired locale.
3. Click **Save**.

System Locale Properties

The properties in the following list should have been configured during the installation of the language pack on your CS system. They are listed here for troubleshooting purposes. For example, if the workflow e-mails are displaying odd characters, perhaps the e-mail properties were not set correctly.

For instructions on how to start and use the Property Editor to verify or modify property values, see the *Property Files Reference*.

Property	Property File	Description
`cs.contenttype`	`futuretense.ini`	Determines the HTTP header character set for all Content Server pages, as well as other HTTP interaction (Content Server Explorer, CatalogMover, and so on). This value is set to `text/html; charset=UTF-8` by default. This value must be set to an appropriate value for the online site that your CS system is delivering.
`cs.emailcharset`	`futuretense.ini`	Determines the character set used in the subject lines of e-mail messages sent by Content Server. (Typically these are workflow e-mails.)
`cs.emailcontenttype`	`futuretense.ini`	Determines the character set used in the body of e-mail messages sent by Content Server.

Property	Property File	Description
`cs.mirrorrowsperpost`	`futuretense.ini`	Specifies the number of table rows that can be mirrored during each HTTP POST while a mirror publish session is underway. If you are using UTF-8 and your database serves non-ASCII text, this value must be set to 4 or lower.
`xcelerate.charset`	`futuretense_xcel.ini`	Determines the HTTP header character set used for all Content Server pages. This value must be set to UTF-8.

Single-Language Restrictions

Although you can configure a multi-lingual management system, certain parts of the user interface can be displayed in one language only.

For example, the names of tables and columns in the Content Server database as well as individual items such as categories and source codes can have one name only. This means that although much of the text on an individual Content Server form can be displayed in any of the languages that you have installed on your CS system, items such as field names and asset type names can be displayed in one language only.

Following is a list of items in CS that can have one name only, which means that they can be displayed in one language only:

- Asset type names
- Field names
- Asset names
- Categories
- Source codes
- Tree tab names
- Site names
- Names of workflow building blocks (actions, e-mail objects, conditions, states, steps, processes)
- Role names
- Start menu items—both Search and New

On a system that provides two or more languages, you must determine which language is going to be used by the majority of content providers and then use that language to name your sites, tabs, asset types, and so on.

Locale and Asset Types

When your site developers create asset types (either basic or flex), they can specify the field names for those asset types in whatever language they have available on the CS system. However, there are several core columns/fields that Content Server creates for all asset types by default:

- id
- name
- description
- status
- createdby
- createddate
- updatedby
- updateddate
- startdate
- enddate
- subtype
- filename
- path
- template
- category

These column names are created in English, by default. When Content Server displays them as fields in its interfaces, it uses the language that is set as the system default language.

For descriptions of the default columns for asset types, see the “Data Design” section in the *Content Server Developer’s Guide*.

Locale and the Article Asset Type

If your site developers decided to use the sample site asset type named Article, you must also modify the value for the `xcelerate.body.length` property in the `futuretense_xcel.ini` file. This property determines how many characters are stored in the `urlbody` column of the `Article` table.

Because `urlbody` is a URL column, the data entered in it is actually stored as a file outside of the Content Server database. However, the first *n* number of characters, where *n* equals the value specified for the `xcelerate.body.length` property, is also stored in the `urlbody` column so that you can search for text in the body of an article asset with the search feature in the Advanced interface.

When this column holds non-ASCII text, the value for this property should be set to 500.

Customizing the Dash Interface

You can customize the following elements of the Dash interface:

- Asset types available in the quick access pane, including their descriptions and icons
- Contents of the “Help and Support” pane
- **Help** link in the top bar
- Client logo in the upper left corner of the navigation pane and on the login page
- Help links (such as **Forgot your password?**) in the login form

These interface elements are configured via properties stored in the `ui.properties` and `ui.propertiesadmin` property files. For detailed information, see the *Property Files Reference*.

Additionally, you can customize the asset type and locale icons displayed in the “Type” and “Locale” columns in the search results pane whenever a user performs a search or runs a tag. For detailed information, see the *Content Server Developer’s Guide*.

CS-Desktop

Content Server Desktop (CS-Desktop) enables Microsoft Word users to create, import, and edit Word documents as Content Server assets using the familiar Word user interface. They save these assets to, and open them from, the Content Server database.

Note

CS-Desktop supports the use of standard Word documents and templates. In Microsoft Word 2000, web page templates that enable Word users to create HTML pages using Word were introduced.

The Word web page templates are neither supported nor needed by CS-Desktop.

Overview

When CS-Desktop is configured for a system, users can log in to Content Server from a special toolbar in the Microsoft Word user interface.

You, the administrator, configure which asset types will be available in the Word interface. You enable the asset types for CS-Desktop and then determine which fields the Word users can enter data for. This information determines the appearance and the functionality of the Content Server toolbar when users create assets of those types in Word. Your content providers use the Content Server toolbar to insert field markers that mark the data for each field.

When someone saves the Word document as an asset, CS-Desktop sends that asset to Content Server. A temporary copy of the document is also saved locally on the user’s hard drive.

The file conversion subsystem converts the binary form of the document into XML, which, in turn, is parsed into a form suitable for storing in the Content Server database. The asset is then managed as follows:

- The Word document is also stored with the asset. That document is retrieved when a user opens the asset in Word.
- When a visitor to your online site (on the delivery system) requests the asset, Content Server serves the asset, not the Word document.

Note that when you examine a CS-Desktop asset in the Content Server interface, you can access its “Inspect” form, but you cannot edit the asset. When the source of the asset is Word, the asset can be edited only in Word.

Sources of Data for a Word Asset

The Microsoft Word document is the complete source for a word asset. There are several ways to enter data for an asset’s field in the Word interface:

- A user enters text and then marks it with the appropriate field marker.
- Certain required fields such as **Name** and **Description** are hidden in Word. For these fields, the user selects the field name from the Content Server toolbar and then enters the data in a dialog box.

  This kind of data is considered to be Word metadata.

  In Word, you can select **File > Properties** and click the **Custom** tab to see these hidden fields and other metadata that results from storing the Word asset in the Content Server database. Do **not** modify this data.
- Obtain information from Word template `.dot` files. You can create Word templates for the Word assets that provide field/value pairs for specific fields. (You can also use Word templates to create a form for your content providers to use to enter information about their Word assets.)

Preserving the Formatting of the Word Data

You, the administrator, determine how the data from an individual field is managed by using the **Preserve formatting** option when you configure asset types for CS-Desktop.

When you select **Preserve formatting** for a field, Content Server converts the Word format to the corresponding HTML format and stores the HTML in the column that represents the field in the Content Server database. Paragraph breaks are converted to `<p>` tags, bold text is marked with `<b></b>` pairs, tables are stored as HTML tables, and so on.

Therefore, it is best if you use the **Preserve formatting** option for fields that contain text only. Do not use the **Preserve formatting** option for fields that contain files (image or other types of files).

It is also best to keep text and images separate—that you do not allow the CS-Desktop users to save both text and images to the same blob attribute or upload field. For example, if you have an article asset type that contains an image with a caption, use one attribute or field for the caption and another attribute or field for the image file.

By keeping text and images in separate fields and controlling whether the formatting is preserved or not, your template writers can know what kind of data will be extracted from a particular field and can code their templates to display that data appropriately. If you decide to allow users to save both text and image to the same blob attribute or upload field, both you and your template developers should note the following:

Format Preserved	Word Content Marked to Be Saved in the Blob Attribute or Upload Field	How the Word Content Is Stored in the Database
Yes	text and image	The image is written to the directory identified by the `keyview.imgdir` property in `futuretense_xcel.ini` and the text is stored as HTML in the URL column; the HTML has an unmanaged link to the image in the `keyview.imgdir` directory. In such a case, you must implement a process that regularly mirrors the `keyview.imgdir` directory to the delivery system because the publishing system does not mirror this directory.
Yes	text only	The text is stored as HTML in the URL column.
Yes	image only	The image file is written to the URL column.
No	text and image	The text is stripped out and the image file is written to the URL column.
No	text only	The formatting of the text is stripped out and the text is written to the URL column.
No	image only	The image file is written to the URL column.

Configuring the Word Asset Types for CS-Desktop

The term “Word asset type” refers to any Content Server asset type configured to be created and edited in Microsoft Word. It is not a new asset type, but rather, an asset type already known to Content Server that you (or another administrator) have enabled for Word (CS-Desktop).

While there are no restrictions on which asset types might fall into this category, practical limits require that you restrict the qualifying asset types to those that have text content or are otherwise document-related.

To configure the asset types for CS-Desktop

- Enable the asset types that you want to make available through CS-Desktop.
- For each subtype for the enabled basic asset types and for each definition for the enabled flex asset types, specify which fields will appear on the Content Server toolbar within the Microsoft Word interface.
- For each field that you enable, determine whether Content Server should retain the formatting of the data that was entered in the Microsoft Word application. For fields that hold blobs, use the formatting option to determine how text and files are managed when they are written to that field.
- Create CS-Desktop start menu items for each of the Word asset types.

The following sections describe each step in detail.

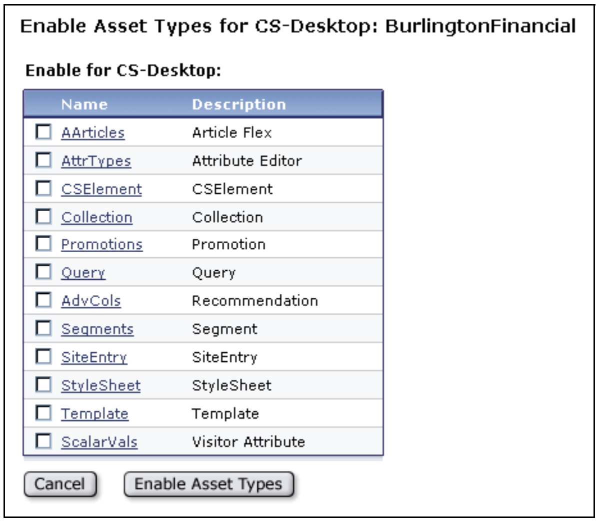
Enabling Asset Types for CS-Desktop

You enable an asset type for CS-Desktop by adding it to those that are selected for CS-Desktop.

To enable asset types for CS-Desktop

1. In the tree, do one of the following:
   - In the **Admin** tab, expand **Sites**, then the site for which you are configuring CS-Desktop.
   - If you have logged in to the site that you are configuring and you have access to the **SiteAdmin** tab, select the **SiteAdmin** tab.
2. Expand **CS-Desktop** and double-click **Enable CS-Desktop**.

   Content Server displays the “Enable Asset Types for CS-Desktop” form, showing asset types that have not yet been enabled for CS-Desktop in this site:



Enable Asset Types for CS-Desktop: BurlingtonFinancial

Enable for CS-Desktop:

	Name	Description
□	AArticles	Article Flex
□	AttrTypes	Attribute Editor
□	CSElement	CSElement
□	Collection	Collection
□	Promotions	Promotion
□	Query	Query
□	AdvCols	Recommendation
□	Segments	Segment
□	SiteEntry	SiteEntry
□	StyleSheet	StyleSheet
□	Template	Template
□	ScalarVals	Visitor Attribute

3. Select the asset types you want to enable for CS-Desktop in this site.
4. Click **Enable Asset Types**.
5. Continue to step 4 in “Specifying Which Fields Are on the Content Server Toolbar,” on page 313.

Specifying Which Fields Are on the Content Server Toolbar

Before you begin this procedure, be sure that you have enabled the asset type(s) that you want to configure for CS-Desktop. If you have not yet enabled the asset type(s), see “Enabling Asset Types for CS-Desktop,” on page 313.

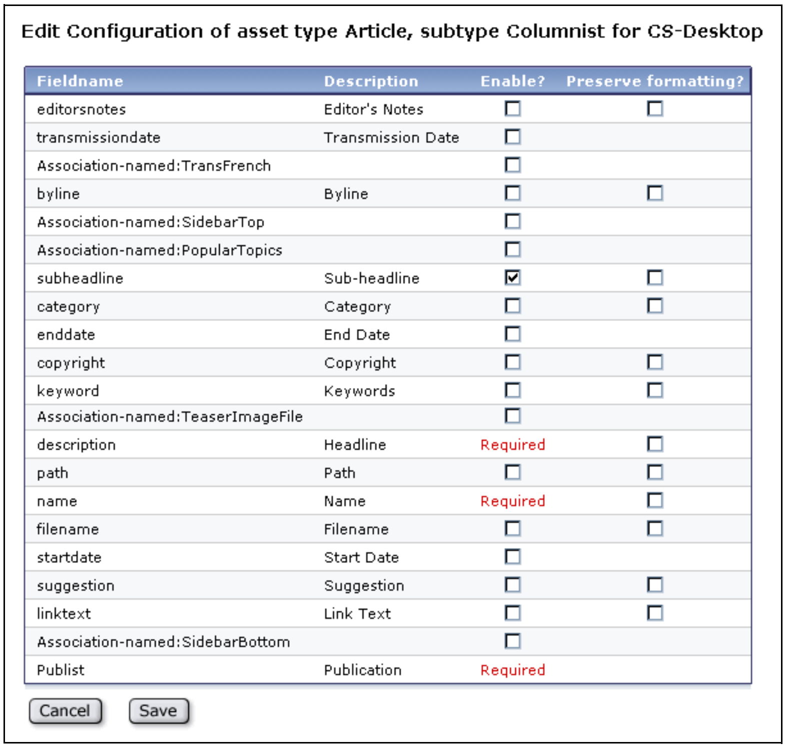
To specify which fields of an asset type will be displayed on the CS toolbar in the MS Word interface

1. In the tree, do one of the following:
   - In the **Admin** tab, expand **Sites**, then the site for which you are configuring CS-Desktop.

- If you have logged in to the site that you are configuring and you have access to the **SiteAdmin** tab, select the **SiteAdmin** tab.

2. Expand **CS-Desktop**, then the asset type you want to configure.
3. Double-click the desired subtype (basic asset types) or definition (flex and flex parent asset types), and click **Edit Configuration**.

   Content Server displays a form similar to this one:



Edit Configuration of asset type Article, subtype Columnist for CS-Desktop

Fieldname	Description	Enable?	Preserve formatting?
editorsnotes	Editor's Notes	☐	☐
transmissiondate	Transmission Date	☐	
Association-named:TransFrench		☐	
byline	Byline	☐	☐
Association-named:SidebarTop		☐	
Association-named:PopularTopics		☐	
subheadline	Sub-headline	☑	☐
category	Category	☐	☐
enddate	End Date	☐	
copyright	Copyright	☐	☐
keyword	Keywords	☐	☐
Association-named:TeaserImageFile		☐	
description	Headline	Required	☐
path	Path	☐	☐
name	Name	Required	☐
filename	Filename	☐	☐
startdate	Start Date	☐	
suggestion	Suggestion	☐	☐
linktext	Link Text	☐	☐
Association-named:SidebarBottom		☐	
Publist	Publication	Required	

4. In the "Edit Configuration" form, select the **Enable** check box for each field you want to appear in the CS-Desktop toolbar in Word. As a general rule, select only text-entry fields. Note that required fields are pre-elected and cannot be deselected.
5. (Optional) If you want the format of the text that is entered in a field through Word to be preserved when that data is stored in the Content Server database, select the **Preserve formatting** option for that field.

   As a general rule, enable the **Preserve formatting** option only for fields that store text. Do not enable the **Preserve formatting** option for fields that hold uploaded files.
6. Click **Save**.
7. (Optional) Repeat steps 3–6 for each additional subtype or definition you want to configure.
8. (Optional) Repeat steps 2–7 for each additional asset type you want to configure.

Creating CS-Desktop Start Menu Items

The subtype configuration determines which fields are available on the Content Server toolbar in the Microsoft Word interface, but your Word asset types do not become available in Word until you create start menu items for them.

To create CS-Desktop start menu items

1. In the **Admin** tab, expand **Asset Types**, then the desired asset type.
2. Under the selected asset type, expand **Start Menu** and double-click **Add New**.
3. In the **Name** field, enter a short, descriptive name for the item. This is the name that is displayed in the list of asset types that are available from the CS-Desktop toolbar in Word.
4. In the **Description** field, enter a short, informative description for the start menu item.
5. In the **Type** field, select **CS-Desktop**.

> **Note**
>
> You cannot set default field values for assets whose source is Word. Ignore the **Default Values** section when creating start menu items for Word asset types.

6. In the **Sites** field, select which sites can use this start menu item. Use **Ctrl-click** to select more than one site.
7. In the **Roles** field, select all the roles that can have access to the start menu item. Use **Ctrl-click** to select more than one role.
8. (Optional) To configure workflow process details for the assets that are created with this start menu item, complete the following steps:
   a. Click **Select Workflow**.
   b. In the “Select Workflow” form, select the appropriate workflow process from the drop-down list and click **Select Workflow**.
   c. In the “Set Participants” form, select at least one user for each role that appears and then click **Set Participants**.
   d. Click **Continue**.

   Content Server saves your changes and redisplays the “Start Menu” form.
9. In the “Start Menu” form, click **Save**.
10. Repeat steps 1–9 for each additional asset type you want to configure.

User Accounts and CS-Desktop

Like the content providers who use the Content Server interfaces, the content providers who use the CS-Desktop application must have Content Server user accounts as well. User accounts for the individuals who use CS-Desktop must have the following ACLs:

- Browser
- ElementReader
- PageReader
- RemoteClient
- xceleditor
- UserReader
- Visitor (if you have Engage installed)

For information about creating user accounts, see "Editing a User," on page 80, and "Required ACLs for Custom Users," on page 360.

Give the CS-Desktop users the following information:

- Their user name/password combination
- The appropriate URL for logging in to your CS management system

They need both pieces of information to log in to Content Server through CS-Desktop.

The Content Server URL uses the following convention:

```
http://<server>:<port>/<context>
```

where:

- `<server>` is the host name or IP address of the machine running Content Server.
- `<port>` is the port on which Content Server is listening for connections.
- `<context>` refers to the application context root (URI) assigned to the Content Server application when it was installed.

Installing the CS-Desktop Client Application

In order to use the CS-Desktop feature, the CS-Desktop client must be installed on the client machine.

> **Note**
>
> Before installing CS-Desktop, make sure that the client machine:
>
> - Is running supported versions of Microsoft Windows and Word, and
> - Supports the entry and display of text using the UTF-8 character set.

To install the CS-Desktop client

1. Point your browser to the following URL:

   ```
   http://<server>:<port>/<context>/DownloadPage.html
   ```

   where:

   - `<server>` is the host name or IP address of the machine running Content Server.
   - `<port>` is the port on which Content Server is listening for connections.
   - `<context>` refers to the application context root (URI) assigned to the Content Server application when it was installed.

2. On the "FatWire Content Server Applications" page, click **Download** in the "CS-Desktop" row.

   > **Note**
   >
   > If one or more language packs are installed on your CS system, the page lists localized versions of the CS-Desktop client corresponding to the installed language packs. In such case, download the version appropriate to the user's language preference.

3. Extract the downloaded archive to a temporary directory and open that directory.
4. Launch the CS-Desktop installer (`setup.exe`) and follow the instructions it displays.

Specifying Locale for the Client Application

The Download page is generated dynamically based on the language packs that are installed on your CS system. If your CS system has a language pack installed, the Download page presents a CS-Desktop option in that language and the csdesktop.exe file installs a version of the help file in that language.

