Content Server Enterprise Edition

Version: 5.5.1

Administrator’s Guide

Document Revision Date: May 5, 2004

FatWire SOFTWARE

CSEE Administrator’s Guide
Document Revision Date: May 5, 2004
Product Version: 5.5.1

FatWire Technical Support
Web: http://www.fatwire.com/Support

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

Table of

Contents

Section 1: Introduction

1 Overview 13

Introduction 14

Logging In to the Content Server Interface 14

The Administrator Tree Tabs 16

Admin 17

Site Admin 18

Workflow 19

Configuring Your CSEE Management System 19

Section 2: Users and Security

2 User Management 23

Overview 24

ACLs 24

Sites and Roles 24

User Profiles 24

User Management on the Delivery System 25

Content Server and User Managers or Directory Servers 25

Granting User Access with the Native Content Server User Manager 25

Granting User Access to CSEE with the LDAP Plug-in 26

Granting User Access to CSEE with the NT 4.0 Plug-in 27

ACLs 27

ACL Permissions 28

Default System ACLs 29

Creating a New ACL 32

Editing an ACL. 33
Deleting an ACL. 34
Assigning ACLs to Tables 34
Assigning ACLs to Content Server Pages (SiteCatalog Page Entries) 35
Setting the ACL Restriction Error Message 36
Users, User Profiles, and User Attributes 36
Default System and Sample Site Users 36
Users and Required ACLs 39
Creating a New User 40
Editing a User 42
Deleting a User Profile 42
Deleting a User 43
User Attributes 43
Roles 44
Default System and Sample Site Roles 45
Creating a Role 48
Editing a Role 48
Deleting a Role 48
Granting Users Access to Sites (Assigning Roles to Users) 49
Viewing Site Users and Changing Role Assignments 50
Deleting Users from a Site 51

3 Setting Up Security 53
Overview 54
ACLs and Security 54
DefaultReader, secure.CatalogManager, and secure.TreeManager 55
BlobServer and Security 55
Security Goals 55
Implementing Security 56
Properties That Configure Security Settings 56
Users and Passwords 57
URLs and the Web Server (Delivery System Only) 58
CSEE Forms and Pages (Delivery System Only) 59
Testing Security 59
Security Tests for All Systems 59
Additional Security Tests for the Delivery System 60

Section 3: Site Configuration

4 Sites, Start Menu, and Tree Tabs 63
Overview 64
Sites 64
Start Menu Items 65

Tree Tabs 66
Summary of Site Configuration 67
Managing CSEE Sites 67
Creating a Site 67
Editing a Site Description 68
Deleting a Site 69
Configuring Asset Types for Sites 69
Enabling an Asset Type for a Site 69
Removing an Asset Type from the Site 70
Creating Start Menu Items for Asset Types 71
Displaying a List of All Start Menu Items 75
Configuring the Tree 76
Creating Tree Tabs 76
Creating the Workflow Groups Tab 78
Editing Tree Tabs 79
Deleting Tree Tabs 79
Changing the Sort Order of the Tree Tabs 79
Configuring the Number of Items Displayed Under Nodes on Tabs 80
Configuring Whether the Tree Is Displayed by Default 81
Configuring Other Asset Type Details 82

5 Workflow 83
Overview 84
Workflow Participants 84
Workflow States 84
Workflow Steps: Moving Assets from State to State 85
Multiple Paths for the Asset 86
Managing Deadlocks 86
Workflow Groups 88
Delegating and Clearing Assignments 88
Placing an Asset in Workflow 89
Restricting Access to Assets While They Are in Workflow 89
Deadlines 90
Scheduling a Deadline Calculation 91
How Does a Workflow Process End? 92
Roles Required to Configure Workflow Processes 92
Planning Your Workflow Processes 93
Start with a Sketch 93
Determine Roles and Participants 93
Determine the E-mail Objects, Actions, and Conditions 95
Determine the States 101
Determine the Steps 102
Determine the Function Privileges 105
Determine Additional Workflow Process Details 108
Sample Site Workflow Processes 109

HelloArticle Process 109
BF: Normal Article Process 111
GE Lighting Process 113
Configuring Your Workflow Processes 116
Overview 116
Set Up E-Mail Objects 117
Set Up the Workflow Actions and Conditions 119
Set Up the Timed Action Event 122
Set Up the States 123
Set Up the Workflow Processes 125
Test Your Workflow Process 134
Moving Your Work 135
Clearing Workflow Assignments 135

6 Sample Sites and Configuration 137
Hello Asset World 138
Users, ACLS, and Roles for Hello Asset World 138
Asset Type Configuration for Hello Asset World 139
Start Menu Configuration for Hello Asset World 140
Tree Tab Configuration for Hello Asset World 141
Workflow for Hello Asset World 141
Burlington Financial 142
Users, ACLS, and Roles for Burlington Financial 142
Asset Type Configuration for Burlington Financial 143
Start Menu Configuration for Burlington Financial 145
Tree Tab Configuration for Burlington Financial 147
Workflow for Burlington Financial 147
GE Lighting 148
Users, ACLS, and Roles for GE Lighting 148
Asset Type Configuration for GE Lighting 150
Start Menu Configuration for GE Lighting 151
Tree Tab Configuration 153
Workflow 154
Burlington Financial Extension (for CS-Engage) 155
Users, ACLS, and Roles for Burlington Financial Extension 155
Asset Type Configuration for Burlington Financial Extension 157
Start Menu Configuration for Burlington Financial Extension 158
Tree Tab Configuration for Burlington Financial Extension 160
Workflow for Burlington Financial Extension 161

Section 4: System Configuration

7 Configuring the User Interfaces 165
Setting the Locale for the Content Server Interface 166
System Default Locale for the CSEE Users 166
System Locale Properties 167
Single-Language Restrictions 168
Locale and Asset Types 168
Locale and the Article Asset Type 169
CS-Desktop 170
Overview 170
Configuring the Word Asset Types for CS-Desktop 172
User Accounts and CS-Desktop 175
Installing the CS-Desktop Client Application 176
Specifying Locale for the Client Application 176
Testing the CS-Desktop Configuration 177
Configuring Word Templates for the Word Assets 178
CS-DocLink 178
Overview 178
Configuring the CS-DocLink Asset Types 179
Users and CS-DocLink 183
Installing the CS-DocLink Client Application 184
Testing the CS-DocLink Configuration 185
InSite Editor 185
Enabling the InSite Editor 186
User Accounts and the InSite Editor 186
eWebEditPro 187
Configuring eWebEditPro 187
User Accounts and eWebEditPro 187
Maintaining Separate Browser Sessions for Preview 187

8 Managing the CSEE Publishing System 189
Overview 190
Delivery Types 191
Publishing Destinations 191
Publishing and the Approval Process 192
The Publishing Schedule (Publish Events) 197
What Happens During the Publishing Session? 198
Obtaining Information About a Publishing Session 198
The Export to Disk Delivery Type 199
How Export to Disk Works 199
Publishing Arguments for Export to Disk 201
File and Directory Naming Conventions 203
Export Starting Points 208
The Mirror to Server Delivery Type 208

How Mirror to Server Works 209
Users and Mirror to Server 211
Publishing Arguments for Mirror to Server 211
CacheManager 211
The Initialize Mirror Destination Feature 212
The Export Assets to XML Delivery Type 213
The XML Output 213
Publishing Arguments for Export Assets to XML 215
File Naming Conventions for Export Assets to XML 215
Configuring the Publishing Process for Your CSEE System 216
Create the Batch User Account 216
Configuring Your System for Export to Disk Publishing 217
Configuring Your System for Mirror to Server Publishing 222
Configuring Your System for Export Assets to XML 229
Additional Publishing Procedures 232
Migrating a Site from One System to Another System 232
Approving Multiple Assets 237
Creating Destinations 239
Editing Destinations 239
Deleting Destinations 240
Creating Export Starting Points 240
Scheduling Publish Events 240
Reading the Schedule Abbreviations 242
Editing Publish Events 243
Overriding the Schedule 243
Assigning Approval or Preview Templates 244
Monitoring Sessions in the Publishing Console 246
Verifying Publishing Readiness 246
Managing Publishing History Information 247
Publishing All Approved Assets 249
Troubleshooting 249
About Publishing System Error Messages 249
Numeric Messages 250
Other Indicators of System or Configuration Issues 253

9 Revision Tracking 257

Overview 258
Tracker Tables and Storage Directories 258
The RTInfo Table 259
Revision Tracking and the Two Asset Models 259
Implicit vs. Explicit Checkin and Checkout 259
Revision Tracking and Non-Asset Tables 260
How Many Versions? 260
Enabling Revision Tracking 261
Enabling Revision Tracking for Asset Types 261

Enabling Revision Tracking for Non-Asset Tables 261
Editing Revision Tracking Settings 262
Deleting Revisions 263
Disabling Revision Tracking 264
Disabling Revision Tracking for Asset Types 264
Disabling Revision Tracking for Non-Asset Tables 264
Unlocking Revisions 265
Clearing Checkouts for Assets 265
Unlocking Versions for Non-Asset Tables. 265
Additional Revision Tracking Functions for Non-Asset Tables. 266
Lock 266
Commit 266
Release 267
Rollback 267
History 267

10 Search Engines 269

Overview 270
The Searching Option 270
Search Engine Properties 270
Asset Types and Search Engines 270
Location of Indexes 272
Configuring Asset Types for Your Search Engine 272
Enabling Asset Types for Search 272
Disabling Asset Types for Search 273
Rebuilding an Index 273

11 Properties and Property Files 275

Using the Property Editor 276
Starting the Property Editor 276
Setting Properties 276
Adding Properties 277
Deleting Properties 278
Property Files by Product 279
Analysis Connector Property Files 280
commercedata.ini 280
databaseloader.ini 284
writequeue.ini 285
Content Server Property Files 285
batch.ini 286
dir.ini 287
futuretense.ini 296
jsprefresh.ini (WebLogic Only) 334
ldap.ini 335
logging.ini 337

satellite.ini 338
xmles.ini 344
CS-Direct Property File 346
futuretense_xcel.ini 346
CS-Direct Advantage Property Files. 362
assetframework.ini 362
catalog.ini 362
gator.ini. 364
transact.ini 367
visitor.ini. 375
CS-Engage Property File 377
CS-Satellite Property Files 377

12 System Information: Content Server Database. 379
Cache Management Tables (Content Server) 380
Approval System Tables (CS-Direct) 380
Publishing System Tables (CS-Direct) 381
Workflow Tables (CS-Direct). 381
Basic Asset Tables (CS-Direct) 382
Flex Family Tables (CS-Direct Advantage) 382
Visitor Tables (CS-Engage) 383
Managing the Attribute Tables. 384
Managing the Session Objects Table (scratch) 384
Deleting Unnecessary .class Files 385

Index 387

Section 1

Introduction

This section provides an overview of all the configuration tasks necessary for your CSEE system.