The first time a CS-Desktop user logs in to Content Server, the Log In dialog box is in English. The user clicks the Select Language link to select the language that matches the version of CS-Desktop help file that he or she installed.

After a user selects a language, CS-Desktop copies the appropriate XML language file from Content Server to that user's hard drive. CS-Desktop uses the information in that language file for menu names and dialog box text until the user selects a different language.

Additionally, whenever the language file on the Content Server server is updated, be sure to remind your CS-Desktop users to update their language file on their hard disks by using the Select Language link again the next time they log in to Content Server.

Testing the CS-Desktop Configuration

To determine whether your CS-Desktop application is configured correctly, you must create and save an asset from the Microsoft Word interface and then examine it through Content Server. Complete the following steps:

1. Open Microsoft Word.
2. On the button bar, select **Content Server** > **Login**.
3. In the **Login** dialog box, enter the following information:
   - The name and password of a user who has the RemoteClient ACL.
   - The server URL, as follows:

     ```
     http://<server>:<port>/<context>
     ```

     where:

     - `<server>` is the host name or IP address of the machine running Content Server.
     - `<port>` is the number of the port on which Content Server is listening for connections.
     - `<context>` refers to the application context root (URI) assigned to the Content Server application when it was installed.
4. Select the site, asset type, and subtype, as necessary, to set the field markers list.
5. Enter text in the Word document.
6. Select (highlight) the entire range of text.
7. Click the **Field Markers** button on the toolbar and select the marker for the field that you are entering text for.

8. Repeat steps 5–7 for all required asset fields that require you to enter text in the Word document.
9. Set values for the field markers with ellipses (…). These field markers represent hidden fields; that is, their values do not appear in the Word document, but are part of the asset definition, and are thus stored in the Content Server database. When you click a field marker with an ellipsis, a dialog box opens. Enter the appropriate data.
10. In the button bar, select **Content Server**, then **Save to Content Server**.
11. Log in to the Advanced interface as a user who has access to the type of asset that you just created.
12. Select the site that you created the asset for and then click **Search**.
13. Find the asset that you entered through CS-Desktop.
14. Click **Inspect**. (Do **not** select **Edit**.)
15. Examine the data listed for the fields and verify that the data that you marked in CS-Desktop matches the data displayed in the "Inspect" form.

Configuring Word Templates for the Word Assets

The field markers on the Content Server toolbar are Word bookmarks. Therefore, you can create `.dot` files for your Word assets, marking the appropriate areas of those templates with either the Content Server field markers or your own Word bookmarks.

Create a new Word document and mark placeholder text with the appropriate field markers, just as you would for an asset of that type (that is, be sure that you do not mark text for the metadata fields such as **Name** and **Description** and so on). Then save the document as a template (a `.dot` file) rather than to Content Server.

CS-DocLink

Content Server DocLink (CS-DocLink) provides a drag-and-drop interface for uploading and downloading documents, graphics, or other files that are managed as flex assets by Content Server. This application presents the hierarchical data structure of the flex parents and flex assets in your Content Server database as folders and files in the Windows Explorer application.

Overview

CS-DocLink enables your content providers to use their third-party tools to create content for your online sites, and then save it to the Content Server database without having to log in to the Content Server interfaces.

CS-DocLink enables content providers to drag and drop Word files, Excel spreadsheets, graphic files, and so on—that is, single binary files—into the Content Server database by dropping each file onto the appropriate Microsoft Windows Explorer folder that represents the asset type or parent asset type that the document should be saved as.

After a content provider drops the file onto a folder, CS-DocLink displays a dialog box that prompts the user to enter any metadata that is required, depending on how you, the administrator, have configured your system. Examples of such metadata include name, description, and so on.

CS-DocLink differs from CS-Desktop for Word as follows:

- With CS-Desktop, content providers produce the content and tag the data in that document that should be inserted into various fields from within the Word interface. The tagging process creates a structured asset when the document is stored in the database as an asset.
- With CS-DocLink, content providers produce their content in their third-party applications but they do not use those applications to provide structure to the data.

  They drop a binary file into the Windows Explorer interface, and then provide information about the file through the CS-DocLink dialog boxes that appear in response to the drop. The information required depends on the kind of asset type the file is going to be stored as, based on the place in the graphical representation of the database that the user dropped the file.

Configuring the CS-DocLink Asset Types

The term “CS-DocLink asset type” refers to a flex or flex parent asset type that is configured to be accepted by CS-DocLink. CS-DocLink asset types must be flex asset types that store an uploaded, binary file. That is, one (and **only** one) of the flex attributes that define the asset **must be** of type **blob**.

To configure the CS-DocLink asset types

- Verify that the flex or flex parent asset type is enabled for the site. Be sure that the asset type has **one** flex attribute of type **blob**.
- Enable the flex or flex parent asset type for CS-DocLink.
- For each definition for those enabled asset types, specify which fields will be available in CS-DocLink and specify which field is the upload field.
- Create CS-DocLink start menu items for each of the CS-DocLink asset types.

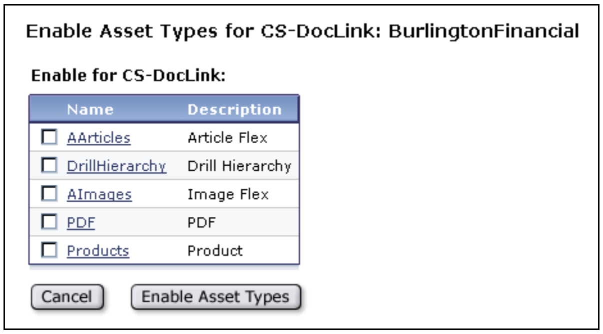
Enabling Asset Types for CS-DocLink

You enable an asset type for CS-DocLink by adding it to those that are selected for CS-DocLink.

To enable asset types for CS-DocLink

1. In the tree, do one of the following:
   - In the **Admin** tab, expand **Sites**, then the site for which you want to enable CS-DocLink.
   - If you have logged in to the site that you are configuring and you have access to the **SiteAdmin** tab, select the **SiteAdmin** tab.
2. Under the selected asset type, expand **CS-DocLink**, then double-click **Enable CS-DocLink**.

Content Server displays the "Enable Asset Types" form, listing asset types that have not yet been enabled for CS-DocLink in the selected site.



Enable Asset Types for CS-DocLink: BurlingtonFinancial

Enable for CS-DocLink:

Name	Description
□ AArticles	Article Flex
□ DrillHierarchy	Drill Hierarchy
□ AImages	Image Flex
□ PDF	PDF
□ Products	Product

3. Select the check boxes next to the asset types that you want to enable for CS-DocLink in the selected site.
4. Click **Enable Asset Types**.

   The enabled asset types appear in the **Admin** tab under the CS-DocLink node for the selected site.
5. Continue to step 3 in "Specifying Which Fields Are on the Content Server Toolbar," on page 313.

Specifying Which Fields Are Available in CS-DocLink

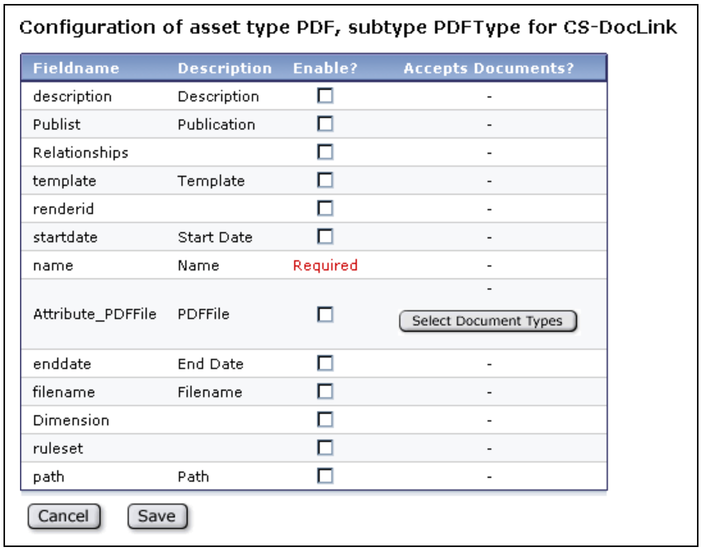
Before you begin this procedure, be sure that the following conditions are true:

- You have enabled the asset type that you want to configure for CS-DocLink. If you have not yet enabled the asset type, see "Enabling Asset Types for CS-DocLink," on page 319.
- The asset type that you want to configure has one attribute of type blob.

To specify which of an asset type's fields will be available to content providers who are using CS-DocLink to create or edit assets of that type

1. In the tree, do one of the following:
   - In the **Admin** tab, expand **Sites**, then the site for which you are configuring CS-DocLink.
   - If you have logged in to the site that you are configuring and you have access to the **SiteAdmin** tab, select the **SiteAdmin** tab.
2. Under the selected site, expand **CS-DocLink**, then the desired asset type and subtype.
3. Click **Edit Configuration**.

Content Server displays a form similar to this one:



Configuration of asset type PDF, subtype PDFType for CS-DocLink

Fieldname	Description	Enable?	Accepts Documents?
description	Description	□	-
Publist	Publication	□	-
Relationships		□	-
template	Template	□	-
renderid		□	-
startdate	Start Date	□	-
name	Name	Required	-
Attribute_PDFFile	PDFFile	□	- Select Document Types
enddate	End Date	□	-
filename	Filename	□	-
Dimension		□	-
ruleset		□	-
path	Path	□	-

4. In the “Configuration of Asset Type” form, select the **Enable** option for each field that you want to be displayed in the CS-DocLink interface. An enabled field means that users can view and edit the data in that field.

   As a general rule, enable the following kinds of fields:

   - The blob attribute that will hold the file (just one)
   - Text entry fields (**Description**, for example)

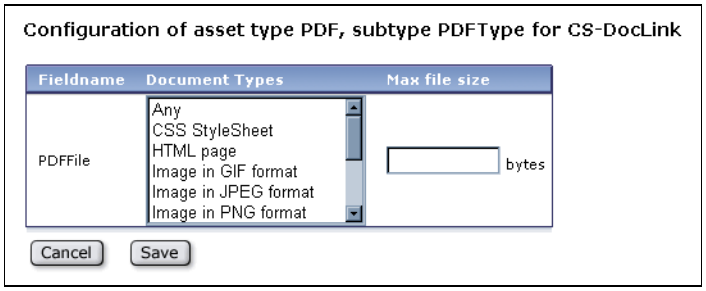
   Do **not** enable the **Publist** field—assets cannot be shared with other sites through the CS-DocLink interface.

   Note that users are prompted to fill in the data for required fields when they create an asset—whether or not you have selected the **Enable** option for those fields. Therefore, if you do not want users to be able to edit the data in a required field for existing assets of this type, do not enable the field.

5. Click **Select Document Types** next to the upload field (flex attribute) you want to specify as the field in which documents are to be stored.

> **Note**
>
> If your developers have created a flex filter for assets of this type, the flex attribute that you specify as the upload field must match the field that is designated as the input attribute for the flex filter.
>
> For information about flex filters, see the *Content Server Developer’s Guide* and talk to your developers.

Content Server displays a form similar to this one:



Configuration of asset type PDF, subtype PDFType for CS-DocLink

Fieldname	Document Types	Max file size
PDFFile	Any CSS StyleSheet HTML page Image in GIF format Image in JPEG format Image in PNG format	bytes

6. In the **Document Types** field, select the types of documents that this upload field can store. Use **Ctrl-click** to select more than one.

   The values displayed in the **Document Types** field come from the `description` column of the entries in the `MimeType` table. If the document type that you want to select does not appear in the list, use Content Server Explorer to add it to the `MimeType` table.

> **Note**
>
> When you select the **Any** option, it means that CS-DocLink accepts any type of file for assets of this type, whether or not there is an entry for it in the `MimeType` table.
>
> However, when there is no entry for a file type in the `MimeType` table, a file of that type cannot be displayed correctly in the Content Server interfaces.

7. In the **Max file size** field, specify a size limit (if there is one) for the documents that can be stored in this upload field.
8. Click **Save**.
9. In the "Configuration of Asset Type" form, click **Save** again.
10. Repeat steps 2–8 for each additional subtype or definition you want to configure.
11. Repeat steps 2–10 for each additional asset type you want to configure.

Creating CS-DocLink Start Menu Items

The CS-DocLink configuration determines which fields are available in the CS-DocLink interface, but your content providers cannot create assets of this type in CS-DocLink until you create start menu items for those asset types.

To create CS-DocLink Start Menu items

1. In the **Admin** tab, expand **Asset Types**, then the desired asset type.
2. Under the selected asset type, expand **Start Menu** and double-click **Add New**.
3. In the **Name** field, enter a name for the item.
4. In the **Description** field, enter a short, informative description for the start menu item.

5. In the **Type** field, select **CS-DocLink**.

> **Note**
>
> You cannot set default field values for assets whose source is CS-DocLink in this version. Ignore the **Default Values** section when creating CS-DocLink start menu items.

6. In the **Sites** field, select which sites can use this start menu item. Use **Ctrl+click** to select more than one site.
7. In the **Roles** field, select all the roles that can have access to the start menu item. Use **Ctrl+click** to select more than one role.
8. (Optional) To configure workflow process details for the assets that are created with this start menu item, complete the following steps:
   a. Click **Select Workflow**.
   b. In the “Select Workflow” form, select the appropriate workflow process from the drop-down list and then click **Select Workflow**.
   c. In the “Set Participants” form, select at least one user for each role that appears and then click **Set Participants**.
   d. Click **Continue**.

      Content Server saves the workflow information and redisplays the “Start Menu” form.
9. In the “Start Menu” form, click **Save**.
10. Repeat this procedure for each additional asset type you want to configure.

Users and CS-DocLink

Like the content providers who use the Content Server interfaces, the content providers who use the CS-DocLink application must have Content Server user accounts as well. Accounts for users who use CS-DocLink must have the following ACLs:

- Browser
- ElementReader
- PageReader
- RemoteClient
- UserReader
- xceleditor

For information about creating user accounts, see “Creating a New User,” on page 78.

Give the CS-DocLink users the following information:

- Their user name/password combination
- The appropriate URL for logging in to Content Server

They need both pieces of information to log in to Content Server through CS-DocLink.

The Content Server URL uses the following convention:

`http://<server>:<port>/<context>`

where:

- `<server>` is the host name or IP address of the machine running Content Server.
- `<port>` is the port on which Content Server is listening for connections.
- `<context>` refers to the application context root (URI) assigned to the Content Server application when it was installed.

Installing the CS-DocLink Client Application

In order to use the CS-DocLink feature, the CS-DocLink client must be installed on the client machine.

To install the CS-DocLink client

1. Point your browser to the following URL:

   `http://<server>:<port>/<context>/DownloadPage.html`

   where:

   - `<server>` is the host name or IP address of the machine running Content Server.
   - `<port>` is the port on which Content Server is listening for connections.
   - `<context>` refers to the application context root (URI) assigned to the Content Server application when it was installed.
2. On the “FatWire Content Server Applications” page, click **Download** in the “CS-DocLink” row.

> **Note**
>
> If one or more language packs are installed on your CS system, the page lists localized versions of the CS-DocLink client corresponding to the installed language packs. In such case, download the version appropriate to the user’s language preference.

3. Extract the downloaded archive to a temporary directory and open that directory.
4. Launch the CS-DocLink installer (`setup.exe`) and follow the instructions it displays.

Testing the CS-DocLink Configuration

To determine whether your CS-DocLink application is configured correctly, you must create and save an asset from the CS-DocLink interface and then examine it through Content Server.

To test the CS-DocLink configuration

1. Open Windows Explorer.
2. Double-click **CS-DocLink**.

3. In the **Login** dialog box, enter the following information:
   - The CS-DocLink URL. See "Users and CS-DocLink," on page 323 for the appropriate URL for your system.
   - The user name and password of a user who has the necessary ACLs (listed in "Users and CS-DocLink," on page 323).
4. Drag a document from the file system to the file structure represented under the CS-DocLink node and drop it in the appropriate place.
5. If there are required fields that must be filled in, the **Properties** dialog box appears. Enter the required information and save the asset.
6. Open your browser and log in to the Advanced interface as a user who has access to the type of asset that you just created.
7. If prompted, select the site in which you created the asset.
8. Find the asset that you created through CS-DocLink and open it in the "Inspect" form.
9. Examine the data listed for the fields and verify that the data that you specified in CS-DocLink matches the data displayed in the "Inspect" form.

InSite Interface

The InSite interface is a Content Server feature that enables infrequent users or users who perform a limited role to find, edit, and submit content directly from the rendered (Preview) version of an asset, which means that they do not have to learn how to use the Content Server interfaces.

Instead, they enter a Content Server URL that is appropriate for your system: for example, a URL that displays the home page of the rendered site. If the template for that page is coded with the `INSITE.EDIT` tags, the InSite interface appears.

The control panel provides a subset of the Content Server features and functions. Your InSite interface users interact with the CS system by using the control panel instead of the Content Server interfaces.

Enabling this feature requires four general steps:

- You, the administrator, set the `xcelerate.enableinsite` property in the `futuretense_xcel.ini` file to `true` on the CS management system.
- Your developers code templates that invoke the InSite feature for the fields that you want content providers to be able to edit in this way. Note that these same templates do not display the InSite interface on the CS delivery system because the `xcelerate.enableinsite` is set to `false` on that system.
- You determine the appropriate URL and then make it available to the CS users who need to use the InSite interface.

> **Note**
>
> The InSite interface cannot display assets that are created or edited with the CS-Desktop feature. That is, if an asset has a value in its `externaldoctype` column, the InSite interface cannot display it.

Enabling the InSite Interface

To enable the InSite Interface

1. Start the Property Editor and open the `futuretense_xcel.ini` file.
2. On the **xcelerate** tab, select `xcelerate.enableinsite` and set its value to `true`.
3. (Optional) If you want to use Ektron's eWebEditPro HTML editor within the InSite interface, verify that it is installed and confirm that the `xcelerate.ewebeditpro` property is set to `true`.

   For information about eWebEditPro, see the *Content Server Developer's Guide*.

   Note that to use eWebEditPro within the InSite interface, your developers must use the `EWEBEDITPRO` parameter with the `INSITE.EDIT` tags on your templates. For more information, see the *Content Server Developer's Guide*.
4. Save the property file and close the Property Editor.

After you enable the InSite interface, Content Server displays the InSite interface whenever a CS content provider uses a browser to navigate to an asset that is rendered by a template coded with the `INSITE.EDIT` tag.

For information about how to use the control panel, see the InSite interface help file.

User Accounts and the InSite Interface

Like the content providers who use the Content Server interfaces, the content providers who use the InSite interface must have Content Server user accounts as well. User accounts for the content providers who use the InSite interface must have the following ACLs:

- Browser
- ElementReader
- PageReader
- RemoteClient
- UserReader
- Visitor (if you are using Engage)
- xceleditor

For information about creating user accounts, see "Creating a New User," on page 78.

Be sure that you give the InSite interface users the following information:

- Their user name/password combination
- The appropriate URL for your system; that is, the URL that serves the appropriate site page (that is, `SiteCatalog` page entry) for a session in which someone uses the InSite interface

Maintaining Separate Browser Sessions for Preview

If the online site that you are serving from your CS delivery system requires visitors to log in, or it logs them in by default, your content providers need a separate browser session for preview if they must log in to the previewed site on the CS management system with a user name that is **different** from the one they are using in Content Server.

Without a new, separate browser session for preview, the Content Server session ends when the preview session begins, which means that content providers can lose any unsaved work.

To ensure that a preview session does not end the Content Server session, you can do the following:

- Have your site designers configure the online site so that your content providers are logged in to the site that they are previewing with their Content Server user accounts.
- Configure your CS management system so that the web server opens a new browser session for the preview window without closing the current session in the Content Server window.

To configure your CS management system so that the web server opens a new browser session for the previewed asset, you configure properties in the `futuretense_xcel.ini` file: `xcelerate.previewhost` and `xcelerate.previewservlet`.

You use these properties to point to the URL for the CS management system, but the URL must be different in some way from the URL that your content providers use to log in to the management system. For example:

- If content providers are not required to include the port number in the URL that they use to log in to CS, include it in the value for the `xcelerate.previewhost` property.
- If content providers are required to include the port number, specify a different port number for the property—the web server port number or the port number of a redirect request to the web server.
- Create an alias and use it only for the `xcelerate.previewhost` property.
- Use the `xcelerate.previewservlet` property to specify the Satellite Server servlet rather than the Content Server servlet.

Start the Property Editor, open the `futuretense_xcel.ini` file, select the **xcelerate** tab, and do one or both of the following:

- Set the `xcelerate.previewhost` property as follows:

  For most application servers, including Sun JES Application Server:

  `http://<server>:<port>/servlet/`
- Change the value of the `xcelerate.previewservlet` property from `ContentServer` to `Satellite`

Chapter 19

Configuring the Lucene Search Engine

To use the search feature in the Dash interface, you must enable and configure the built-in Lucene search engine. This chapter shows you how to set up, maintain, and disable Lucene on your system.

This chapter contains the following sections:

Overview

Lucene is a third-party search engine that ships with Content Server. Lucene provides the global search feature in Content Server’s Dash interface. Before you can search for assets in the Dash interface, you must set up Lucene.

Once you have set up Lucene, you can perform a range of maintenance tasks, such as making additional asset types searchable, or pausing indexing to perform bulk operations on assets.

If you need to, you can disable the indexing of assets of one or more types, globally disable the indexing of binary files, or disable Lucene completely, as necessary.

Setting Up Lucene

When you install Content Server, Lucene is disabled. The search feature in Content Server’s Dash interface will not function until you enable and configure Lucene on your system. The steps for setting up Lucene are as follows:

1. Enabling Lucene. Before you can configure Lucene, you must enable it on your system.
2. Adding Asset Types to the Indexing Queue. The queue indicates to Lucene which assets it should index.
3. Enabling Indexing of Binary Files. If one or more asset types which you added to the indexing queue are set up to reference binary files, you can configure Lucene to examine the contents of those files when indexing assets that reference them.

   Note

   You may choose not to enable this option if your assets do not reference any binary files, or if the files they reference contain content that is not indexable (text-based). For example, you will want to index Word or PDF documents, but not images or videos.

4. Building Asset Index Data. Once you have populated the indexing queue and enabled binary file indexing (if desired), you must start the indexing process.

   During indexing, Lucene examines the contents of assets of the selected types (and the binary files the assets reference, if applicable) and creates entries for those assets in the index. Once the assets are indexed, they will be returned by the search feature in Content Server’s Dash interface.

Enabling Lucene

This section shows you how to enable the Lucene search engine on your system.

To enable the Lucene search engine (search feature in the Dash interface)

1. In the **Admin** tab, expand **Search** and double-click **Start/Stop Search Engine Indices**.
2. Click **Start Search Engine**.
3. Click **OK** and continue to "Adding Asset Types to the Indexing Queue" below.

Adding Asset Types to the Indexing Queue

This section shows you how to add asset types to Lucene's indexing queue.

> **Note**
>
> Assets of the selected types will not be returned by the search feature in Content Server's Dash interface until Lucene has indexed them. To start the indexing process, you must complete the steps in "Building Asset Index Data," on page 334 after completing this procedure.

To add asset types to Lucene's indexing queue

1. In the **Admin** tab, expand **Search** and double-click **Configure Global Search**.
2. In the "For index:" drop-down list, select **Add**.

   Content Server displays a list of asset types that have not yet been added to the queue.

   > **Note**
   >
   > If no asset types are displayed when you select **Add** from the drop-down list, stop here. All asset types have already been enabled for indexing.

3. In the list, select the asset types you want to add to the queue.
4. Click **OK**.
5. Continue to one of the following sections:
   - If one or more asset types which you added to the indexing queue are set up to reference binary files, continue to "Enabling Indexing of Binary Files," on page 334 to configure Lucene to examine the contents of those files when indexing assets that reference them.
   - If you do not want to enable binary file indexing, continue to "Building Asset Index Data," on page 334 to build the index data for assets of the selected types.

Enabling Indexing of Binary Files

If one or more asset types which you added to the indexing queue are set up to reference binary files stored in the CS file system, you can configure Lucene to examine the contents of those files when indexing the assets that reference them. (By default, Lucene is set up to ignore all binary files referenced by assets being indexed.)

To enable binary file indexing (Lucene)

1. In the **Admin** tab, expand **Search** and double-click **Configure Global Search**.
2. Click **Start Binary Indexing**.
3. Continue to “Building Asset Index Data” below to build the index data for assets of the selected types.

Building Asset Index Data

This section shows you how to build or rebuild the index data for assets of types present in the indexing queue.

You will carry out the steps below for:

- Each asset type you have placed in the indexing queue but which has not yet been indexed. The search feature in Content Server’s Dash interface will not return assets of those types until Lucene has indexed them.
- Each asset type for which you want to rebuild the index from scratch. For example, you would rebuild the index after you have imported or modified a large number of assets of the same type.

Once the initial index data has been created, Lucene updates the index incrementally every 30 seconds. By default, index data is stored in the `<cs_shared_dir>/lucene` directory (where `<cs_shared_dir>` is the CS shared file system directory).

To build or rebuild asset index data (Lucene)

> **Note**
>
> Indexing is not an instantaneous process. When indexing, Lucene analyzes the contents of each asset (and any associated binary files, if you enabled binary file indexing), decides what is relevant, and stores the extracted information in the index. Therefore, the amount of time it takes to index assets depends on the number of assets being indexed and your system configuration.

1. If you have not already done so, add the desired asset types to the indexing queue. For instructions, see “Adding Asset Types to the Indexing Queue,” on page 333.
2. In the **Admin** tab, expand **Search** and double-click **Configure Global Search**.
3. In the “For index:” drop-down list, select **Re-index**.

Content Server displays the contents of the indexing queue.

> **Note**
>
> If no asset types are displayed when you select **Re-index** from the drop-down list, stop here. No asset types have yet been added to the indexing queue or indexing has been paused for all asset types in the queue.

4. In the list, select the asset types whose index data you want to build (or rebuild).
5. Click **OK**.
6. In the confirmation pop-up dialog that appears, click **OK**.
7. Wait until the indexing process is complete. The more assets are queued for indexing, the longer it will take to index them.

> **Note**
>
> The search feature in Content Server's Dash interface will not return assets of the affected types until the indexing process is complete.

Maintaining Lucene

Once you have set up Lucene, you will likely need to perform tasks such as making new asset types searchable in the Dash interface, or temporarily suspending indexing in order to perform bulk operations on assets. These maintenance tasks are described in the following sections:

- Pausing Asset Indexing
- Other Maintenance Tasks

Pausing Asset Indexing

In certain situations, you may want to temporarily suspend the indexing of assets. For example, you would pause indexing for an asset type when performing a bulk import of assets of that type into the CS database.

When you pause indexing for an asset type, Lucene does the following:

- Removes the asset type from the indexing queue
- Preserves the index data for the affected assets

When indexing is paused, the search feature in Content Server's Dash interface continues to return the affected assets. However, when changes are made to the affected assets, the index data is not updated.

To pause asset indexing (Lucene)

1. In the **Admin** tab, expand **Search** and double-click **Configure Global Search**.
2. In the "For index:" drop-down list, select **Pause**.

Content Server displays the list of asset types for which you can pause indexing.

> **Note**
>
> If no asset types are displayed when you select **Pause** from the drop-down list, stop here. Either indexing has already been paused for all asset types in the indexing queue, or no asset types have yet been added to the queue.

3. In the list, select the asset types for which you want to pause indexing.
4. Click **OK**.
5. In the pop-up warning dialog that appears, click **OK**.

   Content Server removes assets of the selected types from the indexing queue and preserves their index data. The search feature in Content Server's Dash interface will continue to return the affected assets, but the results will not account for any changes made to the assets after indexing has been suspended.
6. Resume indexing of the affected assets by doing the following:
   a. Add the affected asset types back to the indexing queue. For instructions, see "Adding Asset Types to the Indexing Queue," on page 333.
   b. Restart indexing for the affected asset types. For instructions, see "Building Asset Index Data," on page 334

Other Maintenance Tasks

In addition to pausing asset indexing, the following maintenance tasks are available once you set up Lucene:

- Adding Asset Types to the Indexing Queue. Add more asset types to the indexing queue (and then index the assets) so that Dash interface users can search for them.
- Building Asset Index Data. Index assets of types that have not yet been indexed (such as asset types you have added to the indexing queue after the initial setup), or rebuild index data for an asset type that's already been indexed (for example, after a bulk import).
- Disabling Asset Indexing. Exclude assets from being indexed (and thus returned by the search feature in Content Server's Dash interface).
- Disabling Indexing of Binary Files. To increase performance, you may choose not to index binary files, especially if the majority of the binary files associated with assets on your site does not contain any indexable data (for example, images, videos, and sound clips do not contain any text that Lucene can analyze).

Disabling Lucene Functions

This section shows you how to perform the following tasks:

- Disabling Asset Indexing
- Disabling Indexing of Binary Files
- Disabling Lucene

Disabling Asset Indexing

This section shows you how to exclude assets of one or more asset types from being indexed (and thus, being returned by the search feature in Content Server's Dash interface).

When you disable indexing, Lucene does the following:

- Removes the affected assets from the indexing queue
- Deletes the index data for the affected assets

After you perform the steps below, the search feature in Content Server's Dash interface will no longer return assets of the selected types.

To disable asset indexing (Lucene)

1. In the **Admin** tab, expand **Search** and double-click **Configure Global Search**.
2. In the "For index:" drop-down list, select **Delete**.

   Content Server displays the contents of the indexing queue.

   > **Note**
   >
   > If no asset types are displayed when you select **Disable** from the drop-down list, stop here. No asset types have yet been added to the indexing queue.

3. In the list, select the asset types whose index data you want to delete.
4. Click **OK**.
5. In the pop-up confirmation dialog that appears, click **OK**.

   Content Server removes assets of the selected types from the indexing queue and deletes their index data. The assets will no longer be returned by the search feature in Content Server's Dash interface.

6. To make the assets searchable again, you must:

   a. Add the affected asset types back to the indexing queue. For instructions, see "Adding Asset Types to the Indexing Queue," on page 333.

   b. Rebuild index data for the assets. For instructions, see "Building Asset Index Data," on page 334.

Disabling Indexing of Binary Files

If you decide that you no longer want Lucene to examine the contents of binary files referenced by assets it indexes, you can disable the feature to improve performance.

To disable binary file indexing (Lucene)

1. In the **Admin** tab, expand **Search** and double-click **Configure Global Search**.
2. Click **End Binary Indexing**.

   Lucene will now ignore all binary files referenced by the assets it indexes.

Disabling Lucene

This section shows you how to completely disable Lucene on your system.

> **Note**
>
> If you disable Lucene, the search feature in Content Server's Dash interface will not function.

To disable Lucene

1. In the **Admin** tab, expand **Search** and double-click **Start/Stop Search Engine Indices**.
2. Click **Stop Search Engine**.

   Lucene is now disabled.

Chapter 20

Revision Tracking

Content Server provides revision tracking functionality that prevents a row in a table from being edited by more than one user at a time. When you enable revision tracking for a table, Content Server maintains multiple versions of a row in that table.

Content Server applies this revision tracking functionality to your asset types. You decide which asset types should be tracked and determine how many revisions to store. The content providers then check their assets in and out and can compare versions, if necessary.

The *Content Server Advanced Interface User's Guide* describes how content providers work with assets when revision tracking is in use. This chapter describes how to enable revision tracking and how to manage versions of assets.

This chapter contains the following sections:

Overview

Content Server provides revision tracking functionality through its revision tracking API. This API is used to provide additional revision tracking functionality for asset type tables.

When you enable revision tracking for an asset type, Content Server creates a new table, called a **tracker** table, for assets of that type. You specify how many versions you want to keep and you also specify a storage directory that the tracker table uses to store supporting files for the assets that it is tracking.

Content Server's implementation of revision tracking provides the following features:

- **Check out** and **check in**.

  Check out locks an asset so that only one user can edit it at a time.

  Check in releases the lock on the asset, increments the version number, and determines whether the number of versions falls within the configured limit. If the new version exceeds the limit, the oldest version is deleted to make room for the next version.

- Storage of **multiple versions** of an asset.

  When you enable revision tracking for an asset type, Content Server stores versions in a **tracker** table; upload data is stored in a storage directory that you specify, as shown in "The RTInfo Table," on page 341. Because past versions are stored (that is, a history exists), a user can **roll back** an asset to a previous version or examine the **differences** between two versions of the asset.

- Administrative or maintenance features.

  An administrator can **delete past versions** of an asset or **clear the checkout** for an asset by overriding the check out on it and checking it back in.

When an asset type is being tracked, Content Server provides checkin, checkout, and other revision tracking features on the **New** and **Edit** forms for assets of those types.

Tracker Tables and Storage Directories

When you enable revision tracking for an asset type, Content Server creates a tracker table that stores revision information for the records in the source table. A tracker table has the same name as the main storage table for the asset type with `_t` appended to it. For example, the tracker table for the article asset type would be named `Article_t`. The tracker table for the attribute type asset type would be named `AttrTypes_t`.

For each record in a tracked asset type table, there are several rows in the corresponding tracker table that stores its version information. Tracker tables have two kinds of columns:

- Columns that store the system information that the revision tracking system needs to keep track of all the versions
- Columns that hold the IDs of text files that are stored in a storage directory

When a new version of an asset is checked in, Content Server creates a separate text file to hold the data in each of the asset type's upload (URL) fields and in any text fields that are configured to hold more than 64 characters. These files are stored in the storage directory that you specify when you enable revision tracking. There is a set of these text files stored in the storage directory for each version of the asset.

Note

Because tracker tables hold system information only, they are hidden in Content Server Explorer. Do **not** attempt to modify the information in any of the tracker tables with a database tool.

The RTInfo Table

While the tracker tables are kept hidden, the `RTInfo` table is visible through Content Server Explorer. This table holds information about which tables are being revision tracked. It has the following columns:

Column	Description
`tblname`	The name of a table that is being revision tracked. For asset types, this is the name of the main asset type table.
`versions`	The number of versions to store for each asset of this type.
`storage`	The path to the storage directory that holds the text files for each version of assets of this type. If, when you enable revision tracking, you do not specify a storage directory, Content Server creates a revision tracking subdirectory in the `defdir` directory for the source table.
`recordupdate`	A timestamp of the last time a version was stored in the tracker table.
`trackingupdate`	The time at which revision tracking was enabled for the source table.

Revision Tracking and the Two Asset Models

Because the data model for basic assets is different from the data model for flex assets, the revision tracking system works differently for the two asset models:

- For basic assets, only the row in the main asset storage table is tracked.
- For flex assets and the other multi-table asset types (template and CSElement), the information from the appropriate rows from all of their tables are serialized into an object and stored in the tracker table for that asset type.

For example, if you enable revision tracking for template assets, the appropriate rows from the `Template`, `SiteCatalog`, and `ElementCatalog` tables are serialized into an object and stored in the `Template_t` table.

Implicit vs. Explicit Checkin and Checkout

When revision tracking is on for an asset type, Content Server provides both implicit (or automatic) and explicit (or manual) checkout/checkin functionality. When users create or edit assets of a type that is being revision tracked, they do not have to manually check out the asset: it is automatically assigned to them. Then, when they click **Save**, the asset is checked back in. This is known as implicit checkin/checkout.

This may or may not be the behavior that you want. For example, if an author is making extensive revisions to an asset that was checked out implicitly, each save creates an archived version. Depending on how many revisions the asset type is configured to store, that author might overwrite an older version that he or she really wanted to keep.

When a content provider manually (explicitly) checks out an asset, a version is not stored—no matter how many times he or she saves it—until it is manually checked back in.

Revision Tracking and Non-Asset Tables

In addition to using revision tracking for your asset types, you can implement revision tracking on your non-asset tables.

To do so, you use the **Content Server Management Tools** feature in the **Admin** tab to enable tracking for the table. Content Server then creates a corresponding tracker table to support the tracked table. It is named the same way as a tracker table for an asset type: `nameOfTable_t`. For example, if you enabled revision tracking for the Source table, the tracker table would be called `Source_t`.

When revision tracking is enabled for a non-asset table, you can use either the revision tracking features accessible from the menus in Content Server Explorer or the "Content Server Management Tools" node on the "Admin" tab to lock (check out) a row and then unlock the row (check it back in) when you are finished with it.

If you need to provide additional support outside of the Content Server Explorer tool for revision tracking of a non-asset table, your developers can code additional forms, using the Content Server revision tracking API and revision tracking XML or JSP tags. For information, see the *Content Server Developer's Tag Reference* and the *Content Server Java API Reference*.

Note

If you need to delete a non-asset table from the Content Server database and that table is being revision tracked, be sure to untrack the table before deleting it.

How Many Versions?

Each revision of an asset or a database row occupies disk space. Therefore, your decision about how many revisions to keep must be based on the following factors:

- The amount of disk space that you have available
- The typical data size of the asset (or row)
- The likelihood that there could be a need for a rollback of several versions

For example, an asset that consists of a small amount of ASCII data occupies so little space that a large number of revisions would take little disk space. However, each version of an asset that holds a large amount of binary data could occupy a significant amount of disk space. In the second case, you must strike the appropriate balance, storing the fewest number of versions necessary for rollback purposes.

Enabling Revision Tracking

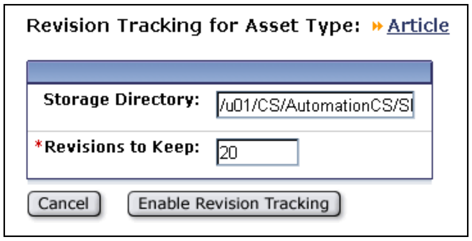
This section shows you how to enable revision tracking for assets and for non-asset tables.

Enabling Revision Tracking for Assets

To enable revision tracking for assets

1. In the **Admin** tab, expand **Asset Types,** then the asset type for which you want to enable revision tracking.
2. Under the selected asset type, expand **Revision Tracking** and double-click **Track**.

   The “Track Asset Type” form appears:



Revision Tracking for Asset Type: Article

Storage Directory:	/u01/CS/AutomationCS/SI
*Revisions to Keep:	20

3. In the **Storage Directory** field, enter the full path to the directory in which revisions of assets of the selected type will be stored. Do not add a slash (or backslash) character after the directory name.
4. In the **Revisions to Keep** field, enter the number of revisions you want Content Server to store for each asset of the selected asset type. Once this many revisions are stored, the oldest revision is overwritten by the next revision.