It contains the following chapter:

- Chapter 1, “Overview”

Chapter 1

Overview

When you are using a Content Server Enterprise Edition (CSEE) content management system, there are two sites:

- The online site that is delivered from your delivery system and visited by readers and/or customers.
- The Content Server content management site that your content providers use to input the data that you use on your delivery system.

This guide is written for the site and system administrators of the content management site. It provides a general overview of the configuration steps that you must take to fully configure a CSEE content management site.

This chapter contains the following sections:

- Introduction
- Logging In to the Content Server Interface
- The Administrator Tree Tabs
- Configuring Your CSEE Management System

Introduction

When you are using CSEE as your content management system, you and the others on your team work with up to four different systems:

- **Development system**, where developers and designers plan and create the online site. They design and create asset types in addition to designing the online site. All of the products that you have purchased are installed on this system (including Commerce Connector or Analysis Connector, if you are using them).
- **Management system**, where content providers such as writers, editors, reviewers, graphic artists, product managers, and marketers develop the content that is delivered from the online site. Revision tracking and workflow functions track changes to the assets (the content), monitoring them until they are approved to be published to the delivery system.

  Only Content Server and the content applications such as CS-Direct, CS-Direct Advantage, CS-Engage are installed on this system. Because this system does not deliver the online site to your visitors, there is no need for Commerce Connector or Analysis Connector to be installed on it.
- **Delivery system**, where the content that you are making available or the products that you are selling are served to your visitors or customers.

  If you are delivering your content dynamically, all of the CSEE products that you purchased are installed on this system. If you are delivering your content statically—that is, if you are serving static HTML pages only—your delivery system is a web server only and you do not have any of your CSEE products installed on that system.
- **Testing system**, where you or your QA engineers test the performance of both the management system and the delivery system.

As an administrator, you spend the majority of your time using the Content Server interface on the management system, adding users and managing the day-to-day use of the system. Before the management system is up and running, however, you and the other administrators will most likely use a development system to pilot and test the configuration of your management system.

Logging In to the Content Server Interface

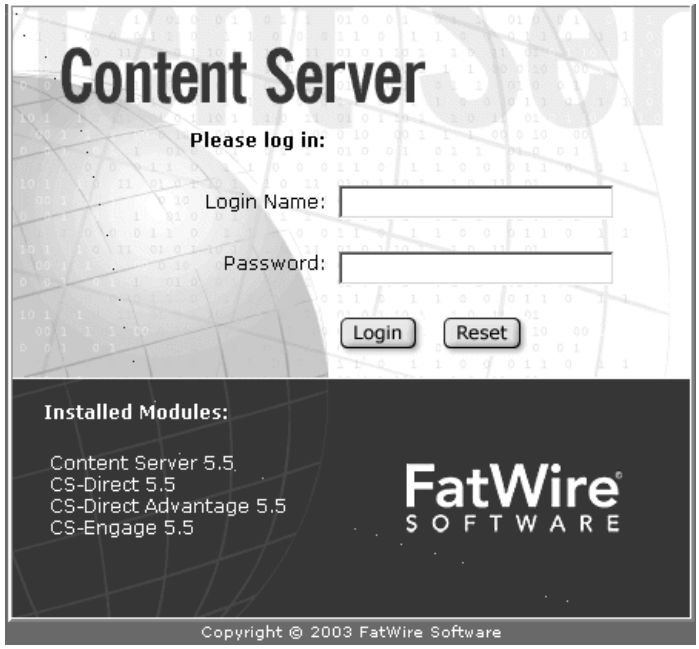
To log in to the Content Server interface, complete the following steps:

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://`*`servername`*`/Xcelerate/LogingPage.html`

   where *`servername`* is the name of the server that hosts the system that you want to log in to. This value might be either the host name or the IP address. Depending on how the system was set up, you might also need to include the port number (`serverX:8080` for example).

Content Server displays the login form:



4. Click in the **Login Name** field and enter your user name.
5. Click in the **Password** field and enter your password.

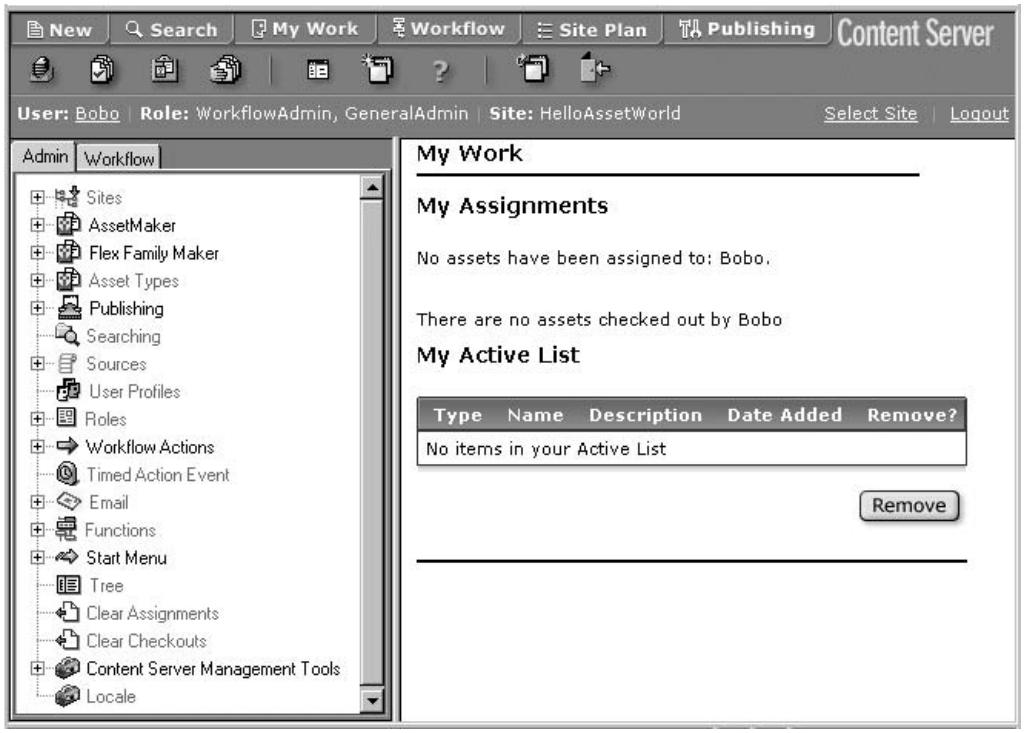
   As an administrator, your user name must have the xceladmin ACL (Access Control List) assigned to it. See Chapter 2, “User Management” for information about user accounts and ACLs.

   If you are the administrator who will configure the CSEE sites for this system for the first time, log in as the default system user named admin (password is xceladmin) and then immediately change the password for that user.
6. Click **Login**.

When you log in to the Content Server interface, you are logging in to a CSEE site. See Chapter 4, “Sites, Start Menu, and Tree Tabs” for a full definition of a CSEE site.

If you have access to more than one site, the first decision that you make after logging in is about which site you want to work on. From that point on, all of the tasks that you complete are completed in the context of that site, until you switch sites, with the exception of the administrative functions located on the **Admin** tab.

The Content Server interface displays three frames in your browser:



- The top frame displays button bars.
- The left frame displays the Tree, a collection of tabs with features, functions, and assets displayed on them.
- The right frame displays the forms that you use to create, edit, delete, copy, and configure the various building blocks, configuration items, and assets that are necessary on your system.

The Administrator Tree Tabs

There are three administrator tabs that can be displayed in the tree in the Content Server interface:

- Admin
- Site Admin
- Workflow

All of the administrator functions described in this book are accessible from one or more of these tabs.

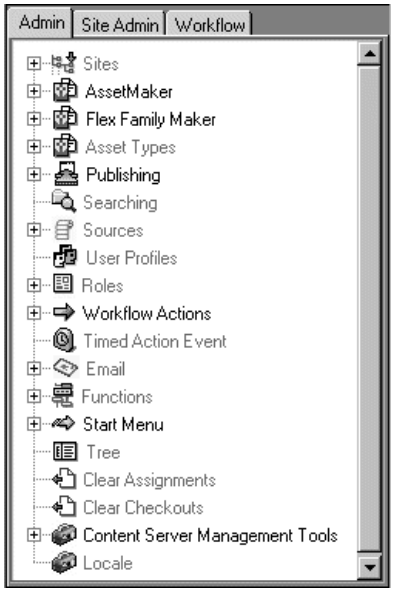
Whether you have access to these tabs depends on the ACLs assigned to your user account and the roles that are assigned to you for the site that you have logged in to. For information about users, ACLs, and roles, see Chapter 2, "User Management."

When you are the person who will configure the CSEE sites on a new CSEE system, you log in as the system default user named admin. It has the ACLs and the roles necessary for you to access the **Admin** tab and begin configuring the system.

When you log in to a system that has no sites created yet, only the **Admin** tab is displayed. You then use the features on that tab to create and configure your sites.

Admin

The **Admin** tab is the main administrative tab. To access this tab, you must have both the **xceladmin** ACL and the **GeneralAdmin** role. (For information, see Chapter 2, "User Management.")