   If you are using revision tracking solely for its record-locking feature and you do not need the ability to roll back to previous versions, you can set this field to 1.
5. Click **Enable Revision Tracking**.

Enabling Revision Tracking for Non-Asset Tables

To enable revision tracking for database tables that do not hold assets, use the **Content Server Management Tools** node.

To enable revision tracking for non-asset tables

1. In the **Admin** tab, expand **Content Server Management Tools** and double-click **Revision Tracking**.
2. In the form that appears, select **Track Tables** and click **OK**.
3. In the **Enter root storage directory** field, enter the full path to the directory in which revisions of table rows will be stored. Do not add a slash (or backslash) character after the directory name.
4. In the **Enter number of revisions to keep** field, enter the number of revisions Content Server should store for rollback purposes.
5. Select the **Track?** check box next to each table for which you are enabling revision tracking.
6. Click **Track Tables**.

Editing Revision Tracking Settings

You must have the SiteGod ACL to either inspect or edit revision tracking settings. If you attempt to examine the current revision tracking settings for a tracked asset type or non-asset table and you do not have the SiteGod ACL, the system does not display the name of the root storage directory or the number of revisions.

Because revision tracking settings directory affect the database, you must be careful when changing these settings.

Changing the Root Storage Directory

> **Caution**
>
> If you change the root storage directory for an asset type or non-asset table, you will lose all the versions currently stored for it.

If you must change the root storage directory, follow these steps:

1. Disable revision tracking for the asset type or table. When you do this, all the version data for the asset type is orphaned.
2. Enable revision tracking for the asset type or table, entering the new root storage directory.

Changing the Number of Revisions

Increasing

If you want to increase the number of revisions to be stored for an asset type or table, simply increase the value in the **Revisions to Keep** field:

- **For asset types:** in the **Admin** tab, expand **Asset Types**, the asset type you want to modify, **Revision Tracking,** and double-click **Set Revisions**. Use the form that appears to increase the value.
- **For non-asset tables:** in the **Admin** tab, expand **Content Server Management Tools**, **Revision Tracking**, and double-click **Set Table Revisions**. Use the form that appears to increase the value.

Decreasing

Although you might need to decrease the number of versions while you are testing configuration settings on a CS development system or while you are fine-tuning the CS management system, it is best if you do not decrease the number of versions being stored by Revision Tracking on a fully functioning management system.

If you decrease the value in the **Revisions to Keep** field to a number that is less than the number of revisions currently being stored for that asset type or table, the following occurs:

- The text files in the storage directory for the extra versions are orphaned.
- The rows in the tracker table for the extra versions are orphaned.

You can avoid creating orphan rows in the tracker table by deleting the extra versions (the oldest ones) before you decrease the number of revisions. However, you cannot avoid creating orphan text files in the storage directory and you should not attempt to delete them because it is very difficult to determine which ones to delete.

If you must decrease the number of revisions, complete the following steps (in this order):

1. Follow the steps in the procedure "Deleting Revisions."
2. To decrease the number of revisions, do one of the following:

- **For asset types:** in the **Admin** tab, expand **Asset Types**, the asset type you want to modify, **Revision Tracking,** and double-click **Set Revisions**. Use the form that appears to decrease the value.
- **For non-asset tables:** in the **Admin** tab, expand **Content Server Management Tools**, **Revision Tracking**, and double-click **Set Table Revisions**. Use the form that appears to decrease the value.

Deleting Revisions

Under certain conditions, you might need to delete old versions for an asset type or table that is being revision tracked.

To delete revisions

1. In the **Admin** tab, expand **Content Server Administrator Tools** and double-click **Revision Tracking**.
2. Enter the name of the table you want to work with. If you do not know the name of the table, do one of the following:
   - Leave the field blank. Content Server will return a list of all tables in the database.
   - Enter a partial name, ending with the wildcard character (**%**). Content Server will return a list of tables named similarly to your criteria.
3. In the **Enter Value for Key** field, enter the ID of the asset or table row whose versions you want to delete. (You can obtain the ID of an asset by inspecting it in the Content Server interfaces. You can obtain the ID of a row by examining the table with Content Server Explorer.)
4. Select **Delete Revisions** and click **OK**.
5. In the "Delete Revisions" form, select the option next to the versions that you want to delete. You can use the **All** button to select all of the boxes and the **None** button to clear all of the boxes.
6. Click **Delete Revisions**.

Disabling Revision Tracking

When you disable revision tracking for an asset type or a non-asset table, the following occurs:

- The tracker table is inactivated, but not deleted. This means that the versions stored in it are orphaned, but not deleted. And, after revision tracking is disabled for a table, you can no longer delete versions by using the Content Server revision tracking forms.
- All links to the text files in the storage directory are broken, but the files themselves are not deleted. They, too, are orphaned.

If you later decide to enable revision tracking for that asset type or non-asset table, the old versions and text files are ignored. They remain orphaned.

Because these orphaned versions and text files take up disk space and there could be a large number of them stored on disk, it is best practice to complete the following tasks in the following order when you want to disable revision tracking:

1. Use the Content Server Management Tools revision tracking forms to delete all the versions stored for the table. See the procedure “Deleting Revisions,” on page 345.
2. Disable revision tracking for the table. See the procedures in this section.
3. Manually delete all the text files from the revision tracking directory located inside the storage directory for that asset type or database table.

   For example, if the storage directory for the asset type is `/Storage/AssetType`, then delete all the files from the `/Storage/AssetType/AssetType` revision tracking directory.

   Note

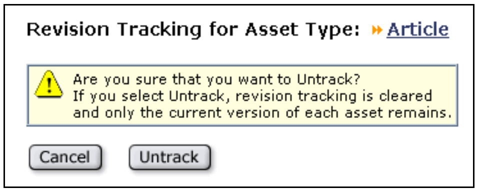
   Do **not** delete any files from the asset type storage directory.

Disabling Revision Tracking for Asset Types

To disable revision tracking for asset types

1. In the **Admin** tab, expand **Asset Types**, then the asset type for which you want to disable revision tracking.
2. Under the selected asset type, expand **Revision Tracking** and double-click **Untrack**.

   Content Server displays a warning message:



Revision Tracking for Asset Type: Article

Are you sure that you want to Untrack?
If you select Untrack, revision tracking is cleared and only the current version of each asset remains.

3. Click **Untrack**.

Disabling Revision Tracking for Non-Asset Tables

To disable revision tracking for non-asset tables

1. In the **Admin** tab, expand **Content Server Management Tools** and double-click **Revision Tracking**.
2. In the "Revision Tracking" form, select **Untrack Tables** and click **OK**.
3. In the "Untrack Tables" form, select the **Untrack?** check box next to each table for which you want to disable revision tracking.
4. Click **Untrack Tables**.

Unlocking Revisions

Occasionally, user operations may leave an asset in an inappropriate locked state. Resolving these states is an administrative responsibility.

- For asset types, use the **Clear Checkouts** function in the **Admin** tab.
- For non-asset tables, use the **Revision Tracking** function under **Content Server Management Tools**.

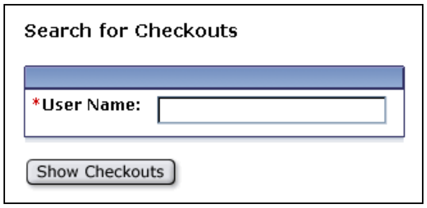
Clearing Checkouts for Assets

Before you begin, find out the name of the user who has the asset locked. This information is listed on the "Inspect" form for the asset.

To clear checkouts for assets

1. In the **Admin** tab, double-click **Clear Checkouts**.

   Content Server displays the "Search for Checkouts" form:



Search for Checkouts

*User Name:

2. Enter the name of the user who has left the asset locked and click **Show Checkouts**.
3. In the list of assets checked out to the selected user, select the **Clear** check box next to each asset for which you want to undo the checkout.
4. Click **Clear Checkouts**.

Unlocking Versions for Non-Asset Tables

To unlock rows for a non-asset table that you are revision tracking, you need to determine the object ID of the row that you want to unlock. You can use Content Server Explorer to examine the table and determine the object row.

Once you have obtained the table's object ID, complete the following steps:

1. In the **Admin** tab, expand **Content Server Management Tools** and double-click **Revision Tracking**.
2. In the form that appears, enter the table's object ID, select **Unlock Rows**, and click **OK**.
3. In the "Unlock Rows" form, select the **Unlock** check box next to each row you want to unlock. Use the **All** button to select all rows in the list; use the **None** button to clear your selection.
4. Click **Unlock Records**.

Additional Revision Tracking Functions for Non-Asset Tables

The Advanced interface allows you to change the status of revisions in every way that a user normally does. This allows you to correct inappropriate states by administrative intervention when some error occurs.

The options all appear as radio buttons on the **Revision Tracking** screen:

- Lock
- Commit
- Release
- Rollback
- History
- Track Tables (see "Enabling Revision Tracking," on page 343)
- Untrack Tables (see "Disabling Revision Tracking," on page 346, above)
- Set Table Revisions (see "Editing Revision Tracking Settings," on page 344)
- Delete Revisions (see "Deleting Revisions," on page 345
- Unlock Rows (see "Unlocking Revisions," on page 347)

A summary of each of the other operations appears below.

Lock

The "Lock" form presents a list of rows that is restricted by any criteria that you specified on the "Revision Tracking" form:

- A red lock icon appears next to the rows that are locked by another user.
- A blue lock icon appears next to rows locked by you.

You can lock any asset by clicking the check box in the **Lock?** column and clicking the **Lock** button. The attempt fails if you do not have permission to lock the checked item. If it succeeds, the item is now checked out to you.

Commit

The "Commit" form presents a comment box and a list of rows checked out to you.

You can enter a comment and check the box in the **Commit** column for each item that you want to update. The check box in the **Keep Locked?** column specifies whether you want the item to be unlocked after it is committed (checked in).

The Commit action updates the revision to the version you have been editing.

Release

The “Release” form presents a list of rows that are checked out to you.

You can check the box in the **Release?** column for each item that you want to release. A release differs from a commit in that any changes that you have made to the asset since locking it are lost.

This form differs from the “Unlock Rows” form in that it only lists assets checked out to you.

Rollback

The “Rollback” form presents a list of rows and their versions. The list is restricted by any criteria that you entered on the “Revision Tracking” form.

Note

To roll back an asset to a previous version, use the buttons on the asset’s “Inspect” or “Status” form.

You can select any version for the rollback operation by clicking the radio button in the **Rollback?** column. A rollback creates a new version. It matches the version that you rolled back to but adds the comment “Version created by Rollback.”

History

The “History” form is similar to the “Rollback” form, except that it is informational only. For each row that it shows, it provides the date and revision information.

Appendices

This part contains information about Content Server's system defaults and sample sites. It provides LDAP configuration procedures and describes the system behavior that results when user and site management operations are carried out when the LDAP options are installed.

This part contains the following appendices:

Appendix A
System Defaults

This appendix lists and describes the Content Server system defaults that you will be using routinely.

This appendix contains the following sections:

ACLs

Content Server and its applications use several default ACLs to control user access to their features and functions. This section summarizes the permissions that can be specified in an ACL, and describes Content Server's default system ACLs.

Permissions

An ACL specifies a set of permissions. When an ACL is assigned to a database table, only the permissions specified in the ACL can be exercised on the database table. Only a user with the same ACL as the table can exercise those permissions.

Table A-1 lists all the permissions that can be specified in an ACL.

Table A-1: Permissions Supported by Content Server

Permission	Bit Mask[1]	Action
Read	1	Read data from a table.
Retrieve	16	Retrieve the contents of a URL column, also known as an upload field. For information about URL columns, see the *Content Server Developer's Guide*
Write	2	Write information to a table.
Create	4	Create a table.
Delete	8	Delete information from a table.
Revision Tracking Audit	32	Access all the revision tracking information for the rows (records) in a tracked table.
Revision Tracking Admin	64	Assign or remove revision tracking on a table.

[1] When an ACL is created, the bit mask numbers for each permission assigned to an ACL are added together and the totals are listed with the ACL in the `SystemACL` table.

Accessing ACLs

Content Server's ACLs and their permissions are accessible as either a listing or an individual entry.

- To obtain the list of ACLs and their permissions, open the `SystemACL` table directly.
- To obtain an individual ACL and its permissions, use the Advanced interface:

1. In the **Admin** tab, expand **Content Server Management Tools** and double-click **ACLs**.
2. In the drop-down list, select the ACL you want to work with.

3. Select **Modify ACL** and click **OK**.

Caution

Never modify a default system ACL. And never modify the ACLs assigned to any of the system tables.

For descriptions of the system ACLs, see “System ACLs,” on page 356.

System ACLs

Table A-2 lists the system ACLs and their permissions. Each system ACL exists in order to control access to specific parts of the database tables, and subsequently, the product features that use those tables. Although several of the default ACLs have the same set of permissions, the ACLs are all necessary because they are assigned to different tables.

Table A-3 describes the functions of each ACL and how each ACL is used by Content Server and the CS content applications.

Table A-2: System ACLs and Their Permissions

ACL Name	Permissions						
	Read	**Retrieve**	**Write**	**Create**	**Delete**	**Rev. Track Audit**	**Rev. Track Admin**
Browser	x						
ContentEditor	x	x	x	x	x	x	
ElementEditor	x	x	x	x	x	x	
ElementReader	x						
PageEditor	x	x	x	x	x	x	
PageReader	x						
RemoteClient	x	x	x	x	x	x	x
SiteGod	x	x	x	x	x	x	x
TableEditor	x	x	x	x	x	x	
UserEditor	x	x	x	x	x	x	
UserReader	x						
Visitor	x	x	x	x	x	x	
VisitorAdmin	x	x	x	x	x	x	x
WSAdmin							
WSEditor							
WSUser							
xceladmin	x	x	x	x	x	x	x
xceleditor	x	x	x	x	x	x	
xcelpublish	x	x	x	x	x	x	

Table A-3: System ACLs and Their Descriptions

ACL Name	Description
Browser	Allows read-only access to the content in the Content Server database. It is assigned to most of the system default and sample site users. Content Server requires that all visitors to an online site that it manages have user accounts. For this reason, Content Server is delivered with a default user account, named DefaultReader, that it assigns to all non-authenticated visitors, that is, those who do not have a user account of their own. The Browser ACL is assigned to the DefaultReader user account, which gives non-authenticated visitors read-only access rights to the content in the Content Server database.
ContentEditor	Used for the sample Content Server portal site. This ACL is assigned to the tables that support the sample portal. Anyone who works with the portal sample needs this ACL.
ElementEditor	Allows users to write data to the `ElementCatalog` and `SystemSQL` tables. Site designers and anyone who creates templates, CSElement, and SiteEntry assets need this ACL.
ElementReader	Allows users to read data in the `ElementCatalog` and `SystemSQL` tables. Content Server users need this ACL so they can inspect the templates assigned to their assets.
PageEditor	Allows users to create page entries in the `SiteCatalog` table. Site designers and anyone who creates a template, CSElement, or SiteEntry asset need this ACL.
PageReader	Allows users to read page entries from the `SiteCatalog` table. Content Server users need this ACL so they can inspect the templates assigned to their assets.
RemoteClient	Grants users the ability to log in to the Content Server management system through a remote client like CS-Desktop. All CS-Desktop users need this ACL.
SiteGod	Enables complete access to all the tables in the Content Server database. At least one user of the management system, typically an administrator, must have the SiteGod ACL.
TableEditor	Allows users to create and delete tables in the Content Server database. Site designers who create database tables or who create new asset types (which causes new tables to be created) need this ACL. Administrators or anyone else who will use the Initialize Mirror Destination feature also needs this ACL.
UserEditor	Allows users to manage user accounts. Administrators need this ACL.

Table A-3: System ACLs and Their Descriptions *(continued)*

ACL Name	Description
UserReader	Allows users to read information about user accounts, but not to change that information. Content Server users need this ACL so they can use workflow.
Visitor	Grants the holder of the ACL the ability to write data to the Content Server Engage tables that hold visitor data, and to create recommendation assets. Any visitor whose data you are collecting on the delivery system must have this ACL assigned to their user account. Any Engage user who needs to create Recommendation assets needs this ACL.
VisitorAdmin	Grants users the ability to create visitor attributes, history attributes, and history types. Any Engage user who needs to create assets of those types needs this ACL.
WSUser	Assigned to `SiteCatalog` page entries for the Web Services feature. Grants users the ability to access Content Server through the Content Server web services.
WSEditor	Assigned to `SiteCatalog` page entries for the Web Services feature. Grants users the ability to access Content Server through the Content Server web services.
WSAdmin	Assigned to `SiteCatalog` page entries for the Web Services feature. Grants users the ability to access Content Server through the Content Server web services.
xceladmin	Grants users the ability to create user profiles, roles, sites, asset types, and so on—that is, to use all the functions in the **Admin**, **Site Admin**, and **Workflow** tabs. System, site, and workflow administrators need this ACL. Also, because the **Admin** tab has both administrative and site design functions, site designers also need this ACL.
xceleditor	Grants users the ability to log in to the CS content applications. The login request code verifies whether or not a user has the ACL. All users of the management system need this ACL.
xcelpublish	Grants users the ability to view the Publish Console.

ACLs of Default Users

Table A-4: Default Users and Their ACLs

User Name	ACLs	Description
fwadmin	Browser ElementEditor PageEditor PageReader RemoteClient TableEditor UserEditor UserReader Visitor VisitorAdmin xceladmin xceleditor xcelpublish wsadmin wseditor wsuser	The basic administrator user that Content Server creates so that you can begin configuring your CS content applications. Do **not** delete this user unless another user with an identical ACL assignment already exists.
Content Server (the installation's user account)	Browser ContentEditor ElementEditor ElementReader PageEditor PageReader SiteGod TableEditor UserEditor UserReader	The user account that the installation program creates during the installation of the products. The name of this account is whatever the installers chose for it.
DefaultReader	Browser Visitor	The default user name that Content Server assigns to non-authenticated site visitors on the delivery system.

Required ACLs for Custom Users

Table A-5: System ACLs Required by Users

User	Required ACLs
All users	Browser, Element Reader, PageReader, UserReader, xceleditor
Workflow administrator Site administrator	xceladmin
Administrator	TableEditor, UserEditor, VisitorAdmin (if you have Engage installed), xceladmin
Site designer	ElementEditor, PageEditor, TableEditor, Visitor (if you have Engage installed), Visitor Admin (if you have Engage installed), xceladmin
Engage users	Visitor
CS-Desktop, CS-DocLink, and InSite interface users	RemoteClient, Visitor (if your installation uses Engage)

System Roles

Table A-6: System Roles

Role	Description
GeneralAdmin	A default system role. Required for users who need access to the **Admin** tab in the tree. Note that in addition to having the GeneralAdmin role for a site, a user must also have the xceladmin ACL in order to use any of the functions in the **Admin** tab.
SiteAdmin	A default system role. Required for users who need access to the **Site Admin** tab, which holds a subset of the features and functions that are in the **Admin** tab. Use the SiteAdmin role if you want certain users to manage, but not create, the users who can access a site. Note that in addition to having the SiteAdmin role for the site, a user must also have the xceladmin ACL in order to use any of the functions on the **Site Admin** tab.
WorkflowAdmin	A default system role. Required for users who need access to the **Workflow** tab in the tree. Note that in addition to having the WorkflowAdmin role for the site, a user must also have the xceladmin ACL in order to use any of the functions on the **Workflow** tab.

System Asset Types

Table A-7 lists the default asset types. Unlike custom asset types, system asset types cannot be deleted.

Table A-7: System Asset Types

Asset Type	Description
Attribute Editor	An attribute editor specifies how data is entered for a flex attribute when that attribute is displayed on a "New" or "Edit" form for a flex asset or a flex parent asset. It is similar to a template asset. However, unlike a template asset, you use it to identify the code that you want Content Server to use when it displays an attribute in the CS interface—not when it displays the value of an attribute on your online site.
CSElement	Stores code (XML or JSP and Java) does not render assets. Typically, you use CSElements for common code that you want to call from more than one template (a banner perhaps). You also use CSElements to provide the queries that are needed to create DynamicList recommendations in Engage.
Collection	Stores an ordered list of assets of one type. You "build" collections by running one or more queries, selecting items from their resultsets, and then ranking (ordering) the items that you selected. This ranked, ordered list is the collection. For example, you could rank a collection of articles about politics so that the article about last night's election results is number one.
Dimension	Represents a locale in a site. You must create a "Dimension" asset for each locale you want to enable on the management system. To enable publishing of content in a given locale, you must publish the corresponding "Dimension" asset to the delivery system, and enable the locale in the site's dimension set.
Dimension Set	Defines which locales and locale filter are enabled on the online site. For locale filtering to work on the delivery site, you must create and publish to the delivery system at least one "DimensionSet" asset. Has no effect on the management system.
History Attribute	Individual information types that you group together to create a vector of information that Engage treats as a single record. This vector of data is the history definition. For example, a history type called "purchases" can consist of the history attributes "SKU," "itemname," "quantity," and "price." Available in Engage.
History Definition	The vector of data in a History Attribute. This vector of data is the history definition. For example, a history type called "purchases" can consist of the history attributes "SKU," "itemname," "quantity," and "price." Available in Engage.
Link	Stores a URL to an external web site. You use this asset to embed an external link within another asset.

Table A-7: System Asset Types *(continued)*

Asset Type	Description
Page	Stores references to other assets. Arranging and designing page assets is how you represent the organization or design of your site. You design page assets by selecting the appropriate collections, articles, imagefiles, queries, and so on for them. Then, you position your page assets on the Site Plan tab that represents your site in the tree on the left side of the Content Server interfaces.
Promotion	Is a merchandising asset that offers some type of value or discount to your site visitors based on the flex assets (products, perhaps) that the visitor is buying and the segments that the visitor qualifies for. Available in Engage.
Query	Stores queries that retrieve a list of assets based on selected parameters or criteria. You use query assets in page assets, collections, and recommendations. The database query can be either written directly in the “New” or “Edit” form for the query asset as a SQL query, or written in an element (with Content Server query tags or a as a search engine query) that is identified in the “New” or “Edit” form.
Recommendation	This is like an advanced collection. It collects, assesses, and sorts flex assets (products or articles, perhaps) and then recommends the most appropriate ones for the current visitor, based on the segments that visitor belongs to. Available in Engage.
Segment	Assets that divide visitors into groups based on common characteristics (visitor attributes and history types). You build segments by determining which visitor data assets to base them on and then setting qualifying values for those criteria. For example, a segment could define people who live in Alaska and own fly fishing gear, or it could define people who bought a personal computer in the past six months, and so on. Available in Engage.
SiteEntry	Represents a Content Server page or pagelet and has a CSElement assigned as the root element that generates the page. Template assets do not have associated SiteEntry assets because they represent both an element and a Content Server page.
Template	Stores code (XML or JSP and Java) that renders other assets into Content Server pages and pagelets. Developers code a standard set of templates for each asset type (other than CSElement and SiteEntry) so that all assets of the same type are formatted in the same way. Content providers can select templates for previewing their content assets without having access to the code itself or being required to code.
Visitor Attribute	Holds types of information that specify one characteristic only (scalar values). For example, you can create visitor attributes named “years of experience,” “job title,” or “number of children.” Available in Engage.

Default Tree Tabs

Table A-8 lists the default tabs in Content Server's tree. These tabs are critical to Content Server. All features which stem from Content Server can be accessed through these tabs; they are automatically created upon installation.

Table A-8: Default Tabs in Content Server

Tab	Description
Active List	The Active List is a tab which displays items that are in the process of being created or edited in Content Server.
Admin	Displays the administrative functions that affect all of the CM sites in the system. By default, only users with the default system role named GeneralAdmin have access to this tab.
Design	The design tab is a source for creating pages on your site. Some of these sources are: Templates, Product Definition, Content Definition, and other sources for the creating pages.
History	Displays the assets that you worked with during the current session. All users see this tab as soon as they create, inspect, edit, or copy their first asset.
Query	This tab is the means for which a user can query for certain types of articles and organize them in that fashion. This is accomplished using a SQL query.
Site Admin	Holds a subset of the administrative functions that apply only to the CM site that you are currently logged in to. By default, only users with the default system role named SiteAdmin have access to this tab. This tab is useful if you have individuals who manage which existing users have access to individual CM sites, but who do not need to create new users or new sites.
Site Plan	A layout and overview of the site. This tab will show each site controlled by Content Server. It lists the "placed pages" and the "unplaced pages." The "placed pages" are pages which are created and have been integrated to the site. "Unplaced Pages" are pages which are finished but are not integrated into the live site.
Workflow	Has the workflow configuration functions. By default, only users with the Workflow Admin role have access to this tab.

Appendix B

System Data: Content Server Database

This appendix contains information about the dynamic tables in the Content Server database and how they grow. DBAs can use this information to determine how to size the Content Server database appropriately.

This appendix contains the following sections:

This appendix also contains information about how to purge inactive data from the visitor tables.

Cache Management Tables

Content Server delivers the CacheManager, a page caching utility that manages both the Content Server page cache and the Satellite Server caches.

Because cached pages need to expire both when their freshness date expires and when an asset that the page refers to in some way is changed, the CacheManager keeps track of expiration times as well as the dependencies that exist between the pages and pagelets stored in the cache. It stores this information in the `SystemPageCache` and `SystemItemCache` tables.

Every CS system uses a CacheManager, which means that these tables grow dynamically on any system—development, management, or delivery. The cache-tracking tables grow at the following rate:

Table	Number of Rows
`SystemPageCache`	One row for every cached page. When a page expires, the row is removed.
`SystemItemCache`	One row for each asset, pagelet, or other item that is referenced by a cached page in the `SystemPageCache` table. For example, if a cached page was created from a page asset that has associations to three article assets, there would be four rows for that cached page.

Approval System Tables

The Content Server approval system keeps track of each asset that has been approved, the dependencies that approved assets have on other assets, and the targets for which assets are approved. It stores this information in the `ApprovedAssets` and `ApprovedAssetDeps` tables.

These tables have the potential to grow very large on a management system, but are not used on a delivery system. The approval system tables grow at the following rate:

Table	Number of Rows
`ApprovedAssets`	One row for each asset that has been approved, for each target destination. That is, if an asset has been approved for two destinations, that asset has two rows in this table.
`ApprovedAssetDeps`	One row for each asset dependency for each approved asset, for each target destination. For example, if an approved asset is dependent on four other assets, it has four rows in this table. If that same asset is approved to two destinations, it has one row each for each dependency for each destination.

Publishing System Tables

The Content Server publishing system keeps track of when assets were published, and where they were published to. It stores this information in the `PubKey` and the `PublishedAssets` tables.

As the number of assets in your CS management system increases, so does the number of rows in these tables. These tables grow at the following rate:

Table	Number of Rows
`PubKey`	• For mirror publishing: one row for every asset that is mirrored, for each target destination. • For export publishing: one row for each page that is created during the export. That is, if 14 assets are rendered into 1 page, the table holds 1 row—not 14—for the entire group of assets.
`PublishedAssets`	• For mirror publishing: one row for every asset that is mirrored, for each target destination. • For export publishing: one row for each asset that is exported. To continue the preceding example, if 14 assets are rendered onto 1 page, the table holds 1 rows (one for each asset) for that page.

Workflow Tables

The workflow system keeps track of all the assets that are involved in a workflow process at any given time. It stores this information in the `Assignment` and `WorkflowObject` tables.

As the number of assets that are placed in a workflow process increases, so does the number of rows in these tables. These tables grow at the following rate:

Table	Number of Rows
`Assignment`	One row for each workflow assignment. For example, if an asset is assigned to four users during the course of a workflow process, that asset has four rows in the table. These rows are not deleted when the workflow process is completed for the asset.
`WorkflowObjects`	One row for each asset in workflow. When an asset leaves workflow, the row is deleted.

Basic Asset Tables

A basic asset type has one primary storage table. For example, the primary storage table for the article asset type is named `Article`; the primary storage table for the `HelloArticle` asset type is named `HelloArticle`.

As the number of assets of a single type increases, so does the size of the table that holds assets of that type. The primary storage table for a basic asset has one row for each asset of that type.

Flex Asset Tables

Each asset type in a flex family has several database tables. The three types of tables in any flex family that can potentially grow quite large are as follows:

- The primary storage table for the **flex asset** type. For example, the primary storage table for the GE Lighting sample site asset type named product is `Products`.
- The `_AMap` tables for flex asset or flex parent asset types. (For example, `Products_AMap`.)
- The `_Group` tables for flex parent asset types. (For example, `ProductGroups_Group`.)
- The `_Mungo` table for the **flex asset** type. (For example, `Products_Mungo`.)
- The `_Mungo` table for the flex **parent** asset type. (For example, `ProductGroups_Mungo`.)
- The `Mungo_Blobs` table

These types of tables grow at the following rate:

Table	Number of Rows
`FlexAssetType` (for example: `Products`)	One row for every asset of this type.
`FlexAssetType``_AMap` (for example, `Products_AMap`)	One row per attribute value—whether the attribute value is inherited or directly assigned—for each of the assets of that type.
`FlexAssetType``_Group` (for example, `ProductGroups_Group)`	One row per ancestor relationship between flex parent asset and flex asset—includes rows for grandparent, great-grandparent, and so on, relationships.
`FlexAssetType``_Mungo` (for example: `Products_Mungo`)	One row for every attribute value for each asset of this type. Note that for one FatWire customer, this equation resulted in 2.4 million rows and for another, this equation resulted in over 10 million rows.
`FlexParent_Mungo` (for example: `ProductGroups_Mungo`)	One row for every attribute value for every parent asset of this type.

Table	Number of Rows
Mungo_Blobs	One row for every attribute value saved for an attribute of type blob.

Visitor Tables (Engage)

Engage captures visitor information and stores it in the visitor data tables. These tables store information such as session IDs for visitors so that they can be linked with their previous sessions and values for the attributes that represent the data you are collecting.

As the number of visitors who visit your online site increases, so do the rows in these tables. These tables grow at the following rate:

Table	Number of Rows
scratch	One row for each visitor context session object that is created for a visitor. Visitor context session objects are things like promotion lists, segment lists, shopping carts, and so on. There are at least five rows added to this table for each visitor in each session.
VMVISITOR	One row for each visitor for each browser session. Engage creates a unique visitor ID for each visitor for each session.
VMVISITORALIAS	Most likely, one row for each visitor for each browser session. A row holds the name/value pair of an alias and the visitor ID (also listed in VMVISITOR) that marks a session.
VMVISITORSCALARVALUE	One row for each visitor attribute value (except for attributes of type binary) that is saved.
VMVISTORSCALARBLOB	One row for each visitor attribute value of type binary (also referred to as scalar objects) that is saved.
VMz ------------	These are dynamically generated tables that are created when values for a history attribute are saved. Engage creates one table for each history type and adds a row to the table each time a record of that type is saved.

Managing the Attribute Tables

Because the tables that hold attribute values can grow very quickly, you should purge inactive data from them regularly.

You use the following Engage XML object method or its JSP equivalent to delete inactive data from these tables:

```
<VDM.FLUSHINACTIVE STARTDATE="cutoffDate"/>
```

Inactive visitor data is data marked with a visitor ID that is not connected through an alias to data that you consider current. You set a cutoff date (`STARTDATE`) for Engage to use. All visitor data recorded before that date is deleted from the previously listed visitor tables **unless** it is linked through an alias with data recorded after that date.

There are several ways to use the `VDM.FLUSHINACTIVE` tag. For example:

- You can create an administrative element that invokes the tag and prompts you to enter the cutoff date.
- You can provide it with an equation that calculates a cutoff date based on some parameter so that you can set it up to run as an automatic event at a regularly scheduled time. To set it up as an automatic event, use the Content Server `APPEVENT` tag (which functions like a `kron` job). For information about this tag, see the *Content Server Developer's Tag Reference*, and the "Coding Basics" section of the *Content Server Developer's Guide*.

Note that the value that you pass to the `STARTDATE` parameter must be in epoch time. You can use the Content Server `DATE.CONVERT` tag to obtain an epoch value for the date that you want to use. For example:

```
<DATE.CONVERT VARNAME="flushtime"
   YEAR="four digit year" MONTH="number in the range 1-12"
   DAY="number in the range 1-31"
   [HOUR="number in the range 0-11 where 0 is midnight" AMPM="am
   or pm" MINUTE="number in the range 0-59" TIMEZONE="timezone"]/>
<VDM.FLUSHINACTIVE STARTDATE="Variables.flushtime"/>
```

For more information about these tags, see the *Content Server Developer's Tag Reference*.

Managing the Session Objects Table (scratch)

The `scratch` table can grow quickly because there are at least five session objects stored for each visitor in each session. Each object has a timestamp; you should purge old objects regularly, based on their timestamps.

You use the following Engage XML object method or its JSP equivalent to delete old session objects from the scratch table:

```
<SESSIONOBJECTS.FLUSH TIMESTAMP="cutoffTime"/>
```

Because "session objects" includes carts, you must set the cutoff time to represent the point at which you consider a cart to be abandoned.

> **Note**
>
> This object method does not affect the `VMSCALARBLOB` table, which means that it does not delete carts that you have stored as a visitor attribute of type binary (a scalar object).

Note that the value that you pass to the `STARTDATE` parameter must be in epoch time. You can use the Content Server `DATE.CONVERT` tag to obtain an epoch value for the date that you want to use. For an example, see the code example for the `VDM.FLUSHINACTIVE` tag provided in the preceding section.

For more information about these tags, see the *Content Server Developer's Tag Reference*.

Deleting Unnecessary .class Files

Engage is a Java-based application and it generates Java `.class` files each time one of the following events occurs:

- A segment, recommendation, or promotion is created.
- A product is configured for a related items recommendation.
- A segment, recommendation, or promotion is calculated or invoked by Engage.

Typically, old `.class` files are deleted when a segment, recommendation, promotion, or product is updated and are then replaced with new `.class` files. However, if the segment, recommendation, product, or promotion is in use when an updated version is published, Engage cannot delete the old `.class` file because it is locked.

The old `.class` files can build up, filling up the disk and using memory. Therefore, depending on how much development work you are doing and how frequently you publish to the delivery system, you must manually delete these `.class` files at a regularly scheduled time.

Complete the following steps to delete old `.class` files:

1. Use the Property Editor to examine the `vis.genclasspath` property and note the directory name designated by that parameter. This is the directory where Engage stores `.class` files.
2. During a quiet time on your site, shut down and restart each instance of JRE runtime that is running. This process releases any old `.class` files that the JRE runtime has locked.
3. Using any file management tool, navigate to the directory that holds the `.class` files and delete the contents of this directory.

Engage regenerates any `.class` files that it needs when it needs them.

Appendix C

Sample Site Configurations

Content Server provides several sample sites that demonstrate the two sides of an online site: the online site itself, and the content management site on the management system, which the content providers interact with in order to supply content for the online site.

The sample sites vary in complexity and purpose:

- **FirstSite II** is FatWire's flagship reference implementation. Utilizing our best practices in Content Server site development, FirstSite II provides an excellent starting point for developers. As with all FatWire sample sites, FirstSite II code can be reused and modified to suit your organization's business needs. FirstSite II is described in detail in the guide, *FirstSite II: Use, Reuse, and Replication*.
- **Burlington Financial** is a complete, fully functioning financial services portal with a rich set of features. It uses the basic asset data model and illustrates most of the abilities of Content Server.
- **Hello Asset World** is an extremely simple site that illustrates the primary coding and design concepts behind extracting and displaying basic assets.
- **GE Lighting** is an online catalog of lighting products. It illustrates the power and flexibility of the flex asset model. When Engage is installed, GE Lighting also uses some of the Engage features.
- **Burlington Financial Engage Extension** provides an extension to the Burlington Financial site that illustrates the Engage features, and adds two families of flex asset types to the data model.
- **Spark** is a sample site that illustrates the content providers' portal environment.

Note

For detailed information about the Spark site and the business users' portal environment, see the *Content Server Portal Interface User's Guide*, where the Spark sample site is used throughout to demonstrate content management procedures. (Note that while administrative portlets are packaged with the Spark sample site, their purpose is to help CS administrators manage the Spark sample site, if it is installed.)

In addition to providing code samples for your site designers, all the sample sites serve as examples of how to configure sites that run on a management system.

This appendix describes the configuration of the sample sites in the following sections:

- Sample Users and Their ACLs
- Sample Roles
- Burlington Financial
- GE Lighting
- Burlington Financial Engage Extension

Sample Users and Their ACLs

Table C-1: Sample Site Users and their ACLs

User Name	ACLs	Description
Bobo	TableEditor, Visitor, VisitorAdmin, UserEditor, UserReader, xceladmin, Browser, xceleditor, PageEditor, ElementEditor, RemoteClient, ElementReader	The administrator of the Hello Asset World sample site.
Coco	PageEditor, ElementEditor, Visitor, VistorAdmin, TableEditor, UserReader, Browser, xceleditor, RemoteClient	The developer/designer for the Hello Asset World sample site.
editor	PageEditor, ElementEditor, Visitor, UserReader, Browser, xceleditor	A sample site user for the Burlington Financial sample site.
Flo	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	The editor for the Hello Asset World sample site.
Joe	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	One of the writers for the Hello Asset World sample site.
mirroruser	Browser, ElementEditor, PageEditor, TableEditor, UserReader, Visitor, VisitorAdmin, xceladmin, xceleditor	An example mirror user that is installed for the sample sites so that the sample site assets can be published with the Mirror to Server delivery method.
Moe	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	One of the writers for the Hello Asset World sample site.
user_analyst	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	A sample site data analyst user who creates segments. This user is installed for the Engage extensions to the Burlington Financial sample site.
user_approver	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	Another user for the Burlington Financial and GE sample sites. This user has the approver role, which means this user has the ability to approve assets for publishing.

Table C-1: Sample Site Users and their ACLs *(continued)*

User Name	ACLs	Description
user_author	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	An author user for the Burlington Financial and GE sample sites.
user_checker	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	A user who does fact checking for the Burlington Financial sample site.
user_designer	PageEditor, ElementEditor, Visitor, VistorAdmin, TableEditor, UserReader, Browser, xceleditor, RemoteClient	A site designer user for the sample sites (Burlington Financial and GE Lighting).
user_editor	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	An editor user for the Burlington Financial and GE sample sites.
user_expert	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	A sample site user who is the supervisor of both user_marketer and user_analyst. This user is installed for the Engage extensions to the Burlington Financial sample site.
user_marketer	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	A sample site user who works on segments and funds. This user is installed for the Engage extensions to the Burlington Financial sample site.
user_pricer	PageReader, ElementReader, Visitor, UserReader, Browser, xceleditor, RemoteClient	A sample site user who prices the products in the GE sample site.