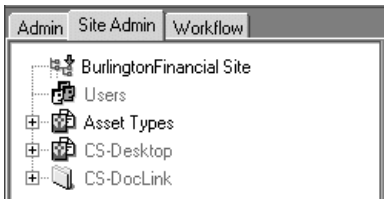
The **Admin** tab provides access to the following functions:

- **Sites**, which you use to create and then configure asset types for sites. See Chapter 4, "Sites, Start Menu, and Tree Tabs" for information. CSEE provides four sample sites as a learning tool for you. For a description of how they are configured, see Chapter 6, "Sample Sites and Configuration."
- **AssetMaker**, which your site developers use to create new asset types that use the basic asset data model. See the *CSEE Developer's Guide* for information.
- **Flex Family Maker**, which site developers use to create new flex families and flex asset types. See the *CSEE Developer's Guide* for information.
- **Asset Types**, which you use to further configure asset types after they are created:
  - 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 9, "Revision Tracking" for information.
  - To enable assets for CS-Desktop. For information, see Chapter 7, "Configuring the User Interfaces."
  - To enable assets for CS-DocLink. For information, see Chapter 7, "Configuring the User Interfaces."
- **Publishing**, which you use to set up the publishing system on your CSEE development and management systems. See Chapter 8, "Managing the CSEE Publishing System" for information.

- **Searching**, which you use to configure asset types so that they are indexed correctly when you are using one of the supported third-party search engine modules. See Chapter 10, "Search Engines" for information.
- **Sources**, which you or your site developers use to add new sources if your asset types are designed to use them. See the *CSEE Developer's Guide* for information.
- **User Profiles**, which you use to specify e-mail addresses for your users. See Chapter 2, "User Management" for information.
- **Roles**, which you use both for determining the recipients of workflow assignments and to control your users' access to functions in the Content Server interface.
- **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 5, "Workflow" for information.
- **Timed Action Event,** which you use to configure how frequently the due dates for assets that are in workflow are calculated. See Chapter 5, "Workflow" for information.
- **Email**, which you use to create the workflow e-mail messages that can be used by any workflow process. See Chapter 5, "Workflow" for information.
- **Functions**, which your developers or FatWire professional services staff use to add customized functions to the Content Server interface.
- **Start Menu**, which you use to provide links from the form that is displayed when you click the **New** button in the Content Server interface and the links from the form that is displayed when you click the **Search** button. You use them to create new assets or search for existing ones. See Chapter 4, "Sites, Start Menu, and Tree Tabs" for information.
- **Tree**, which you use to control access to the existing tree tabs or to create new ones. See Chapter 4, "Sites, Start Menu, and Tree Tabs" for information.
- **Clear Assignments**, which you use to clear workflow assignments. See Chapter 5, "Workflow" for information.
- **Clear Checkouts**, which you use to unlock and check back in the assets that are checked out by specific users. See Chapter 9, "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 2, "User Management" and Chapter 9, "Revision Tracking."
- **Locale**, which you use to set the default locale for the CSEE system. See Chapter 7, "Configuring the User Interfaces" for information.

Site Admin

The **Site Admin** tab provides access to site configuration functions for the site that you are currently logged in to:

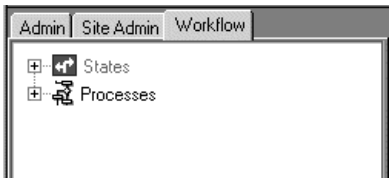


To access this tab, you must have the **SiteAdmin** role for the site that you are logged in to.

Because the site configuration functions are presented on this separate tab (in addition to the **Admin** tab), you can configure your management system so that the person who assigns the roles to new users or enables asset types for a specific, individual site is not the same person who adds users to the system and configures system wide options such as publishing.

Workflow

You use the options on the **Workflow** tab to create workflow processes from the workflow building blocks that were created on the **Admin** tab.



To access this tab, you must have the **WorkflowAdmin** role for the site that you are logged in to.

Configuring Your CSEE Management System

Because Content Server sites address both design issues and access issues, you and the other administrators work with the developers, system administrators, and managers to determine how to use CSEE sites and how many sites you need for your system. Before you begin, you meet with the team and determine the following kinds of things:

- How many users and ACLs (access control lists) do you need? (Remember that you may need to create ACLs to assign to the visitors of the online site, as well.)
- How many site roles you do you need?
- Which asset types need a workflow process?
- Which asset types should use revision tracking?
- Who should have access to which asset types on which sites?

Use both this book and the *CSEE Developer's Guide* to help you make these decisions. The *CSEE Developer's Guide* provides an overview of the development process for implementing your entire CSEE system, of which configuring the management system is only one part.

This section provides an outline of the configuration steps that you complete when you first configure your CSEE management system, presented in one possible order. The tasks introduced in this outline are discussed in detail in this guide.

1. Create the users for the site. See Chapter 2, “User Management.”
2. Create the roles that the users will fulfill for the site. See Chapter 2, “User Management.”
3. Create the site. See Chapter 4, “Sites, Start Menu, and Tree Tabs.”

4. After your developers design the asset types, help them move those asset types from the development system to the management system. See "Troubleshooting" on page 249.
5. (Optional) If you plan to use workflow, create workflow processes for the asset types that will use workflow. Chapter 5, "Workflow."
6. Enable and configure the asset types that will be used on the site. This includes creating start menu items so the asset types are accessible and, perhaps, creating tree tabs that also provide access to the asset types. See "Creating Start Menu Items for Asset Types" on page 71 and "Configuring the Tree" on page 76.
7. Enable the appropriate users for the site by assigning roles to them. See Chapter 2, "User Management."
8. Configure the publishing system so the information that your content providers create can be made available on the delivery system. See Chapter 8, "Managing the CSEE Publishing System."
9. (Optional) If you have more than one language pack installed on your CSEE management system, configure the default locale for the system. See Chapter 7, "Configuring the User Interfaces."
10. (Optional) If you plan to use revision tracking, enable revision tracking for the appropriate asset types. See Chapter 9, "Revision Tracking."
11. (Optional) If you purchased one of the supported third-party search engine modules and you are using any basic asset types, configure your asset types so that your assets are indexed correctly and can be found by your search engine. See Chapter 10, "Search Engines."
12. (Optional) If you plan to use any of the additional user interfaces such as InSite Editor, Ektron's eWebEditPro, CS-Desktop, or CS-DocLink, configure those user interfaces. See Chapter 7, "Configuring the User Interfaces."

Section 2

Users and Security

This section describes how to create and manage users and how to configure security for your CSEE system.

It contains the following chapters:

- Chapter 2, “User Management”
- Chapter 3, “Setting Up Security”

Chapter 2

User Management

There are several user management components that control access to a CSEE system: ACLs (access control lists), user accounts, user profiles, roles, and sites.

Some of these components 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 *CSEE Developer’s Guide*.

This chapter contains the following sections:

- Overview
- ACLs
- Users, User Profiles, and User Attributes
- Roles

In this chapter, sites are discussed in terms of users. For information about creating and configuring CSEE sites, see Chapter 4, “Sites, Start Menu, and Tree Tabs.”

Overview

The components that provide information about a user's access privileges on a CSEE system are ACLs (access control lists), user accounts, user profiles, roles, and sites.

ACLs

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

Sites and Roles

CS-Direct introduces the concept of a CSEE **site**.

A CSEE site is an object that the site developers use as a design aid or organizational construct for the online site that you are delivering from your CSEE delivery system and that you (the administrators) use as an access control tool on the management system.

You use sites to control or restrict the content providers who have access to the assets and asset types on the management system.

Site-specific access rights are managed through **roles**. Roles control which assets a user can create, which tabs are displayed in the tree for that user, whether a user is eligible for a workflow step, and so on.

Roles describe the function of an individual on a site.

User Profiles

User profiles are typically used to support workflow actions by mapping a user name to an e-mail address. However, if your management system has been customized, user profiles might hold additional information (user attributes) to support other actions or events.

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 have access to areas of the site, you create the ACLs that you need 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 *CSEE 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.

Content Server and User Managers or Directory Servers

Content Server has a Directory Services API that enables your CSEE system to connect to external directory servers or user managers that contain authentication information, user information, and so on.

There are three user manager or directory server plug-ins available with Content Server, one of which was configured when your CSEE systems were installed:

- The Content Server native user manager, which uses the native Content Server user management tables; that is, the `SystemUsers` and `SystemUserAttrs` tables.
- The LDAP plug-in. When you use this option, user names and attributes are stored in your directory server rather than in the Content Server database.
- The NT 4.0 authentication plug-in, which uses NT 4.0 to authenticate users but stores the user information in the Content Server user management tables.

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

Installing and configuring the plug-ins is documented in the *Installing the CSEE Content Applications*. The properties that configure the plug-ins are in the `futuretense.ini` (the **authentication** tab), `ldap.ini`, and the `dir.ini` files and they are all described in Chapter 11, "Properties and Property Files"in this guide.

Granting User Access with the Native Content Server User Manager

When you are using the native Content Server user manager, this is the general process that you follow to grant users access to your CSEE management system:

1. Examine the list of default ACLs to determine if you need to add more for your system. Under no circumstances should you modify or delete an existing default ACL.
2. If you need additional ACLs, create them and then assign them to the appropriate database tables and 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 functions. For help with this step, see the procedures in "ACLs" on page 27.

3. Create users and assign the appropriate ACLs to them. For help with this step, see the procedures in "Users, User Profiles, and User Attributes" on page 36.
4. If you plan to use a workflow process that sends e-mail messages to participants, create user profiles for all the users. For help with this step, see "Creating a User Profile" on page 41. For information about workflow, see Chapter 5, "Workflow."
5. If you want to store any user attributes for your users in addition to e-mail addresses, create them. For help with this step, see "User Attributes" on page 43.
6. Create your CSEE sites. For help with this step, see Chapter 4, "Sites, Start Menu, and Tree Tabs."
7. Create the roles that you need to implement the access control plan designed for your CSEE management system. For help with this step, see "Roles" on page 44.
8. For each site in your system, enable the users that should have access to those sites by assigning the appropriate roles to the users for those sites. For help with this step, see "Granting Users Access to Sites (Assigning Roles to Users)" on page 49.

Granting User Access to CSEE with the LDAP Plug-in

When 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 Content Server interface. However, you must still use the Content Server interface to create and assign roles to those users.

In general terms, this is how you grant users access to your CSEE management system when you are using LDAP:

1. Create LDAP user groups whose names exactly match the Content Server system default ACLs. For a list of the default system ACLs, see "Default System ACLs" on page 29.
2. Use your LDAP interface to create users and assign users to those groups.

   Content Server then assumes that a user who is a member of such an LDAP group has the Content Server ACL with the corresponding name.