Sample Roles

Table C-2: Default Sample Site Roles

Role	Description
Analyst	Used by the Burlington Financial with Engage extensions sample site in the Segment workflow process to designate users who are allowed to create segments. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.
Approver	Used by the Burlington Financial and GE Lighting sample sites in their workflow processes to designate the users who are allowed to approve assets for publishing. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.
Author	Allows users to do the following: • Burlington Financial – write articles as part of the Normal Article Process workflow. • GE Lighting – write flex articles, create products, and place products in workflow to obtain a price as part of the GE Lighting workflow process. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.
Checker	Allows users to do the following: • Burlington Financial – perform fact checking for articles as part of the Normal Article Process workflow. • Burlington Financial with Engage extensions – verify funds for the Fund Process and segments for the Segments Process. • GE Lighting – verify the prices set for products in the GE Lighting Process. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.
Designer	Allows users to create site design assets by giving them access to the **Design** tab, a custom tab created for the Burlington Financial and GE Lighting sample sites, and to design and configure asset types by giving them access to the **Admin** tab. The designer role does not participate in any of the sample workflow processes. Grants access to the following tree tabs in the Burlington Financial and GE Lighting sample sites: **Active List**, **Admin**, **Content**, **Design**, **Marketing**, **Product**, **Site Plan**.

Table C-2: Default Sample Site Roles *(continued)*

Role	Description
Editor	Allows users to do the following: • Burlington Financial – edit articles. • GE Lighting – review products and product prices. • Burlington Financial with Engage extensions—review funds and segments. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.
Expert	Used by the Burlington Financial with Engage extensions sample site in the Segment workflow process to designate the users who are allowed to create as well as approve segments. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.
HelloAuthor	Used by the Hello Asset World sample site to designate the users who are allowed to create HelloArticle assets (that is, write articles) and place them in workflow. Grants access to the following tabs: **Active List**, **HelloContent**, **Site Plan**.
HelloDesigner	Allows users to create site design assets by giving them access to the **HelloDesign** tab, a custom tab created for the Hello Asset World sample site, and to design and configure asset types by giving them access to the **Admin** tab. The HelloDesigner role does not participate in the sample workflow process for the Hello Asset World site. Grants access to the following tabs: **Active List**, **Admin HelloContent**, **HelloDesign**, **Site Plan**.
HelloEditor	Used by the Hello Asset World sample site to designate the users who edit the HelloArticles placed in workflow. Grants access to the following tabs: **Active List**, **HelloContent**, **HelloDesign**, **Site Plan**.
Marketer	Used by the Burlington Financial with Engage extensions sample site in both the workflow processes. Users with this role are allowed to create segments, create funds, and review funds. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.
Pricer	Used by the GE Lighting sample site in the GE Lighting workflow process to designate the users who are allowed to price products. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan**.

Hello Asset World

The Hello Asset World site is the simplest of the four sample sites. It has five users, six roles, two custom tabs, and one workflow process. For asset types, it uses only four of the system asset types and has only two custom asset types. The example users of this sample site provide articles and images for an online webzine called “Hello World.”

For a description of the site design of the Hello World webzine, see the *Content Server Developer’s Guide*.

Users, ACLs, and Roles for Hello Asset World

The Hello Asset World site has five users:

- Coco, the site designer
- Bobo, the site administrator
- Flo, the editor
- Joe, an author
- Moe, another author

There are no custom ACLS—the user accounts are assigned only the system default ACLs.

Hello Asset World users are assigned one or more of the following five roles:

- HelloAuthor
- HelloDesigner
- HelloEditor
- GeneralAdmin
- WorkflowAdmin

The GeneralAdmin and WorkflowAdmin roles are system default roles, while the other three are custom roles for the users of the Hello Asset World site.

The following tables shows how the Hello Asset World user accounts and role assignments are configured:

Username	Password	ACLs	Roles
Coco	hello	Browser, ElementEditor, ElementReader, PageEditor, PageReader, TableEditor, UserReader, xceladmin, xceleditor	HelloDesigner, GeneralAdmin (a system default role)
Bobo	hello	Browser, ElementReader, PageReader, UserEditor, UserReader, xceladmin, xceleditor	GeneralAdmin, WorkflowAdmin (another system default role)
Flo	hello	Browser, ElementReader, PageReader, UserReader, xceleditor	HelloEditor

Username	Password	ACLs	Roles
Joe	hello	Browser, ElementReader, PageReader, UserReader, xceleditor	HelloAuthor
Moe	hello	Browser, ElementReader, PageReader, UserReader, xceleditor	HelloAuthor

Asset Type Configuration for Hello Asset World

Asset type configuration for Hello Asset World is also quite simple. The site uses only four of the standard system default asset types—but adds an association to the page asset type—and provides two custom asset types (built with the AssetMaker feature).

These are the asset types that are used in Hello Asset World:

- HelloArticle (custom). It has one association field named Associated HelloImage. Moe and Joe, the Hello Asset World authors, create assets of this type. There are seven HelloArticle assets in the site.
- HelloImage (custom). Moe and Joe also create the HelloImages that illustrate their HelloArticles. There are eight HelloImage assets in the site.

> **Note**
>
> For information about the data structure of these asset types, use Content Server Explorer to examine their asset descriptor files and the HelloArticle and HelloImage database tables.
>
> The Content Server Explorer tool is described in the *Content Server Developer's Guide*.

- Template (default). There are three template assets: one for pages, one for HelloArticles, and one for collections.
- Page (default). The Hello Asset World site adds an association field named HelloAssetWorld Collection to the page asset type. The site has one page asset.
- Collection (default). The site has one collection asset.
- Query (default). The site has one query asset.

Source and Category

Hello Asset World does not use categories but it does add one new source item. This source is also named HelloAssetWorld.

Start Menu Configuration for Hello Asset World

Even though Hello Asset World uses only 6 asset types, it has 16 start menu items. In addition to the New and the Search start menu items for the six asset types that are in use, there are New and Search start menu items for the CSElement and SiteEntry asset types in case Coco the site designer ever decides to use them.

The start menu items are also very simple: only one of them provides a default value for an asset created with it. The following table describes all the start menu items for Hello Asset World.

Name	Asset Type	Start Menu Type	Roles	Default Values
New CSElement, HelloAssetWorld	CSElement	New	HelloDesigner	
Find CSElement, HelloAssetWorld	CSElement	Search	HelloDesigner	
New Collection, HelloAssetWorld	collection	New	HelloEditor, HelloDesigner	
Find Collection, HelloAssetWorld	collection	Search	HelloEditor, HelloDesigner	
New HelloArticle	HelloArticle	New	HelloAuthor	workflow = HelloWorkflow
New HelloArticle for Testing	HelloArticle	New	HelloDesigner, HelloEditor	
Find HelloArticle	HelloArticle	Search	HelloAuthor, HelloEditor, HelloDesigner	
New HelloImage	HelloImage	New	HelloAuthor, HelloEditor, HelloDesigner	
Find HelloImage	HelloImage	Search	HelloAuthor, HelloEditor, HelloDesigner	
New Page, HelloAssetWorld	page	New	HelloDesigner	
Find Page, HelloAssetWorld	page	Search	HelloDesigner	
New Query, HelloAssetWorld	query	New	HelloDesigner	
Find Query, HelloAssetWorld	query	Search	HelloDesigner	
New SiteEntry, HelloAssetWorld	SiteEntry	New	HelloDesigner	
Find SiteEntry, HelloAssetWorld	SiteEntry	Search	HelloDesigner	
New Template, HelloAssetWorld	template	New	HelloDesigner	
Find Template, HelloAssetWorld	template	Search	HelloDesigner	

Tree Tab Configuration for Hello Asset World

Hello Asset World adds two custom tabs to the tree and adds the Hello Asset World roles for its users to the system default tabs.

Tab	Description	Roles	Users with Access
Admin (default)	Provides access to the administrative features	GeneralAdmin	Bobo, Coco
Active List (default)	Provides access to the user's active list	HelloAuthor, HelloDesigner, HelloEditor	Coco, Flo, Joe, Moe
Hello Content (custom)	Provides access to the HelloArticle and HelloImage assets	HelloAuthor, HelloEditor, HelloDesigner	Coco, Flo, Joe, Moe
Hello Design (custom)	Provides access to the template, CSElement, SiteEntry, page, collection, and query assets	HelloDesigner, HelloEditor	Coco
History (default)	Displays the assets that the user worked with during the current session.	All	All
Site Plan (default)	Displays a graphical representation of the online site	HelloAuthor, HelloDesigner, HelloEditor	Coco, Flo, Joe, Moe
Workflow (default)	Provides access to the workflow states and processes	WorkflowAdmin	Bobo

Workflow for Hello Asset World

Hello Asset World provides a sample workflow process named HelloArticle Process.

It has two states and four steps. For the most part, this workflow process was built with the default workflow building blocks. However, one additional e-mail object and one additional step action were created for this workflow process.

For a complete description of the Hello Asset World workflow process, see "HelloArticle Process," on page 153.

Burlington Financial

The Burlington Financial sample site Burlington Financial is a fictitious financial news portal. As is the design of the online site, the configuration of the management system site is also more complex than that of Hello Asset World.

This site has seven users, eight roles, and one workflow process. For asset types, it uses all of the system default types and provides three custom asset types.

The example users of this sample site provide articles and images for an online financial news site and service called “Burlington Financial.” For a description of the site design of the Burlington Financial news site, see the *Content Server Developer’s Guide*.

Users, ACLS, and Roles for Burlington Financial

The Burlington Financial site has six users:

- user_approver, a manager who approves assets so that they can be published
- user_author, an author
- user_checker, a fact-checker
- user_designer, the site designer
- user_editor, an editor
- fwadmin, the site and system administrator (note that this is also a system default user)

There are no custom ACLS—the user accounts are assigned only the system default ACLs.

Burlington Financial users are assigned one or more of the following eight roles:

- Approver
- Author
- Checker
- Designer
- Editor
- GeneralAdmin
- SiteAdmin
- WorkflowAdmin

The GeneralAdmin, WorkflowAdmin, and SiteAdmin roles are system default roles, while the other five are custom roles for the users of the Burlington Financial site.

The following table shows how the Burlington Financial user accounts and role assignments are configured:

Username	Password	ACLs	Roles
user_author	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, xceleditor	Author
user_approver	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, xceleditor	Approver

Username	Password	ACLs	Roles
user_checker	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, xceleditor	Checker
user_designer	user	Browser, ElementEditor, PageEditor, RemoteClient, TableEditor, UserReader, xceleditor	Designer
user_editor	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, xceleditor	Editor
fwadmin	xceladmin	Browser, ElementEditor, TableEditor, PageEditor, RemoteClient, UserEditor, UserReader, xceladmin, xceleditor	GeneralAdmin, SiteAdmin, WorkflowAdmin

Asset Type Configuration for Burlington Financial

Asset type configuration for Burlington Financial is also more complex than that of Hello Asset World. The site uses all of the standard system default asset types, adds association fields to the page, collection, and query asset types, and provides two custom asset types built with the AssetMaker feature and one completely custom asset type built without AssetMaker (the article asset type). It also makes extensive use of categories and sources, and, for the article asset type, subtypes.

Note

For information about the data structure of the custom asset types, see the chapter on creating basic asset types in the *Content Server Developer's Guide* and then use Content Server Explorer to examine their asset descriptor files and their database tables.

The following table describes the asset type configuration for the Burlington Financial sample site:

Asset Type	Number of Assets	Association Fields	Subtype	Categories
Article (custom, non-AssetMaker type)	540	Main ImageFile, Popular Topics, Related Stories, Sidebar Bottom, Sidebar Top, Spot ImageFile, Spot ImageFile1, Teaser ImageFile, French Translation, German Translation, Spanish Translation	Standard, Columnist	Banking and Loans, Bonds, Companies, Consumer, Currency, Economy, Entertainment, French, Funds, General, German, Investing, Markets, News, Personal Finance, Portfolio, Rates and Bonds, Retirement, Services, Small Business, Spanish, Special, Sports, Stocks, Tax Guide, Technology, US Treasuries, World Business
CSElement (default)	42	none	none	none
Collection (default)	29	Query1, Query2, Query3	none	General, Money
ImageFile (custom)	105	none	none	General
Page (default)	21	HomeStoryGroups, PopularTopics, Recent Stories, Section Highlight, Sidebar Bottom, Sidebar Middle, Sidebar Top, Top Stories, WireFeed	none	Account Page, Content Page, Home, List Page, Section Front
Query (default)	28	none	none	Entertainment, General
SiteEntry (default)	9	none	none	none
StyleSheet (custom)	4	none	none	CSS HTML 4.0, XSL, General
Template(default)	47	none	none	Banner, Entertainment, General, Navigation, Page Template, Subpage Template

Sources

Burlington Financial uses the source feature to identify the news agency that provides an article when the article is not written by a Burlington Financial author. It adds the following 12 sources to the `Source` table:

- 123Jump
- AsiaPulse

- BurlingtonFinancial
- Efe
- FWNSelect
- ItarTass
- M2Comm
- NYPost
- TechWeb
- UPI
- WireFeed
- Xinhua

Start Menu Configuration for Burlington Financial

Burlington Financial provides 18 start menu items.

The start menu items are very simple: only one of the **New** items provides default values for an asset created with it and none of the **Search** items provide a default value to search by. The following table describes all the start menu items for Burlington Financial.

Name	Asset Type	Start Menu Type	Roles	Default Values
New Article	article	New	Approver, Author, Checker, Designer, Editor	
Find Article	article	Search	Approver, Author, Checker, Designer, Editor	
New CSElement	CSElement	New	Designer	
Find CSElement	CSElement	Search	Designer	
New Collection	collection	New	approver, author, checker, designer, editor	
Find Collection	collection	Search	Approver, Author, Checker, Designer, Editor	

Name	Asset Type	Start Menu Type	Roles	Default Values

New Columnist Article	article	New	Approver, Author, Checker, Designer, Editor	subtype = Columnist
				template = Columnist
				source = Burlington Financial

New ImageFile	image file	New	Approver, Author, Checker, Designer, Editor	
Search ImageFile	image file	Search	Approver, Author, Checker, Designer, Editor	
New Page	page	New	Designer	
Find Page	page	Search	Designer	
New Query	query	New	Designer	
Find Query	query	Search	Designer	
New SiteEntry	SiteEntry	New	Designer	
Find SiteEntry	SiteEntry	Search	Designer	
New Stylesheet	stylesheet	New	Author, Designer	
Find Stylesheet	stylesheet	Search	Author, Designer	
New Template	template	New	Designer	
Find Template	template	Search	Designer	

Tree Tab Configuration for Burlington Financial

Burlington Financial adds no custom tabs to the tree but it does add its roles to the system default tabs.

Tab	Description	Roles	Users with Access
Active List (default)	Provides access to the user's active list	Approver, Author, Checker, Designer, Editor	user_approver, user_author, user_checker, user_designer, user_editor
Admin (default)	Provides access to the administrative features	GeneralAdmin	fwadmin
History (default)	Displays the assets that the user worked with during the current session.	All	All
SiteAdmin (default)	Provides access to site administration functions	SiteAdmin	fwadmin
Site Plan (default)	Displays a graphical representation of the online site	Approver, Author, Checker, Designer, Editor	user_approver, user_author, user_checker, user_designer, user_editor
Workflow Admin (default)	Provides access to the workflow states and processes	WorkflowAdmin	fwadmin

Workflow for Burlington Financial

Burlington Financial provides a sample workflow process named Normal Article Process.

It has three states and six steps and uses four roles (author, approver, checker, and editor). This workflow process was built with the default workflow building blocks only.

For a complete description of the this workflow process, see “BF: Normal Article Process,” on page 154.

GE Lighting

The GE Lighting sample site is a complete example of a full-featured online catalog of lighting products. It uses the flex asset data model.

This site has 9 users, 10 roles, and 1 workflow process. For asset types, it uses all of the system default types and provides two flex families: the content family and the product family.

The example users of this sample site create products and articles about those products. For a description of the site design of the GE Lighting catalog, see the *Content Server Developer's Guide*.

Users, ACLS, and Roles for GE Lighting

The GE Lighting site has eight users:

- user_approver, a manager who approves assets so that they can be published
- user_author, an author
- user_checker, a fact checker
- user_designer, the site designer
- user_editor, an editor
- user_marketer, a marketing specialist
- user_pricer, a product specialist who prices the products in the catalog
- fwadmin, the site and system administrator (a system default user)
- editor, an editor who is allowed to perform all the roles on the site except the administrative roles (another system default user)

There are no custom ACLS—the user accounts are assigned only the system default ACLs.

GE Lighting users are assigned one or more of the following eight roles:

- Approver
- Author
- Checker
- Editor
- Designer
- GeneralAdmin
- Marketer
- Pricer
- SiteAdmin
- WorkflowAdmin

The GeneralAdmin, WorkflowAdmin, and SiteAdmin roles are system default roles, while the others are custom roles for the users of the GE Lighting sample site.

The following table shows how the GE Lighting user accounts and role assignments are configured:

Username	Password	ACLs	Roles
user_author	user	Browser, ElementReader, PageReader, UserReader, xceleditor	Author
user_approver	user	Browser, ElementReader, PageReader, UserReader, xceleditor	Approver
user_checker	user	Browser, ElementReader, PageReader, UserReader, xceleditor	Checker
user_designer	user	PageEditor, ElementEditor, Visitor, VistorAdmin, TableEditor, UserReader, Browser, xceleditor, RemoteClient	Designer
user_editor	user	Browser, ElementReader, PageReader, UserReader, xceleditor	Editor
user_marketer	user	Browser, ElementReader, PageReader, UserReader, xceleditor‘	Marketer, Designer
user_pricer	user	Browser, ElementReader, PageReader, UserReader, xceleditor	Pricer
fwadmin	xceladmin	TableEditor, UserEditor, UserReader, xceladmin, Browser, xceleditor, PageEditor, ElementEditor, RemoteClient	Designer, GeneralAdmin, SiteAdmin, WorkflowAdmin
editor	xceleditor	Browser, ElementReader, PageReader, UserReader, xceleditor	Approver, Author, Checker, Designer, Editor, Pricer, Marketer

Asset Type Configuration for GE Lighting

Asset type configuration for GE Lighting is much different than that of either Hello Asset World or Burlington Financial. This site uses the flex asset data model and provides two example flex families: the content family and the product family.

Because the GE Lighting site uses flex families for its content, it does not use associations or categories. It also does not use sources.

The content family includes the following asset types:

- content attribute
- content parent definition
- content parent
- content definition
- article (flex)
- image (flex)

There are four content attribute assets, one content definition asset, and three flex article assets.

The product family includes the following asset types:

- product attribute
- product parent definition
- product parent
- product definition
- product

There are 35 product attribute assets, 3 product parent definition assets, 31 product parent assets, 2 product definition assets, and 101 products.

Note

For information about the data structure of these asset types, see the chapter on creating flex asset types in the *Content Server Developer's Guide* and then use Content Server Explorer to examine the database tables for these asset types.

The following asset types are also enabled for use in the GE Lighting sample site

- attribute editor
- page
- template

There 2 attribute editors, 10 page assets, and 15 template assets.

If Engage is installed on your system, the GE Lighting sample site also provides 1 history attribute, 1 history definition, 2 visitor attributes, 1 segment, 1 promotion, and 3 recommendations.

Start Menu Configuration for GE Lighting

GE Lighting uses 42 start menu items. In addition to the **New** and **Search** start menu items for the 13 asset types that are in use, there are **New** and **Search** start menu items for the CSElement and SiteEntry asset types in case user_designer (the site designer) ever decides to use them.

The start menu items are very simple: none of the **New** items provide a default value for an asset created with it and none of the **Search** items provide a default value to search by. The following table describes all the start menu items used by the GE Lighting site.

Name	Asset Type	Start Menu Type	Roles
New Article (flex)	article (flex)	New	Author, Approver, Editor, Checker
Find Article (flex)	article (flex)	Search	Author, Approver, Editor, Checker
New Attribute Editor	attribute editor	New	Designer
Find Attribute Editor	attribute editor	Search	Designer
New CSElement	CSElement	New	Designer
Find CSElement	CSElement	Search	Designer
New Content Attribute	content attribute	New	Designer
Find Content Attribute	content attribute	Search	Designer
New Content Definition	content definition	New	Designer
Find Content Definition	content definition	Search	Designer
New Content Parent	content parent	New	Designer
Find Content Parent	content parent	Search	Designer
New Content Parent Definition	content parent definition	New	Designer
Find Content Parent Definition	content parent definition	Search	Designer
New History Attribute	history attribute	New	Designer
Find History Attribute	history attribute	Search	Designer
New History Definition	history definition	New	Designer
Find History Definition	history definition	Search	Designer
New Image (flex)	image (flex)	New	Author, Approver, Editor, Checker
Find Image (flex)	image (flex)	Search	Author, Approver, Editor, Checker
New Page	page	New	Designer

Name	Asset Type	Start Menu Type	Roles
Find Page	page	Search	Designer
New Product	product	New	Author, Approver, Checker, Designer, Editor, Pricer
Find Product	product	Search	Author, Approver, Checker, Designer, Editor, Pricer
New Product Attribute	product attribute	New	Designer
Find Product Attribute	product attribute	Search	Designer
New Product Definition	product definition	New	Designer
Find Product Definition	product definition	Search	Designer
New Product Parent	product parent	New	Designer
Find Product Parent	product parent	Search	Designer
New Product Parent Definition	product parent definition	New	Designer
Find Product Parent Definition	product parent definition	Search	Designer
New Promotion	promotion	New	Marketer, Designer
Find Promotion	promotion	Search	Marketer, Designer
New Recommendation	recommendation	New	Marketer, Designer
Find Recommendation	recommendation	Search	Marketer, Designer
New Segment	segment	New	Marketer, Designer
New Segment	segment	New	Marketer, Designer
New SiteEntry	SiteEntry	New	Designer
Find SiteEntry	SiteEntry	Search	Designer
New Template	template	New	Designer
Find Template	template	Search	Designer

Tree Tab Configuration

GE Lighting adds custom tabs to the tree and adds its roles to the system default tabs.

Tab	Description	Roles	Users with Access
Admin (default)	Provides access to the administrative features	GeneralAdmin	fwadmin

Tab	Description	Roles	Users with Access
Active List (default)	Provides access to the user's active list	Author, Approver, Checker, Designer, Editor, Pricer	all users
Content (custom)	Provides access to the images and articles on the GE Lighting site	Author, Approver, Checker, Editor, Pricer	user_approver, user_author, user_checker, user_editor, user_pricer, editor
Design (custom)	Provides access to the attribute editor, template, CSElement, SiteEntry, and the data design asset types.	Designer	user_designer, ladmin
History (default)	Displays the assets that the user worked with during the current session.	All	All
Marketing (default for Engage)	Provides access to the segment, promotion, and recommendation asset types.	Approver, Author, Checker, Designer, Editor, Pricer	user_approver, user_author, user_checker, user_designer, user_editor, user_pricer, editor
Product (custom)	Provides access to the products and product parents on the GE Lighting site	Author, Approver, Checker, Editor, Pricer	user_approver, user_author, user_checker, user_editor, user_pricer, editor
SiteAdmin	Provides access to site administration functions	SiteAdmin	fwadmin
Site Plan (default)	Displays a graphical representation of the online site	Author, Approver, Checker, Designer, Editor, Pricer	all users
Workflow Admin (default)	Provides access to the workflow states and processes	WorkflowAdmin	fwadmin

Workflow

GE Lighting provides a sample workflow process named GE Lighting Process.

It has three states and six steps. This workflow process was built with the default workflow building blocks only.

For a complete description of the this workflow process, see "GE Lighting Process," on page 157.

Burlington Financial Engage Extension

The Engage extension to the Burlington Financial sample site illustrates the Engage features. It creates a new section for the Burlington Financial site that provides information about mutual funds and uses the segment and recommendation asset types to market those funds to the Burlington Financial site visitors.

Users, ACLS, and Roles for Burlington Financial Extension

The Engage extension adds the following users to the Burlington Financial site:

- user_analyst, the market data analyst who creates determines the segments
- user_expert, the superviser of both user_marketer and user_analyst
- user_marketer, the marketer who works on segments and funds

There are no custom ACLs—these users are assigned only the system default ACLs. However, because Engage adds two new system ACLs (Visitor and VisitorAdmin), those ACLs are assigned to these new users as well as all the existing Burlington Financial users.

There are also three new roles for those users:

- Analyst
- Expert
- Marketer

The following table shows how the user accounts and role assignments are configured for the extension to the Burlington Financial site:

Username	Password	ACLs	Roles
user_analyst	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Analyst
user_approver	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Approver
user_author	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Author
user_checker	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Checker

Username	Password	ACLs	Roles
user_designer	user	Browser, ElementEditor, PageEditor, RemoteClient, TableEditor, UserReader, Visitor, VistorAdmin, xceleditor,	Designer
user_editor	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Editor
user_expert	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Expert
user_marketer	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Marketer, Designer
fwadmin	xceladmin	Browser, ElementEditor, PageEditor, RemoteClient, TableEditor, UserEditor, UserReader, Visitor, VisitorAdmin, xceladmin, xceleditor	GeneralAdmin, SiteAdmin, WorkflowAdmin

Asset Type Configuration for Burlington Financial Extension

The Engage extension adds two flex asset families to the Burlington Financial sample site: the product family and the content family.

- The mutual funds that are featured on the site are products that are created with a product flex family.
- Some of the information that is displayed for those products is created by using the content flex family.

The members of the product family are as follows:

- product attribute
- product parent definition
- product parent (fund parent)
- product definition
- product (fund), a flex asset type
- PDF, another flex asset type

The site provides 28 product attributes, 10 product parent definitions, 2 product definitions (one for each of the flex asset types, product and PDF), 61 product parents (fund parents), 30 products (funds), and 1 PDF asset.

The members of the content family are as follows:

- content attribute
- content parent definition
- content parent
- content definition
- drill hierarchy, the flex asset type

There are two content attributes, one content definition, and nine drill hierarchy assets. There are no content parent definitions or content parents.

Note

For information about the data structure of these asset types, see the chapter on creating flex asset types in the *Content Server Developer's Guide* and then use Content Server Explorer to examine the database tables for these asset types.

The following Engage asset types are also used for the site:

- history attribute
- history definition
- recommendation
- segment
- visitor attribute

The following two asset types are enabled for the site, in case the site designer or marketers ever decide to add them to the site:

- attribute editor
- promotion

Sources

The Engage extension to the Burlington Financial site adds no sources.

Start Menu Configuration for Burlington Financial Extension

The Engage extension to the Burlington Financial site provides an additional 36 start menu items.

The start menu items are very simple: none of the **New** items provide a default value for an asset created with it and none of the **Search** items provide a default value to search by. The following table describes the additional start menu items provided by Burlington Financial Extension.

Name	Asset Type	Start Menu Type	Roles
New Attribute Editor	attribute editor	New	Designer
Find Attribute Editor	attribute editor	Search	Designer
New Content Attribute	content attribute	New	Designer
Find Content Attribute	content attribute	Search	Designer
New Content Definition	content definition	New	Designer
Find Content Definition	content definition	Search	Designer
New Content Parent	content parent	New	Designer
Find Content Parent	content parent	Search	Designer
New Content Parent Definition	content parent definition	New	Designer
Find Content Parent Definition	content parent definition	Search	Designer
New Drill Hierarchy	drill hierarchy	New	Designer
Find Drill Hierarchy	drill hierarchy	Search	Designer
New History Attribute	history attribute	New	Designer
Find History Attribute	history attribute	Search	Designer
New History Definition	history definition	New	Designer
Find History Definition	History Definition	Search	Designer
New PDF	PDF	New	Analyst, Expert, Designer, Marketer

Name	Asset Type	Start Menu Type	Roles
Find PDF	PDF	Search	Analyst, Expert, Designer, Marketer
New Product	product	New	Analyst, Expert, Designer, Marketer
Find Product	product	Search	Analyst, Expert, Designer, Marketer
New Product Attribute	product attribute	New	Designer
Find Product Attribute	product attribute	Search	Designer
New Product Definition	product definition	New	Designer
Find Product Definition	product definition	Search	Designer
New Product Parent	product parent	New	Designer
Find Product Parent	product parent	Search	Designer
New Product Parent Definition	product parent definition	New	Designer
Find Product Parent Definition	product parent definition	Search	Designer
New Recommendation	recommendation	New	Analyst, Expert, Designer, Marketer
Find Recommendation	recommendation	Search	Analyst, Expert, Designer, Marketer
New Segment	segment	New	Analyst, Expert, Designer, Marketer
Find Segment	segment	Search	Analyst, Expert, Designer, Marketer
New Visitor Attribute	visitor attribute	New	Designer
Find Visitor Attribute	visitor attribute	Search	Designer

Tree Tab Configuration for Burlington Financial Extension

The Engage extension to the Burlington Financial site adds three custom tabs to the tree and adds its roles to the system default tabs.

Tab	Description	Roles Assigned	Users with Access
Active List (default)	Provides access to the user's active list	Approver, Author, Checker, Designer, Editor	user_approver, user_author, user_checker, user_designer, user_editor
Admin (default)	Provides access to the administrative features	GeneralAdmin	fwadmin
Content (custom)	For future use.	Analyst, Approver, Author, Checker, Designer, Editor	user_approver, user_author, user_checker, user_designer, user_editor
Design (custom)	Provides access to the site design, data design, and visitor data design asset types	Designer	user_designer
History (default)	Displays the assets that the user worked with during the current session.	All	All
Marketing (default with Engage)	Provides access to the promotion, recommendation, and segment asset types	Analyst, Designer, Expert, Marketer	user_analyst, user_designer, user_expert, user_marketer
SiteAdmin (default)	Provides access to site administration functions	SiteAdmin	fwadmin
Site Plan (default)	Displays a graphical representation of the online site	Approver, Author, Checker, Designer, Editor	user_approver, user_author, user_checker, user_designer, user_editor
Workflow Admin (default)	Provides access to the workflow states and processes	WorkflowAdmin	fwadmin

Workflow for Burlington Financial Extension

The Engage extension to the Burlington Financial site provides two sample workflow processes, Fund Parent Process and Segment Process:

- Fund Parent Process has three states, seven steps, and five roles.
- Segment Process has three states, five steps, and five roles.

These workflow processes are built with the default workflow building blocks only.

Appendix D

Managing Users, Sites, and Roles in LDAP-Integrated CS Systems

This appendix is for Content Server administrators who are managing users, ACLs, sites and roles from the Advanced interface of an LDAP-integrated CS system and need to know how their operations affect both the CS database and the LDAP server. This appendix summarizes:

- Which operations are propagated to the LDAP server (given the type of LDAP server, the operation, and whether you are connected as user with read-only or read-write permissions)
- Which operations produce errors
- How to correct errors

This appendix contains the following sections:

Overview

Content Server can be integrated with LDAP servers that use the LDAP-2 protocol or any other protocol (although, in the latter case, FatWire does not support write operations from the Advanced interface to the LDAP server).

This overview summarizes

- the types of LDAP schema that Content Server supports,
- operations that can be performed from the CS interface when each type of schema is deployed, and
- the outcome of the operations, given the permissions of the Content Server administrator to the LDAP server and the nature of the LDAP server.

LDAP Schema

Four LDAP-schema scenarios are covered in this appendix: hierarchical- and flat-schema LDAP in both web and portal installations. Each schema supports selected operations and installations:

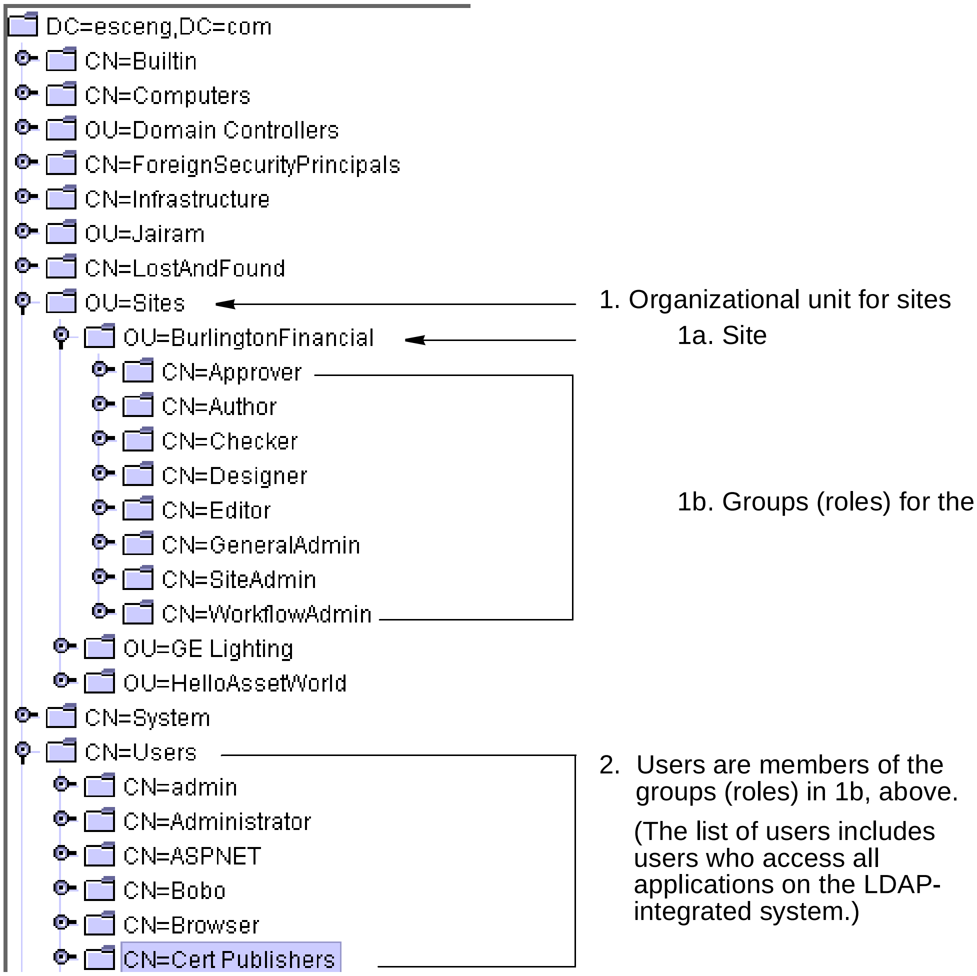
- Both hierarchical- and flat-schema LDAP support operations on ACLs and users (user accounts, user profiles, and user attributes) in CS **web and portal applications**.
- Hierarchical-schema LDAP supports operations on sites and roles in CS **web applications.** This schema requires LDAP users to define a site organizational unit in the LDAP server in which roles must be subordinated to their relevant sites (for an example, see Figure D-1, "LDAP Hierarchies").
- Flat-schema LDAP supports operations on sites and roles in CS **portal applications**. Flat-schema LDAP levels sites and roles so that no hierarchy exists among them.

Table D-1 summarizes LDAP schema and the possible operations. Using the correct schema, the CS administrator can perform operations in the CS interface and have them propagate to the Content Server database, or the LDAP server, or both, depending also on the conditions that are outlined in "LDAP Connectivity for Site and Role Management," on page 405.

Table D-1: LDAP Integration Scenarios

Type of Installation	Create/Modify/Delete Operations on ...	Flat-Schema LDAP	Hierarchical-Schema LDAP
CS web application	ACLs, Users, User Profiles, User Attributes	Table D-2	Table D-2
	Sites	Table D-3	Table D-4
	Roles	Table D-3	Table D-4
CS portal application	ACLs, Users, User Profiles, User Attributes	Table D-2	
	Sites	Table D-3	
	Roles	Table D-3	

Figure D-1: LDAP Hierarchies



```
DC=esceng,DC=com
  CN=Builtin
  CN=Computers
  OU=Domain Controllers
  CN=ForeignSecurityPrincipals
  CN=Infrastructure
  OU=Jairam
  CN=LostAndFound
  OU=Sites
    OU=BurlingtonFinancial
      CN=Approver
      CN=Author
      CN=Checker
      CN=Designer
      CN=Editor
      CN=GeneralAdmin
      CN=SiteAdmin
      CN=WorkflowAdmin
    OU=GE Lighting
    OU=HelloAssetWorld
  CN=System
  CN=Users
    CN=admin
    CN=Administrator
    CN=ASPNET
    CN=Bobo
    CN=Browser
    CN=Cert Publishers
```

1. Organizational unit for sites

1a. Site

1b. Groups (roles) for the site

2. Users are members of the groups (roles) in 1b, above.

(The list of users includes users who access all applications on the LDAP-integrated system.)

LDAP Connectivity for Site and Role Management

For operations on roles and sites to work as described in this appendix, LDAP connectivity must be enabled by setting the values of two properties in the `futuretense_xcel.ini` file:

```
xcelerate.usermanagerclass
xcelerate.rolemanagerclass
```

These properties allow you to manage sites and roles directly in the LDAP server (as shown in this appendix) or exclusively in the Content Server database.

- If the property values specify LDAP, then LDAP connectivity is established. Sites and roles can be managed in the LDAP server (and in the Content Server database).
    - For hierarchical-schema LDAP, only the `xcelerate.usermanagerclass` property must specify LDAP. The `xcelerate.rolemanagerclass` property uses the default value. For example:

```
xcelerate.usermanagerclass=com.openmarket.xcelerate.
 user.LDAPSchemaUserManager
```

```
xcelerate.rolemanagerclass=com.openmarket.xcelerate.
 roles.RoleManager
```

- For flat-schema LDAP, both properties must specify LDAP. For example:

```
xcelerate.usermanagerclass=com.openmarket.xcelerate.
 user.FlatLDAPSchemaUserManager
```

```
xcelerate.rolemanagerclass=com.openmarket.xcelerate.
 roles.FlatLDAPSchemaRoleManager
```

- If the property values do not specify LDAP (for example, `xcelerate.usermanagerclass=com.openmarket.xcelerate.user.usermanager)` then sites and roles can managed only in the Content Server database.

LDAP Users and Their Permissions to LDAP Servers

In an LDAP-integrated CS system, the administrator of the Content Server system may or may not be an administrative user of the LDAP server, depending on the value of the `jndi.connectAsUser` property in the `dir.ini` file. The value determines how the Content Server administrator is connected to the LDAP server, and therefore defines the LDAP user:

- If `jndi.connectAsUser` is set to `true`, then Content Server defines the LDAP user to be the same one that is logged in to Content Server and connects that user to the LDAP server.
- If `jndi.connectAsUser` is set to `false`, then Content Server defines the LDAP user to be the one that is specified in the `jndi.login` property (in `dir.ini`) and connects that user to the LDAP server.