   For example, a user who is a member of the xceladmin LDAP group has all the access rights granted by the xceladmin ACL.
3. Be sure that your LDAP system stores an e-mail attribute for your users. CS-Direct workflow processes typically have steps in which messages are e-mailed to the workflow participants and CS-Direct queries your LDAP directory for e-mail addresses when you are using the LDAP plugin.
4. If you need additional sets of database permissions, create new ACLs by using the Content Server interface and remember to create the matching LDAP user groups and assign the users to the appropriate groups.
5. Create the appropriate roles in the Content Server interface. For help with this step, see "Roles" on page 44.
6. For each site in your system, enable the users that should have access to those sites by assigning the appropriate roles to the appropriate users for each site. For help with this step, see "Granting Users Access to Sites (Assigning Roles to Users)" on page 49.

Granting User Access to CSEE with the NT 4.0 Plug-in

When you are using NT to authenticate your users on either the management or the delivery system, you create user accounts with NT rather than with the Content Server interface. However, you must still use the Content Server interface to create user profiles and roles and to assign roles to those users.

In general terms, this is how you grant users access to your CSEE management system when you are using NT authentication:

1. Create NT user groups that exactly match the Content Server system default ACLs. For a list of the default system ACLs, see "Default System ACLs" on page 29.
2. Use your NT interface to create users and assign users to those groups.

   Content Server then assumes that a user who is a member of such an NT group has the Content Server ACL with the corresponding name.

   For example, a user who is a member of the xceladmin NT group has all the access rights from the xceladmin ACL.
3. If you plan to use a workflow process that sends e-mail messages to participants, create user profiles for all the users. For help with this step, see "Creating a User Profile" on page 41. For information about workflow, see Chapter 5, "Workflow."
4. If you need additional sets of database permissions, create new ACLs by using the Content Server interface and remember to create the matching NT user group and assign the group to the appropriate users.
5. Create the appropriate roles in the Content Server interface. For help with this step, see "Roles" on page 44.
6. For each site in your system, enable the users that should have access to it by assigning the appropriate roles to the appropriate users. For help with this step, see "Granting Users Access to Sites (Assigning Roles to Users)" on page 49.

ACLs

ACLs (access control lists) serve as the foundation of the security and user management model in your CSEE 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 CSEE 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"` on page 296. For information about configuring security on your CSEE system, see Chapter 3, "Setting Up Security."

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

User Names

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

While users have one user account and one set of ACLs, no matter how many CSEE 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 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. Their access rights to that table (read, write, create, and so on) are those that are defined by the ACL.

All of the Content Server system tables and several of the CSEE 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 on 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 Tables" on page 34
- About registering foreign tables, see the "Database" chapter in the *CSEE Developer's Guide*

Page Entries in the SiteCatalog Table

The `SiteCatalog` table holds page entries for all the pages displayed for the CSEE 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.

ACL Permissions

An ACL describes a set of permissions. When the ACL is assigned to a database table, only the operations described by the permissions in the ACL are permitted to be performed on the database table. The following table defines all ACL permissions.

Table 1: ACL Permissions

Permission	Bit Mask	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 *CSEE 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

The bit mask numbers for each permission assigned to an ACL are added together and the total is listed with the ACLs in the `SystemACL` table.

To determine which permissions are granted to an ACL, use the **ACL** node under **CS Management Tools** on the **Admin** tab in the Content Server interface.

Default System ACLs

Content Server and the CSEE content applications use several default ACLs to control user access to their features and functions. The following table lists all of the default system ACLs and their permissions:

Table 2: Default System ACLs and Their Permissions

ACL Name	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

ACL Name	Read	Retrieve	Write	Create	Delete	Rev. Track Audit	Rev. Track Admin
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
xceladmin	X	X	X	X	X	X	X
xceleditor	X	X	X	X	X	X	

Caution

Never modify the set of permissions for a default system ACL. Additionally, never modify the ACLs assigned to any of the system tables.

Each default 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, they are all necessary because they are assigned to different tables.

This table describes how each of the default ACLs is used by Content Server and the CSEE content applications.

Table 3: Default System ACL Usage

ACL Name	How CSEE Uses It
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.

ACL Name	How CSEE Uses It
ElementEditor	Allows users to write data to the `ElementCatalog` and `SystemSQL` tables. Site designers and anyone who creates template, CSElement, and SiteEntry assets need this ACL.
ElementReader	Allows users to read data in the `ElementCatalog` and `SystemSQL` tables. CS-Direct 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. CS-Direct users need this ACL so they can inspect the templates assigned to their assets.
RemoteClient	Grants users the ability to log in to the CSEE 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.
UserReader	Allows users to read information about user accounts, but not to change that information. CS-Direct users need this ACL so they can use workflow.
Visitor	Grants the holder of the ACL the ability to write data to the CS-Engage tables that hold visitor data. Any visitor whose data you are collecting on the delivery system must have this ACL assigned to their user account.
VisitorAdmin	Grants users the ability to create visitor attributes, history attributes, history types, and recommendations. Any CS-Engage user who needs to create any of those asset types needs this ACL.

ACL Name	How CSEE Uses It
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 on 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 CSEE content applications. The login request code verifies whether or not a user has the ACL. All users of the management system need this ACL.

If your user management needs on either the management or delivery system require different or additional ACLs, create them using the procedures in this section.

Caution

Under no circumstances should you modify a default system ACL.

Although creating and applying ACLs is an administrative task, you must work with your site designers and developers to determine which ACLs you need and how to apply them. For example:

- If the public, 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 may need to create additional ACLs or roles to support the new functions and tabs.

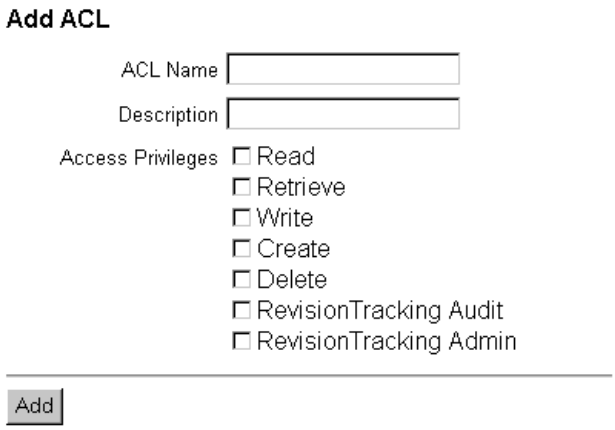
The *CSEE Developer's Guide* describes user management on the delivery system.

Creating a New ACL

To create a new ACL, complete the following steps:

1. On the **Admin** tab, expand the **Content Server Management Tools** tree and then double-click on **ACLs**.
2. In the **ACL Select Operation** form, select **Add ACL** and click **OK.**

   The **Add ACL** form appears.



3. In the **ACL Name** field, enter a unique name.
4. Select each of the access privileges that you want to assign to this ACL. (For information about the permissions listed on the form, see “ACL Permissions” on page 28.).
5. Click the **Add** button.

   Content Server creates the ACL, writing it to the `SystemACL` table.

The ACL that you created now appears in the drop-down list of ACLs in the **ACL Select Operation** form.

If you are using one of the user manager plug-ins (LDAP or NT), be sure to use your LDAP or NT software to create a group that exactly matches the ACL that you created and then assign that group to the appropriate users.

Editing an ACL

Caution

Never modify any of the default ACLs. For a list of these ACLs, see “Default System ACLs” on page 29.

To edit an ACL, complete the following steps:

1. On the **Admin** tab, expand the **Content Server Management Tools** tree and then double-click on **ACLs.**
2. In the **ACL Select Operation** form, from the drop-down list, select the name of the ACL that you want to edit.
3. Select the **Modify ACL** option and click OK.
4. Adjust the set of permissions configured for the ACL. (For information about the permissions listed on the form, see “ACL Permissions” on page 28.)
5. Click **Modify**.

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

Deleting an ACL

> **Caution**
>
> Never delete any of the default ACLs. For a list of these ACLs, see "Default System ACLs" on page 29.

To delete an existing ACL:

1. On the **Admin** tab, expand the **Content Server Management Tools** tree and then double-click on **ACLs.**
2. From the **ACL Select Operation** form, from the drop-down list, select the ACL that you want to delete.
3. Click **OK.**

   A message appears that asks you to verify that you intend to delete the ACL.
4. Click **OK** to delete the ACL.

Assigning ACLs to Tables

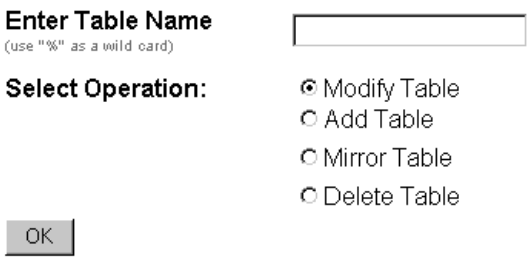
If you or the site designers create new tables, you might need to restrict access to those tables by assigning ACLs to them. Typically, you assign ACLs to new tables when you create those tables. For information, see the *CSEE Developer's Guide.*

Do not assign additional ACLs to the system tables or to any of the core product tables installed by the CSEE content applications.

To apply ACLs to an existing table, complete the following steps:

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

   The **Enter Table Name** form appears.



2. Enter the name of the table that you want to assign ACLs to. You can use the percentage character (%) as a wildcard to obtain a list of all tables or you can enter a partial name, ending with the wildcard character, for a set of tables that match the partial name.
3. Select **Modify Table** and click **OK**.

   The **Modify Table Select** form appears with a list of tables that match the name you entered. It might be a long list with a vertical scroll bar.
4. Click on a table to select it.

The **Modify Catalog** form appears.

Do not change the value in the **File Storage Directory** field. For information about this field, see the description of `defdir` in the chapter about the Content Server database in the *CSEE Developer's Guide*.

5. From the drop-down list, select the ACLs that you want to apply to this table. To select more than one, use the shift key.
6. Click **Modify.**

Assigning ACLs to Content Server Pages (SiteCatalog Page Entries)

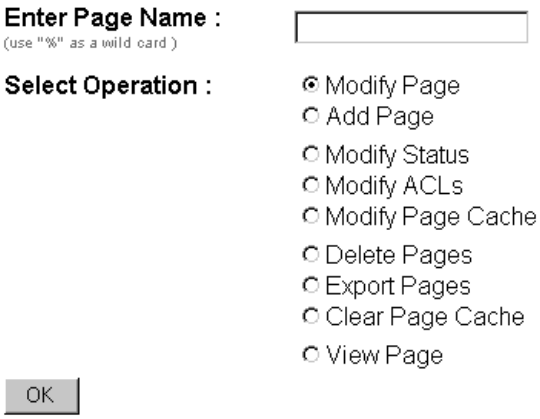
There are at least two ways to assign ACLs to `SiteCatalog` page entries.

- 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 apply ACLs to a page entry that is not associated with a SiteEntry or template asset, complete the following steps:

1. Open the **Content Server Management Tools** item.
2. Double-click the **Site** option.

   The **Page Operation** screen appears in the work area.



3. Enter the path name of the page entry that you want to assign ACLs to. You can use the wildcard character by itself to retrieve all the page entries in the `SiteCatalog` table, or with a partial name, ending with the wildcard character, to narrow the list that is returned.
4. Select **Modify ACLs** and click **OK**.

   The **Modify ACLs** form appears with a list of pages that match the name you entered. It can be a long list with a vertical scroll bar.
5. From the list, select the ACL that you want to assign. You can select more than one.

6. Scroll through the list of pages. To assign the selected ACLs to a page, select **Apply** (to the right of the **Page Name)**. Any ACLs currently assigned to the page are shown in the **ACL** column. You can use the **All** button to select all the pages shown. You can also use the **None** button to clear all selections.
7. Click **Apply**.

Setting the ACL Restriction Error Message

When users attempt to access a page but their ACLs do not give them the correct privileges, the system displays a message. The source of that message is located here:

`WebServerInstallDir/doc/futuretense_cs/formpriv.html`

You can customize this page and the other HTML error message pages in this directory, but with the following restrictions:

- Do not change the file name of any of the files.
- Do not alter the `{0}` string anywhere in any of the files.

  Content Server uses the `{0}`string to automatically generate the error messages.

Users, User Profiles, and User Attributes

When you are using the native Content Server user manager plug-in, you use the Content Server **users** feature to assign access rights to individual users. An ACL is a named set of database permissions; a user account is a named set of ACLs.

A CS-Direct **user profile** holds a set of **user attributes**. By default, the only user attributes a user profile holds is the user's e-mail address and their locale preference:

- The **email attribute** is used to support workflow actions. You can create workflow actions that send workflow participants e-mail about the assets that are assigned to them.
- The **locale attribute** is used to determine which language to use when your CSEE system has a language pack installed.

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 interface 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 *CSEE Developer's Guide*.

Default System and Sample Site Users

There are several default users installed by Content Server and the CSEE content applications. Some are required for the applications to function and others are supplied for the sample sites.

The following table lists the default system users and their ACLs:

Table 4: Defaults System Users and Their ACL Assignments

User Name	ACLs	Description
admin	TableEditor, Visitor, VisitorAdmin, UserEditor, UserReader, xceladmin, Browser, xceleditor, PageEditor, ElementEditor, RemoteClient	The basic administrator user that CS-Direct creates so that you can begin configuring your CSEE content applications. Do **not** delete this user unless another user with an identical ACL assignment already exists.
Content Server (the installation user account)	SiteGod, Browser, ContentEditor, ElementReader, ElementEditor, PageReader, PageEditor, UserReader, UserEditor, TableEditor	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.

The following table lists the default sample site users and their ACLs:

Table 5: Defaults Sample Site Users and Their ACL Assignments

User Name	ACLs	Description
Bobo	TableEditor, Visitor, VisitorAdmin, UserEditor, UserReader, xceladmin, Browser, xceleditor, PageEditor, ElementEditor, RemoteClient	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.

User Name	ACLs	Description
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 CS-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.
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 Name	ACLs	Description
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 CS-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 CS-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.

Note that most of these users have the same set of ACLs. It is their roles for the sites that gives them access to specific functions on those sites through workflow and tree tabs.

If you do not install the sample sites, your system has the following default users only:

- admin
- *Content Server*, the user account created for the installation
- DefaultReader

Users and Required ACLs

This section lists the ACLs that must be assigned to a user for that user to perform specific functions within the Content Server interface.

Note

If you are using the LDAP or NT user manager plug-ins, substitute the word “group” for the word “ACL” and create users who belong to the groups with these names.

Basic ACLs

Every CSEE content application user must have at least the following ACLs assigned to their user accounts:

- Browser
- ElementReader
- PageReader
- UserReader

- xceleditor

Additional ACLs

In addition to the basic ACLs, some users may require the following ACLs:

- Workflow administrators and site administrators: xceladmin
- Administrator users:
  - TableEditor
  - UserEditor
  - VisitorAdmin (if you have CS-Engage installed)
  - xceladmin
- Site designer users:
  - ElementEditor
  - PageEditor
  - TableEditor
  - Visitor (if you have CS-Engage installed)
  - Visitor Admin (if you have CS-Engage installed)
  - xceladmin
- CS-Engage users:
  - Visitor
- CS-Desktop, CS-DocLink, and InSite Editor users:
  - RemoteClient
  - Visitor (if your installation uses CS-Engage)

Creating a New User

To create a new user, you must complete two procedures (which are described in the following sections):

- Create the user
- Create the user profile for that user, if you are using the CSEE content applications.

To give the new user access to your CSEE sites, you enable that user for the appropriate sites by assigning roles to that user name for each site the user will use. For information about enabling users after you have created them, see "Granting Users Access to Sites (Assigning Roles to Users)" on page 49.

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 CSEE applications on the management system:

  `http://servername/Xcelerate/LoginPage.html`

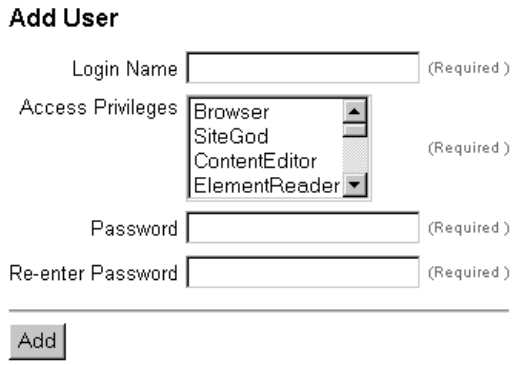
  where *`servername`* is the name of the server that hosts the system that you want to log in to. This value might be either the host name or the IP address. Depending on how the system was set up, you might also need to include the port number (`serverX:8080` for example).

Creating the User

To create a new user, complete the following steps:

1. On the **Admin** tab, expand **Content Server Management Tools** and then double-click on **Users**.
2. In the **Select Operation** form, select **Add User** and click **OK**.

   The **Add User** form appears.



3. Enter a unique name in the **Login Name** field. Do not include spaces or special characters, such as punctuation. The underscore character (_) is allowed.
4. Select the appropriate ACLs for the new user from the **Access Privileges** list. You can use the Shift key to select a group and the Ctrl key to select additional roles.

   For information about which ACLs a user needs, see “Users and Required ACLs” on page 39.
5. Enter a password in the **Password** and **Re-Enter Password** fields.
6. Click **Add**.

If you are using the CSEE content applications, next you create the user profile for the user.

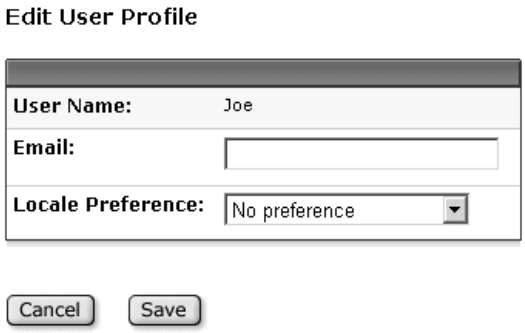
Creating a User Profile

To create or edit a user profile for a user, complete the following steps:

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

   The **User Profile Management** form appears.
2. Enter the name of a user and click **Select**.

   The **User Profile Management** form appears.
3. Click **Edit**.
4. The **Edit User Profile** form appears.



5. Enter an e-mail address.
6. (Optional) If your CSEE system has any language packs installed, select a locale preference for this user.
7. Click **Save**.

Next, enable this user for the sites he or she needs to work with. See "Granting Users Access to Sites (Assigning Roles to Users)" on page 49.

Editing a User

Caution

Do not change the ACLs or the names of any of the default system users: DefaultReader, *Content Server,* or xceladmin.

To edit a user, complete the following steps:

1. On the **Admin** tab, expand the **Content Server Management Tools** tree and then double-click on **Users**.
2. In the **Select Operation** form, enter the name of the user account that you want to edit. You can use the percentage (%) character as a wildcard if you do not know the exact name of the user account.
3. Select **Modify User** and then click **OK**.

   The **Modify User Select** form appears, listing the user name. If you used a wildcard, a set of users that match the string that you entered appear in the list.
4. Click on the user name of the account that you want to edit.

   The **Modify User** form appears.
5. Change the user as needed and then click **Modify**.

Deleting a User Profile

When you want to delete a user, you should delete the user profile first.

To delete a user profile, complete the following steps:

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

   The **User Profile Management** form appears.

2. Enter the name of a user and then click **Select**.
3. In the **User Profile** form, click **Delete**

   The system displays a confirmation message.
4. Click **OK**.

Deleting a User

Caution

Do **not** delete any of the default system users: admin, *Content Server*, or DefaultReader.

Before you delete a user, you should delete the user profile as described in "Deleting a User Profile" on page 42.

To delete a user account:

1. On the **Admin** tab, expand the **Content Server Management Tools** tree and then double-click on **Users**.
2. In the **Select Operation** form, enter the name of the user account that you want to delete.
3. Select **Delete User** and then click **OK**.

   The **Delete User** form appears, listing any user whose name matches the string that you entered in the previous step.
4. Select **Delete** for each user account that you want to delete.
5. Click **Delete**.

   The system displays a confirmation message.
6. Click **OK**.

User Attributes

By default, the only user attributes that the CSEE content applications need are an e-mail address and locale preference. You use the user profile feature to assign these attributes to a user.

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 works exactly like the **Modify User** option except that it changes only the user attributes.