For the connection to take place, the user and his permissions must also be defined in the LDAP server. If the user has read-only permissions to the LDAP server, he is not an administrative LDAP user. If the user has read and write permissions to the LDAP server, he is an administrative LDAP user (or simply, an LDAP administrator).

> **Note**
>
> For more information about LDAP-related properties, see the *Content Server Property Files Reference*.

LDAP-Integrated Operations

For the Content Server administrator to successfully perform an operation (such as creating an ACL in both the Content Server database and the LDAP server), it is critical for Content Server to be properly integrated with the LDAP server. Barring integration issues, the outcome of an operation depends on the following factors:

1. The LDAP user, as defined by the `jndi.connectAsUser` property (see the previous section).
2. The LDAP user’s permissions, as defined in the LDAP server.

   If the CS administrator is connected to the LDAP server as a user without administrative rights, his operations (such as deleting an ACL from the Content Server database) cannot be written to the LDAP server. The system responds by either writing the operations to the Content Server database, or not at all. The operations must then be repeated in the Advanced interface and performed manually in the LDAP server by an LDAP administrator.

3. The nature of the operation.

   Certain operations that are performed in the Advanced interface (such as editing an ACL) are not written to the LDAP server, even when the Content Server administrator is connected with write permissions.

4. Whether the LDAP server supports the LDAP-2 protocol.

   If the LDAP server does not support the LDAP-2 protocol, then FatWire does not support write operations from the Advanced interface to the LDAP server, and the result of an operation cannot always be predicted.

Tables D-2 through D-4 summarize the results of operations that can be performed by a Content Server administrator who is connected to the LDAP server as a user with read-only and read/write permissions.

User Management Operations

Table D-2, on page 408 applies to flat- and hierarchical-schema LDAP web and portal environments. Table D-2 summarizes system response to user management operations. The operations are performed by a Content Server administrator using the interfaces of three Content Server systems, each integrated with one of the following LDAP options:

- LDAP-2 server with read-only permissions for the LDAP user (defined on page 405)
- LDAP-2 server with write permissions for the LDAP user
- LDAP server other than LDAP-2, in which case write operations are not supported

The results of each operation are described on the pages that are noted in the left-hand column of Table D-2.

Note

The term “user management” in this appendix includes the management of ACLs, user accounts, user profiles, and user attributes.

Table D-2: System Response to User Management Operations in Content Server (Web and Portal Applications)

Operation in Content Server / Result	System Response With:		
	LDAP-2: User has Read-Only Permissions [1]	**LDAP-2: User has Write Permissions [2]**	**Not LDAP-2: Write not Supported [3]**
Creating an ACL (p. 409) *ACL is created in:*	Returns error[4]	LDAP + CS database	Unpredictable
Editing an ACL (p. 409) *ACL is modified in:*	CS database	CS database	CS database
Deleting an ACL (p. 409) *ACL is deleted in:*	Returns error	LDAP + CS database	Unpredictable
Assigning ACLs to Tables (p. 410) *ACL is assigned in:*	CS database	CS database	CS database
Assigning ACLs to Content Server Pages (Site Catalog Page Entries) (p. 410) *ACL is assigned in:*	CS database	CS database	CS database
Creating a User (Granting ACLs for Access Privileges) (p. 410) *User is created in:*	Returns error	LDAP	Unpredictable
Editing a User (p. 411) *User is modified in:*	Returns error	LDAP	Unpredictable
Deleting a User (p. 411) *User is deleted from:*	Returns error	LDAP	Unpredictable
Creating/Editing a User Profile (p. 412) *User profile is created/edited in:*	Returns error	LDAP	Unpredictable
Deleting a User Profile (p. 412) *User profile is deleted from:*	Returns error	LDAP	Unpredictable
Creating User Attributes (p. 413) *User profile is created in:*	Returns error	LDAP	Unpredictable
Editing/Deleting User Attributes (p. 413) *User profile is deleted from:*	Returns error	LDAP	Unpredictable

[1] ACLs and users cannot be stored in the LDAP server (the Content Server administrator is connected to the LDAP server as a user without write permissions).

[2] ACLs and users can be stored in the LDAP server (the Content Server administrator is connected to the LDAP server as a user with write permissions).

[3] The LDAP server does not support the LDAP-2 protocol (such as Netscape Server). For non-compliant LDAP servers, FatWire does not support write operations from the Advanced interface to the LDAP server.

[4] "Returns error" means that the operation is not performed (the system returns an error).

Creating an ACL

Operation

The Content Server administrator attempts to create an ACL by using the “Content Server Management Tools” node on the “Admin” tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot create the ACL. The LDAP administrator must first create the ACL manually in the LDAP server and then in the Content Server database.
2. LDAP-2 Server with Write Permissions—The system creates the ACL in both the LDAP server and the Content Server database.
3. Not LDAP-2: Write not Supported—Response is unpredictable. The LDAP administrator must first create the ACL in the LDAP server and then in Content Server.

Editing an ACL

Operation

The Content Server administrator attempts to modify an ACL by using the “Content Server Management Tools” node on the “Admin” tab.

Note

The Content Server administrator can modify either the description of the ACL, or the permission associated with the ACL, but not the ACL’s name.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system modifies the ACL in the Content Server database. The modification is stored only in the database, even with LDAP integration.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Deleting an ACL

Operation

The Content Server administrator attempts to delete an ACL by using the “Content Server Management Tools” node on the “Admin” tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot delete the ACL. The LDAP administrator must first delete the ACL from the LDAP server and then from the Content Server database.

2. LDAP-2 Server with Write Permissions—The system deletes the ACL from the LDAP server and from the Content Server database.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must first delete the ACL from the LDAP server before attempting to delete the ACL from Content Server.

Assigning ACLs to Tables

Operation

The Content Server administrator attempts to assign an ACL to system tables, using the "Content Server Management Tools" node on the "Admin" tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system assigns the ACLs to the database tables. If the ACLs cannot be assigned, the system returns an error message.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Assigning ACLs to Content Server Pages (Site Catalog Page Entries)

Operation

The Content Server administrator attempts to assign an ACL to Content Server Pages by using the "Content Server Management Tools" node on the "Admin" tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system adds the selected ACLs for the specified Content Server Page. If the ACLs cannot be assigned, the system returns an error message.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Creating a User (Granting ACLs for Access Privileges)

Operation

The Content Server administrator attempts to create a user by using the "Content Server Management Tools" node on the "Admin" tab.

Note

Content Server displays only the ACLs that are stored in its database.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot add the user and the user's assigned ACLs. The LDAP administrator must use the LDAP server to manually add the user to the LDAP server and assign ACLs to the user.
2. LDAP-2 Server with Write Permissions—The system creates the user in the LDAP server and assigns ACLs to the user.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must use the LDAP server to manually add the user to the LDAP server and assign ACLs to the user.

Editing a User

Operation

The Content Server administrator attempts to modify the ACLs that are assigned to a user or to change the user's password by using the "Content Server Management Tools" node on the "Admin" tab

> **Note**
>
> The "Modify user" operation allows you to change only the user password and the ACLs that are associated with the user.
>
> The "Modify user attributes" operation allows you to change all attributes that are stored in the database or in the LDAP server.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot modify the user's password and/or ACLs. The LDAP administrator must manually modify the user in the LDAP server.
2. LDAP-2 Server with Write Permissions—The system modifies the user's password and/or assigned ACLs in the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must manually modify the user in the LDAP server.

Deleting a User

Operation

The Content Server administrator attempts to delete a user by using the "Content Server Management Tools" node on the "Admin" tab.

Before deleting a user, the administrator must delete the user profile. If a user profile exists, an error message is triggered. The behavior described below is expected when no user profile exists for the user being deleted.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot delete the user. The LDAP administrator must manually delete the user from LDAP server.
2. LDAP-2 Server with Write Permissions—The system deletes the user from the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must manually delete the user from the LDAP server.

Creating/Editing a User Profile

Operation

The Content Server administrator attempts to create or modify a user profile by using the “User Profiles” node on the “Admin” tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot create/modify the user’s profile and/or Locale Preference. The LDAP administrator must manually create/modify the user profile in the LDAP server.
2. LDAP-2 Server with Write Permissions—The system creates/modifies the user’s profile and/or locale preference in the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must manually create/modify the user profile in the LDAP server.

Deleting a User Profile

Operation

The Content Server administrator attempts to delete a user profile by using the “User Profiles” node on the “Admin” tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot delete the user’s profile. The LDAP administrator must manually delete the user profile in the LDAP server.
2. LDAP-2 Server with Write Permissions—The system deletes the user’s profile from the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must manually delete the user profile from the LDAP server.

Creating User Attributes

Note

User attributes are stored in only one place—either in the database or in LDAP.

Operation

The Content Server administrator attempts to create new attributes for the user from the "Content Server Management Tools" node of the "Admin" tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot add the new attributes to the LDAP server. The LDAP administrator must manually add user attributes to the LDAP server.
2. LDAP-2 Server with Write Permissions—The system adds the user's new attributes to the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must manually add user attributes to the LDAP server.

Editing/Deleting User Attributes

The Content Server administrator attempts to modify the user attributes from the "Content Server Management Tools" node of the "Admin" tab.

System Response

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot modify or delete the user's attributes from the LDAP server. The LDAP administrator must manually modify user attributes in the LDAP server and delete user attributes from the LDAP server.
2. LDAP-2 Server with Write Permissions—The system modifies the user attributes in the LDAP server and deletes them from the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The LDAP administrator must manually modify user attributes in the LDAP server, and manually delete user attributes from the LDAP server.

Site and Role Management Operations

This section covers operations with hierarchical- and flat-schema LDAP servers.

Operations with Flat-Schema LDAP Servers

Table D-4 applies to the portal environment, and summarizes system response to site and role management operations. The operations are performed by a Content Server administrator using the interfaces of three Content Server systems, each integrated with one of the following LDAP options:

- An LDAP-2 server with read-only permissions for the LDAP user (defined on page 405)
- An LDAP-2 server with write permissions for the LDAP user
- An LDAP server other than LDAP-2, in which case write operations are not supported

The results of each operation are described on the pages that are noted in the left-hand column of Table D-4.

Table D-3: System Response to Site and Role Management Operations with Flat-Schema LDAP (Content Server Portals)

Operation in Content Server	Result	System Response with Flat-Schema LDAP:		
		LDAP-2: User has Read-Only Permissions [1]	LDAP-2: User has Write Permissions [2]	Not LDAP-2: Write not Supported [3]
Creating a Site (p. 416)	*Site is created in:*	CS database	CS database	CS database
Editing a Site (p. 416)	*Site is edited in:*	CS database	CS database	CS database
Deleting a Site (p. 417)	*Site is deleted from:*	CS database	CS database	CS database
Creating a Role (p. 419)	*Role is created in:*	Returns error[4]	LDAP + CS database[5]	Unpredictable
Editing a Role (p. 419)	*Role is edited in:*	CS database	CS database	CS database
Deleting a Role (p. 420)	*Role is deleted from:*	Returns error	LDAP + CS database	Unpredictable
Granting Users Access to Sites (p. 417)	*Access is granted in:*	Returns error	LDAP	Unpredictable
Removing Users' Access to Sites (p. 418)	*Access is removed from:*	Returns error	LDAP	Unpredictable

1. Sites and roles cannot be stored in the LDAP server (the Content Server administrator is connected to the LDAP server as a user without write permissions).
2. Sites and roles can be stored in the LDAP server (the Content Server administrator is connected to the LDAP server as a user with write permissions).
3. The LDAP server does not support the LDAP-2 protocol (such as Netscape Server). For non-compliant LDAP servers, FatWire does not support write operations from the Advanced interface to the LDAP server.
4. "Returns error" means that the operation is not performed (the system returns an error).
5. For a role to be created in both the LDAP server *and* the CS database, at least one site must exist in the LDAP server. If no site exists, the role is created only in the CS database.

Operations with Hierarchical-Schema LDAP Servers

Table D-4 applies to the web environment and summarizes system response to site and role management operations. The operations are performed by a Content Server administrator using the interfaces of three Content Server systems, each integrated with one of the following LDAP options:

- An LDAP-2 server with read-only permissions for the LDAP user (defined on page 405)
- An LDAP-2 server with write permissions for the LDAP user
- An LDAP server other than LDAP-2, in which case write operations are not supported

The results of each operation are described on the pages that are noted in the left-hand column of Table D-4.

Table D-4: System Response to Site and Role Management Operations with Hierarchical-Schema LDAP (Content Server Web Applications)

		System Response With Hierarchical-Schema LDAP:		
Operation in Content Server	**Result**	**LDAP-2: User has Read-Only Permissions** [1]	**LDAP-2: User has Write Permissions** [2]	**Not LDAP-2: Write not Supported** [3]
Creating a Site (p. 416)	*Site is created in:*	CS database	CS database	CS database
Editing a Site (p. 416)	*Site is edited in:*	CS database	CS database	CS database
Deleting a Site (p. 417)	*Site is deleted in:*	CS database	CS database	CS database
Creating a Role (p. 419)	*Role is created in:*	CS database	CS database	CS database
Editing a Role (p. 419)	*Role is edited in:*	CS database	CS database	CS database
Deleting a Role (p. 420)	*Role is deleted in:*	CS database	CS database	CS database
Granting Users Access to Sites (p. 417)	*Access is granted in:*	Returns error[4]	LDAP	Unpredictable
Removing Users' Access to Sites (p. 418)	*Access is removed from:*	neither the CS database nor the LDAP server	neither the CS database nor the LDAP server	neither the CS database nor the LDAP server

1. Sites and roles cannot be stored in the LDAP server (the Content Server administrator is connected to the LDAP server as a user without write permissions).
2. Sites and roles can be stored in the LDAP server (the Content Server administrator is connected to the LDAP server as a user with write permissions).
3. The LDAP server does not support the LDAP-2 protocol (such as Netscape Server). For non-compliant LDAP servers, FatWire does not support write operations from the Advanced interface to the LDAP server.
4. "Returns error" means that the operation is not performed (the system returns an error).

Creating a Site

Operation

The Content Server administrator attempts to create a site by using the “Sites” node on the “Admin” tab.

> **Note**
>
> If you are manually creating a site in an LDAP-integrated system, you must ensure that the database and the LDAP server are synchronized. Otherwise, sites will not be properly listed in the Content Server interfaces.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The system creates the site in the Content Server database. The LDAP administrator must create the site entry in the LDAP server.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Flat Schema

1. LDAP server with Read-Only Permissions—The system creates the site in the Content Server database. The administrator must create an entry for all available roles by prefixing *each* site name to the role name in the LDAP server.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Editing a Site

Operation

The Content Server administrator attempts to edit the description of a site by using the “Site Edit” option on the “Admin” tab.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The system modifies the site description in the Content Server database. (The site description is stored only in the Content Server database, even with LDAP integration.)
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Flat Schema

1. LDAP-2 Server with Read-Only Permissions—The system modifies the site description in the Content Server database. (The site description is stored only in the Content Server database, even with LDAP integration.)

2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Deleting a Site

Operation

The Content Server administrator attempts to delete the description of the site by using the “Site Edit” node on the “Admin” tab.

> **Note**
>
> If you are manually deleting a site with LDAP integration, you must ensure that the database and the LDAP server are synchronized. Otherwise, sites will not be properly listed in the Content Server interfaces.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The system deletes the site from the Content Server database. The LDAP administrator must manually delete the site entry from the LDAP server.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Flat Schema

1. LDAP-2 Server with Read-Only Permissions—The system deletes the site from the Content Server database. The LDAP administrator must manually delete the site entry from the LDAP server.
2. LDAP-2 Server with Write permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Granting Users Access to Sites

Operation

The Content Server administrator attempts to grant users access to a site by using the “Site > Users” nodes on the “Admin” tab.

> **Note**
>
> Content Server displays only the roles that are stored in its database.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message because, in the LDAP server, it cannot assign any roles to the specified users of the selected site. The LDAP administrator must manually assign the roles.
2. LDAP-2 Server with Write Permissions—The system assigns roles, within the LDAP server, to the specified users of the selected site.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable.

Flat Schema

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message because, in the LDAP server, it cannot assign any roles to the specified users of the selected site. The LDAP administrator must manually assign the roles.
2. LDAP-2 Server with Write Permissions—The system assigns roles, within the LDAP server, to the specified users of the selected site.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable.

Removing Users' Access to Sites

Operation

The Content Server administrator attempts to remove user access to sites by using the "Site > Users" nodes on the "Admin" tab.

> **Note**
>
> Content Server displays only the sites that are stored in its interface.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The LDAP administrator must manually remove the user's permissions to the sites, directly in the LDAP server.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Behavior is unpredictable.

Flat Schema

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message. The LDAP administrator must manually remove the user's permissions to the sites, directly in the LDAP server.
2. LDAP-2 Server with Write Permissions—The system removes the user's permissions to the sites, directly in the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable.

Creating a Role

Operation

The Content Server administrator attempts to create a role by using the “Roles” node on the “Admin” tab.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The system creates the role in the Content Server database. The LDAP administrator must manually create the role in the LDAP server.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Flat Schema

1. LDAP-2 Server with Read-Only Permissions—The system returns an error message; it cannot create the role in the LDAP server or in the Content Server database. The LDAP administrator must create the role in the LDAP server. The same role must be re-created in the Content Server database.
2. LDAP-2 Server with Write Permissions—The system creates the role in the Content Server database. The system also creates the role for all available sites in the LDAP server by pre-fixing the name of each existing site to the name of the role (*SiteA-Role*, *SiteB-Role*, etc.).

> **Note**
>
> For a role to be created in both the LDAP server *and* the CS database, at least one site must exist in the LDAP server. If no sites exist, the role is created only in the CS database.

3. Not LDAP-2: Write not Supported—Behavior is unpredictable. An error message can be returned.

Editing a Role

Operation

The Content Server administrator attempts to modify the description of a role by using the “Roles” node on the “Admin” tab.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The system modifies the description of the role in the Content Server database. (The role description is stored only in the Content Server database, even with LDAP integration.)
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).

3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Flat Schema

1. LDAP-2 Server with Read-Only Permissions—The system modifies the description of the role in the Content Server database. (The role description is stored only in the Content Server database, even with LDAP integration.)
2. LDAP-2 Server with Read-Only Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Deleting a Role

Operation

The Content Server administrator attempts to delete a role by using the “Roles” node on the “Admin” tab.

System Response

Hierarchical Schema

1. LDAP-2 Server with Read-Only Permissions—The system deletes the role from the Content Server database. The LDAP administrator must manually delete this role from the LDAP server.
2. LDAP-2 Server with Write Permissions—Responds as above (scenario 1).
3. Not LDAP-2: Write not Supported—Responds as above (scenario 1).

Flat Schema

1. LDAP server with Read-Only Permissions—The system returns an error message; it cannot delete the role from either the LDAP server or the Content Server database. The LDAP administrator must create the sites and roles in the LDAP server. The same sites and roles must be re-created in the Content Server database.
2. LDAP-2 Server with Read-Only Permissions—The system deletes the role from both the Content Server database and the LDAP server.
3. Not LDAP-2: Write not Supported—Behavior is unpredictable. The system can return an error message.

Index of Procedures

Index