To modify a user's attributes:

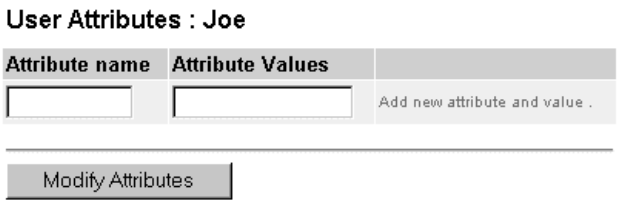
1. On the **Admin** tab, expand the **Content Server Management Tools** tree and then double-click on **Users**.
2. In the **Select Operation** screen, either leave the user name field blank or enter a specific user name. (If you leave the user name field blank, Content Server retrieves all users.)

3. Select **Modify User Attributes** and then click **OK**.

   The **Modify User Select** form appears, listing either all the current users or the one user whose name you entered.

4. Select the user name that you want to modify.

   The **User Attributes** form appears:



5. Make any changes as required:
   - Change the current value (or values) assigned to the attribute by editing the **Attribute Value**s field next to the **Attribute Name**.
   - Delete any unwanted attributes by clearing the associated value field (that is, make it blank).
   - Add an entirely new attribute by entering its name and at least one value in the boxes at the bottom of the display.
6. Click **Modify Attributes**.

Roles

When you are using the CSEE content applications, user management includes the concept of roles. Roles complement users in the following ways:

- The user definition describes an individual's access to the underlying CSEE functionality through ACLs. Roles represent a job description or title of individuals in an organization who have similar functions: content provider, editor, site designer, and administrator, for example.
- A CSEE user has one user definition (account) no matter how many CSEE sites you are using. However, the roles that user has 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.

Users must be assigned all the ACLs that will allow them to fulfill all of their site-specific roles. For example, if you create a role that allows a user to create template assets, the user assigned that role must be assigned the ElementEditor ACL because creating templates writes data to the `ElementCatalog` table.

For information about users and ACLs, see "Users and Required ACLs" on page 39.

Default System and Sample Site Roles

There are several default roles installed by the CSEE content applications. One is required for the applications to function and the rest are supplied for the sample sites.

The sample sites are configured to allow access to tabs in the tree based on the roles assigned to users. Use the sample roles as examples of how you can configure access control on your CSEE sites.

The following table lists the default system roles:

Table 6: Default 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 on 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 on the **Admin** tab. Use the Site Admin role if you want certain users to manage the users who can access a site, but you do not want them to be able to create new users. 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.

The following table lists the default sample site roles:

Table 7: Default Sample Site Roles

Role	Description
Analyst	Used by the Burlington Financial with CS-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 CS-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.**

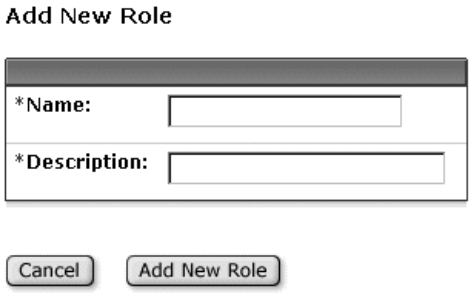
Role	Description
Editor	Allows users to do the following: • Burlington Financial – edit articles. • GE Lighting – review products and product prices. • Burlington Financial with CS-Engage extensions—review funds and segments. Grants access to the following tree tabs in the sample sites: **Active List**, **Content**, **Marketing**, **Product**, **Site Plan.**
Expert	Used by the Burlington Financial with CS-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 CS-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.**

Creating a Role

To create a new role:

1. On the **Admin** tab, double-click the **Roles** option.
2. Double-click the **Add New** option in the **Roles** item.

   The **Add Role** form appears:



3. Click in the **Name** field and enter a unique name of up to 32 characters.
4. Click in the **Description** field and enter a descriptive phrase using up to 255 characters for the role.
5. Click **Save**.

After you create a role, be sure to edit the default tree tabs (**Workflow**, **Site Admin**, **Admin**, **Site Plan**, and **Active List**) and add your new roles to the appropriate tabs.

For more information, see "Tree Tabs" on page 66.

Editing a Role

Although you cannot change the name of a role after you have created it, you can edit the description. To edit the description of a role, complete the following steps:

1. On the **Admin** tab, double-click the **Roles** option.
2. In the list of roles, double-click the name of the role that you want to edit.
3. In the **Role** form, click **Edit**.

   The **Edit Role** screen appears.
4. Click in the **Description** field and enter the new description.
5. Click **Save**.

Deleting a Role

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

To delete a role, complete the following steps:

1. On the **Admin** tab, double-click the **Roles** option.

2. In the list of roles, double-click the name of the that role you want to delete.
3. In the **Role** form, click **Delete**.

   The system displays a confirmation message.
4. Click **OK**.

Granting Users Access to Sites (Assigning Roles to Users)

In order for users to have access to a CSEE site, they must have roles assigned to them for that site. The roles assigned to a user at a site associate the user with the site and describe the user's place in the workflow processes created for that site.

To give a user access to a site, complete the following steps:

1. On the **Admin** tab, expand the **Sites** option.
2. Expand the site that you want the user to have access to.
3. Double-click the **Users** option for that site.

   The **User Role Management** form appears.

4. Enter the user name and click **Select**.

   The **User Role Management** form displays the user name with no assigned roles.
5. Select the edit icon next to the user name.

   The **Edit Roles** form appears:

6. In the **Roles** list box, select a role or any combination of roles from the list.
7. Click **Save**.

Remember that the user cannot perform the functions of this role unless the user has adequate ACL permissions.

Viewing Site Users and Changing Role Assignments

To see a list of users and roles currently defined for the site, and to make changes in any user role assignments, complete these steps:

1. On the **Admin** tab, expand the **Sites** option, and then expand the site.
2. Double-click the **Users** option for that site.
3. In the **User Role Management** form, click **Select** with the user name field blank.
4. The system displays a list of all the users currently assigned roles at that site:

5. If you want to make any changes, click the edit icon next to the user name.
6. In the **Edit Roles** form for the user, select a role or any combination of roles from the list and click the **Save** button.
7. Repeat the previous two steps as needed for other user changes.

Deleting Users from a Site

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

Complete the following steps:

1. On the **Admin** tab, expand the **Sites** option, and then expand the site.
2. Double-click the **Users** option for that site.
3. In the **User Role Management** form, enter the user's name and click **Select**.
4. In the next form, click the delete icon (the trash can) next to the user's name.

   The system displays a confirmation message.
5. Click **Delete User from Site**.

Chapter 3

Setting Up Security

Determining what your security protocols should be and then implementing them is an important part of a CSEE 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:

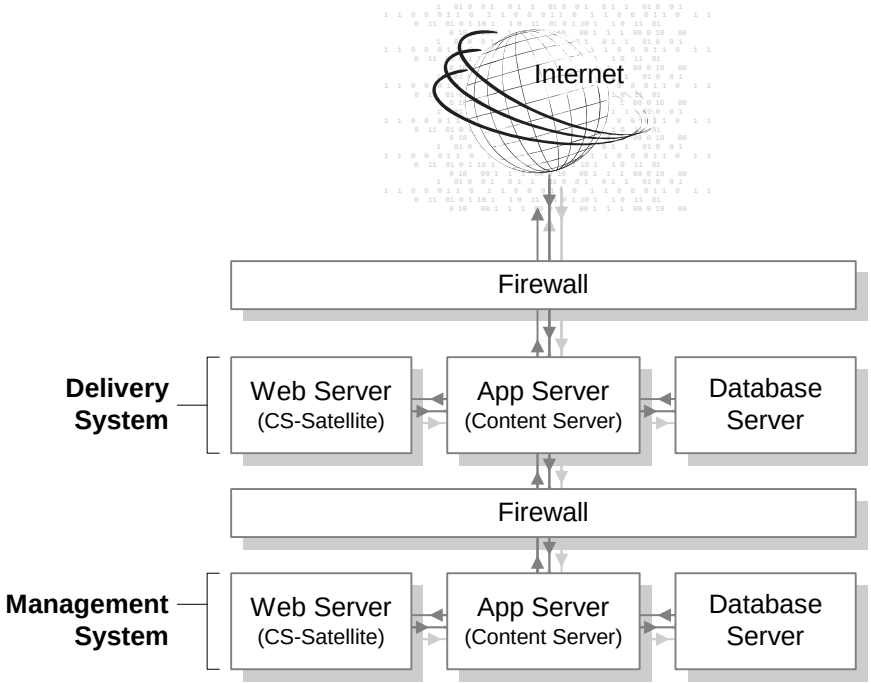
- Overview
- Implementing Security
- Testing Security

Overview

Before the developers begin designing the online site that you plan to deliver or contemplate making 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.

Additionally, at regular intervals, you should review your systems to determine whether things are working as they should.

The following figure shows an example of a secure CSEE system:



ACLs and Security

As mentioned in Chapter 2, "User Management," ACLs (access control lists) serve as the foundation of the security and user management model in your CSEE 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 Chapter 2, "User Management"), 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 CSEE content application table.

DefaultReader, secure.CatalogManager, and secure.TreeManager

Content Server and the CSEE 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 *CSEE Developer's Guide*.

Security Goals

Typically, you have a different set of security goals for each of your CSEE 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 CSEE 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 2, "User Management."

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 interface to prevent anyone from adding assets or code through CS-Direct 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`
- `es.security`

Note

The `es.security` property is present for backward compatibility. When it is set to `true`, any requests that are sent to the EvalServer servlet are immediately routed to the ContentServer servlet, which then checks the ACLs of the user making the request.

Always set `es.security` to `true`.

- `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 "CSEE Forms and Pages (Delivery System Only)" on page 59.

Users and Passwords

Be sure that the default user accounts were made secure after CSEE 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 admin user account (**Users** node under **Content Server Management Tools** on the **Admin** tab in the Content Server interface).
- If the Content Server user that was created during the installation is named `ContentServer`, verify that its password is not `FutureTense`. If the password is `FutureTense`, change it (**Users** node under **Content Server Management Tools** on the **Admin** tab in the Content Server interface).
- Change the default administrator username/password for your database user accounts. Note that you must also modify the `cs.privpassword` and `cs.privuser`

properties on the **Database** tab of the `futuretense.ini` file to match the new username/password.

- 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 (**Users** node under **Content Server Management Tools** on the **Admin** tab in the Content Server 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 account.

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 admin 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 CSEE 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
- EvalServer
- DebugServer

- CacheServer

CSEE Forms and Pages (Delivery System Only)

On the delivery system, be sure to disable or completely remove the following pieces of the CSEE 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 `FutureTense` as the password, change the password immediately.
   - editor

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

     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 as ContentServer/FutureTense using a CatalogManager URL:

   ```
   http://servername/pathToservlet/
   CatalogManager?ftcmd=login&username=ContentServer&password=Futu
   reTense
   ```

4. Verify that you cannot flush the entire cache as ContentServer/FutureTense using a CacheServer URL:

   ```
   http://servername/pathToservlet/
   CacheServer?all=true&authusername=ContentServer&authpassword=Fu
   tureTense
   ```

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

Additional Security Tests for the Delivery System

In addition to the preceding six steps, complete the following tasks to test your security setup on the delivery system:

1. Verify that you removed the developer forms from the web server:

   ```
   http://yourhost/futuretense_cs/Dev
   ```

   If the `futuretense_cs directory` is present, delete it and then set the `cs.wrapper` property to false.

2. Verify that you cannot log in to CS-Direct:

   ```
   http://yourhost/Xcelerate/LoginPage.html
   ```

   If you can log in, change the names of the `LoginPage` and `LoginPost` pages.

3. Verify that you mapped URLs for all servlets other than ContentServer, BlobServer, CookieServer, and Satellite to display a "404 Page Not Found" message. If you can send a request to any other servlet, you should map that URL to an error page immediately.

Section 3

Site Configuration

This section describes how to set up your CSEE sites—how to create and configure them—and how to manage assets through workflow.

It contains the following chapters:

- Chapter 4, “Sites, Start Menu, and Tree Tabs”
- Chapter 5, “Workflow”
- Chapter 6, “Sample Sites and Configuration”

Chapter 4

Sites, Start Menu, and Tree Tabs

In the CSEE content applications (CS-Direct, CS-Direct Advantage, and CS-Engage), a “site” is an object that administrators use to manage or control access to assets and that developers use as a design aid in the Content Server interface. The Burlington Financial sample site is such a site. So are the Hello Asset World and GE Lighting sample sites.

After you create your site (or sites), you use the Start Menu and tree tabs to further manage user access to assets in those sites.

This chapter contains the following sections:

- Overview
- Managing CSEE Sites
- Configuring Asset Types for Sites
- Configuring the Tree
- Configuring Other Asset Type Details

Overview

A CSEE **site** is an object that helps you design and organize the online site that you are delivering from your CSEE delivery system. You also use a site as an access control tool on the management system. A CSEE site represents your online site.

Using Sites to Control Access

You use sites to control or restrict the content providers who have access to your assets and asset types on the management system. Asset types and their subtypes, start menu items, workflow processes, tree tabs, and saved searches are all associated with specific sites. which means that access on a CSEE management system is controlled through sites (in addition to ACLs and users, that is).

Additionally, you can share individual assets between CSEE sites (providing that those sites have that asset type enabled and they have a common set of users that work in both sites). Even if an asset type is enabled through start menu items and so on, an individual asset created in one site is not available in another unless it has been shared to the other site.

Using Sites to Design an Online Site

You use sites to help design the layout of and arrange the content that is displayed on the online site represented by the CSEE site. The developers use the **Site Plan** tab to create a design framework for the online site. Each CSEE site has a separate site plan, which is stored in the `SitePlanTree` table.

Sites

CSEE sites represent real, online sites. However, they can represent those online sites in any number of ways, depending on what makes sense for your implementation. For example:

- One CSEE site can represent one complete online (public) site.
- Several CSEE sites can represent separate sections of one large online site. For example, with a catalog, perhaps people who do the data entry for household goods never do data entry for yard goods so there are separate sites that represent those areas. And, in a publication example, perhaps sports writers have a separate site that represents the sports news section and the financial writers have a separate site that represents the financial news section.

When you log in to Content Server running any of the CSEE content applications, you are logging in to a CSEE site. If you have access to more than one site, the first decision that you make after logging in is which site you want to work on. From that point on, all of the tasks that you complete are completed in the context of that site (until you switch sites).

Note

The **Admin** tab is not site-specific. That is, no matter which site you log in to, the tasks that you perform with the **Admin** tab affect all the sites on the CSEE system.

Because CSEE sites cover both design issues and access issues, you must work with your site developers when determining how to use CSEE sites and how many sites you need for your system.

After you determine how many CSEE sites you need for both design and access control reasons on your management system, you create the appropriate CSEE sites and enable the appropriate asset types for those sites on all of your systems. Part of the process of enabling the asset types is creating **start menu** items that determine which users can create assets of those types. Then, on your development and management systems, you configure which content providers and other users have access to which sites.

To create CSEE sites, you use the **Site** option on the **Admin** tab. To configure them, you can use either the **Admin** tab or the **Site Admin** tab.

Start Menu Items

Start Menu items are the links from the form that is displayed when you click the **New** button in the Content Server interface and the links from the form that is displayed when you click the **Search** button. You use them to create new assets or search for existing ones.

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. Note that if you are upgrading from a version of CS-Direct previous to 4.0, this functionality replaces the default workflow property.

Start menu items also determine which options are available for an asset type that is listed on a tree tab. 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

A tree tab is a tab that appears in the tree area of the Content Server interface. Access to tree tabs is controlled through roles—you assign roles to tree tabs.

There are several default tree tabs:

- **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 CSEE sites in the system. By default, only users with the default system role named GeneralAdmin have access to this tab.
- **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.
- **Marketing** – provides access to the following CS-Engage asset types: promotion, segment, recommendation.
- **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 CSEE 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 CSEE 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 in 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, CS-Direct 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.

Summary of Site Configuration

To create a new site and configure it includes several steps, not all of which are included in this chapter. These are the basic steps for creating and configuring a new site:

1. Create the site. See "Managing CSEE Sites" on page 67.
2. Create the users for the site. See Chapter 2, "User Management."
3. Create the roles that the users will fulfill for the site. See Chapter 2, "User Management."
4. Create workflow processes for the asset types that will use workflow. See Chapter 5, "Workflow."
5. At some point, your developers need to create the asset types for your sites. For information, see the CSEE Developer's Guide.
6. Enable and configure the asset types that are to be used on the site. See "Enabling an Asset Type for a Site" on page 69

   Configuring an asset type includes creating Start Menu items so the asset types are accessible and, perhaps, creating tree tabs that also provide access to the asset types. See "Creating Start Menu Items for Asset Types" on page 71 and "Configuring the Tree" on page 76.
7. Enable the appropriate users for the site by assigning roles to them. See Chapter 2, "User Management."
8. Configure your publishing destinations so that users can approve their assets. See Chapter 8, "Managing the CSEE Publishing System."

See also "Troubleshooting" on page 249.

Managing CSEE Sites

This section presents the basic procedures for creating and deleting your CSEE sites. Note that although you can edit the description of a site, you cannot change the name of a site after you have created it.

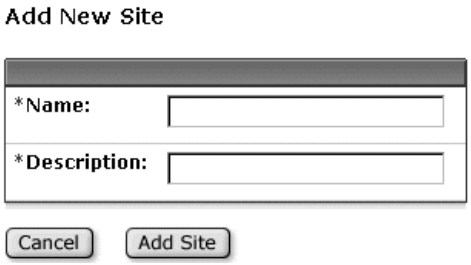
Creating a Site

A site definition consists of a name and a description. To create a site, complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand **Sites**. (This function is not available on the **Site Admin** tab.)

2. Select **Add New**.

   The **Add New Site** form appears.



3. Click in the **Name** field and enter a unique name of up to 255 characters.

   Note

   Be certain that you have entered the name of the site properly. Although you can edit the description of a site, you cannot rename it.

4. Click in the **Description** field and enter the name of the site as you want it to appear in the Content Server interface (you can use up to 64 characters). This is the text that is used in site lists throughout the user interface.
5. Click **Add**.

   CS-Direct creates the site.

Note that this procedure only creates the site.

- To make the site usable, you must configure the asset types for the site and assign roles to the appropriate users. If you plan to use the workflow feature, you should create those processes as well.
- To copy or migrate the site to another CSEE system, you use the Mirror to Server publishing method. See "Troubleshooting" on page 249.

Editing a Site Description

The only part of a site definition that you can change is the description. To change the name of a site, you must delete it and create a new one with the correct name.

To edit a site description, complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand **Sites**.
2. Select the site that you want to change.

3. Click **Edit**.
4. In the **Edit** form, click in the **Description** field and make the appropriate changes.
5. Click **Save.**

Deleting a Site

When you delete a site, not only is the site and any configuration information specified for the site deleted—any assets that were created for the site and are not shared with any other sites become inaccessible. Therefore, before deleting a site, be sure that you have shared any assets that you need to keep.

To delete a site, complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand **Sites**.
2. Select the site that you want to delete. If you want to select a newly created site that does not appear on the tree yet, refresh the tree.
3. Click **Delete**.

   The system displays a confirmation message.
4. Click **OK**.

Configuring Asset Types for Sites

To configure the asset types for a site, you enable the asset types and then create start menu items for them. Before creating the start menu items, however, first you must enable the asset types for the site.

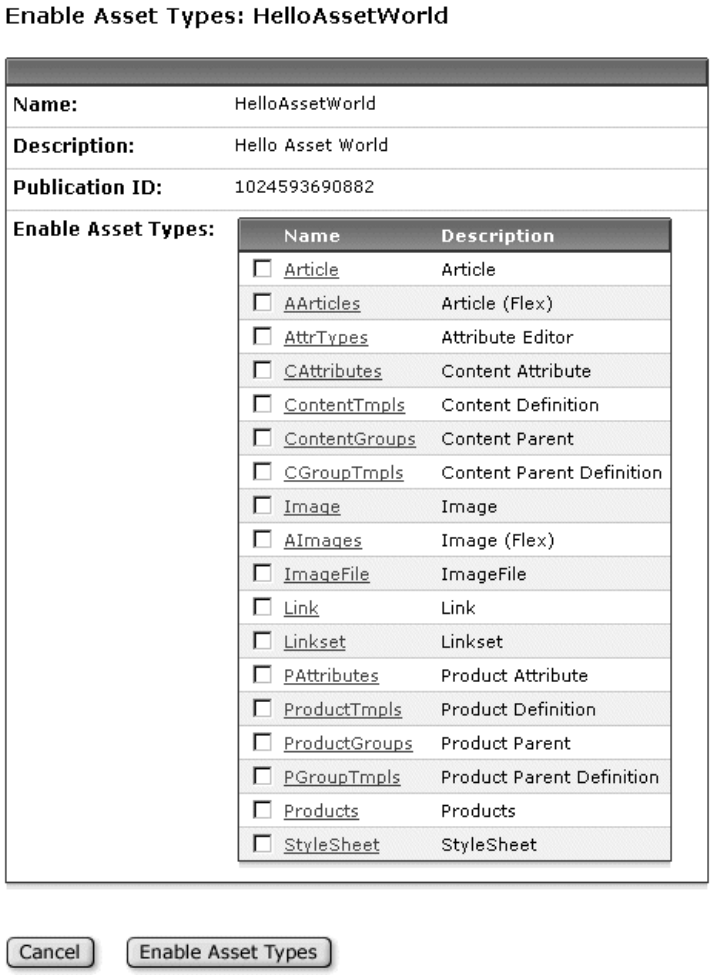
Enabling an Asset Type for a Site

This procedure describes how to enable an asset type for use within the Content Server interface. Enabling an asset type so that it is available within Microsoft Word on the Content Server toolbar is described in “Configuring the Word Asset Types for CS-Desktop” on page 172.

To add an asset type to the list of those available for a site, complete the following steps:

1. In the Content Server interface, do one of the following:
   - If you have the Site Admin role, but not the GeneralAdmin role, select the **Site Admin** tab.
   - If you have the GeneralAdmin role, select the **Admin** tab, expand the **Sites** option, and then expand the site that you want to enable asset types for.
2. Select **Asset Types** > **Enable**.

The **Enable Asset Types** form appears. It displays all the asset types that exist on this system that have not yet been enabled for this site:



3. Select the asset types that you want to enable for this site.
4. Click **Enable Asset Types**.

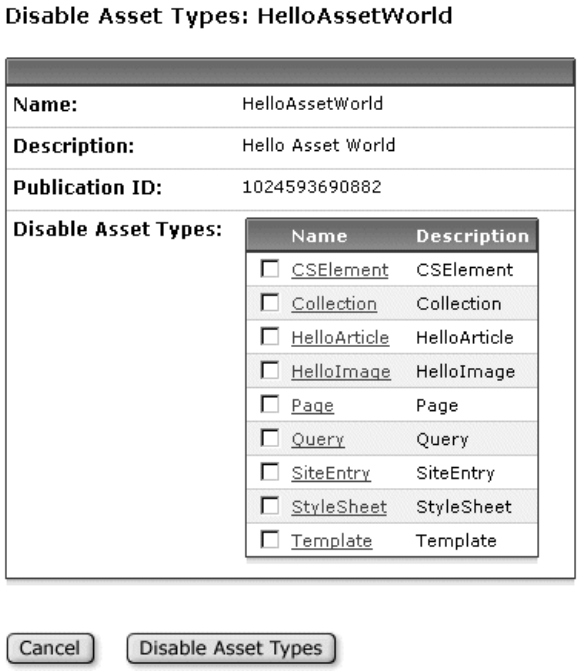
Now the asset types are enabled for the site, but no one can create assets of these types until you create Start Menu items for them. Continue to "Creating Start Menu Items for Asset Types" on page 71.

Removing an Asset Type from the Site

To remove an asset type from the list of those available at the selected site, complete the following steps:

1. In the Content Server interface, do one of the following:
   - If you have the Site Admin role, but not the GeneralAdmin role, select the **Site Admin** tab.
   - If you have the GeneralAdmin role, select the **Admin** tab, expand the **Sites** option, and then expand the site you want to disable asset types for.
2. Select **Asset Types** > **Disable**.

The **Disable Asset Types** form appears. It displays the asset types that have been added for this site:



3. Select the asset types that you want to remove.
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.

Creating Start Menu Items for Asset Types

When you create a start menu item, you specify the sites the item is for and you specify 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.

There are four kinds of start menu items:

- **New** – creates a link for the asset type on the list that is displayed when you click the **New** button located in the button bar at the top of the Content Server 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.
- **Search** – creates a link for the asset type on the list that is displayed when you click the **Search** button located in the button bar at the top of the Content Server 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.

- **CS-Desktop** – makes the asset type available to content providers who use the CS-Desktop interface instead of the Content Server interface. For information about creating CS-Desktop start menu items, see "Configuring the Word Asset Types for CS-Desktop" on page 172.
- **CS-DocLink** – makes the asset type available to content providers who use the CS-DocLink interface instead of the Content Server interface. For information about creating CS-DocLink start menu items, see "Configuring the CS-DocLink Asset Types" on page 179.

Creating Start Menu Items for the New List

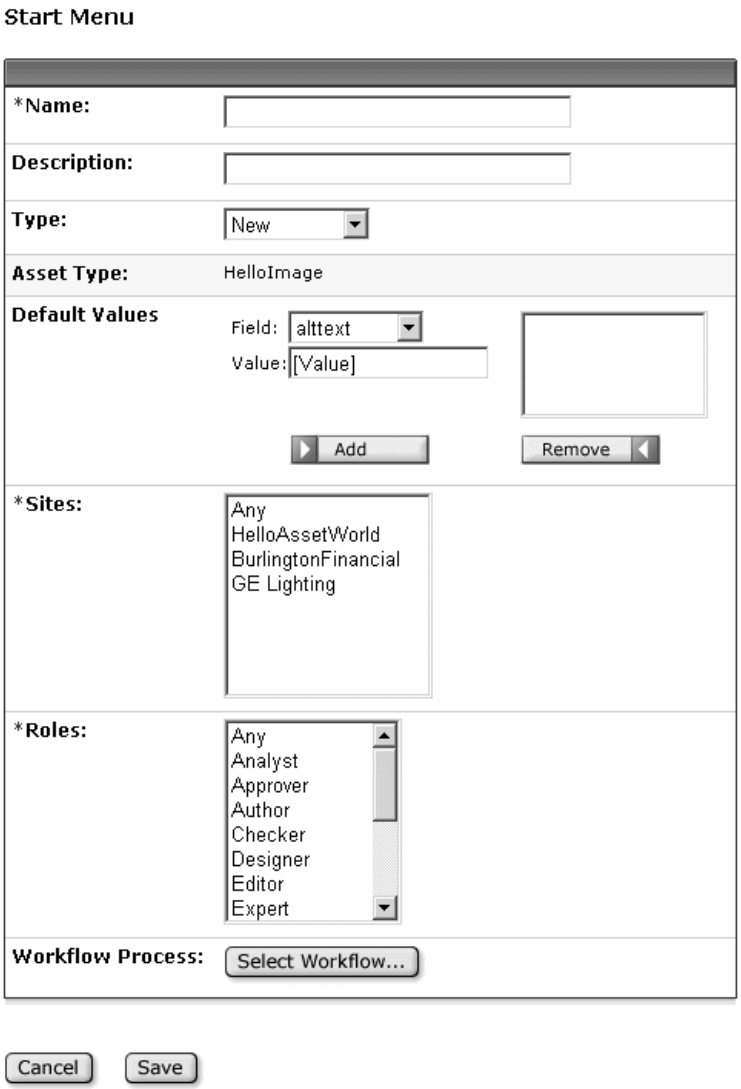
Before you begin creating start menu items, note the following:

- Be sure to create start menu items for only the asset types that are enabled for that site.
- If you want to specify a default workflow process, you should create the workflow process first. For information about workflow, see Chapter 5, "Workflow."
- 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 start menu item for the **New** list, complete the following steps:

1. In the Content Server interface, select the **Admin** tab. (This feature is not available on the **Site Admin** tab.)
2. Select **Asset Types** > ***Asset Type*** > **Start Menu**, where ***Asset Type*** is the asset type that you want to create the item for.
3. Click **Add New**.

The **New Start Menu Item** form appears:



4. Click in the **Name** field and 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”). This is the name that is displayed on the **New** list.
5. Click in the **Description** field and enter a short description using up to 255 characters.
6. In the **Type** field, select **New**.

7. (Optional) If you want field values to be set automatically when a content provider creates an asset with this start menu item, complete these steps:

   a. Select a field from the **Field** drop-down list. This list displays all the fields for assets of this type; for example, Category, Source, Template, and so on.

   Note

   For flex assets, the field named **TemplateChosen** refers to the definition for the asset.

   - If you are creating a start menu for a flex parent, **TemplateChosen** means the name of the parent definition asset.
   - If you are creating a start menu for a flex asset, **TemplateChosen** means the name of the flex definition.

   b. Click in the **Value** field and enter the value. Note the following about this field:

      All values are case-sensitive.

      CS-Direct does not validate the value against the field that you are setting it for. Therefore, you must be sure that an appropriate value is entered for the field. For example, the value for the **TemplateChosen** field must be the name of a valid flex parent definition or flex asset definition.

      If you are setting a value for the **Category** field, use the category code rather than the name.

      Note that 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 you set the value to in the start menu form, the content provider who creates an asset with the start menu item can override that value. These values are for convenience only.

      There are two exceptions:

      First, the **TemplateChosen** field for flex assets. If the start menu item sets the flex definition for a flex asset or the flex parent definition for a flex parent asset, the user cannot change that definition.

      Second, the **Subtype** field for the template asset. If the start menu item sets the subtype for a template, the user cannot change that subtype.

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.

   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 authorized for the start step in the workflow process.

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

10. (Optional) To configure workflow process details for assets 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**.

       The system saves the workflow information and displays the **Start Menu** form again.
11. In the **Start Menu** form, click **Save**.

Creating Start Menu Items for the Search List

To create a start menu item for the **Search** list, complete the following steps:

1. In the Content Server interface, select the **Admin** tab.
2. Select **Asset Types** > ***Asset Type*** > **Start Menu**, where ***Asset Type*** is the asset type that you want to create the item for.
3. Click **Add New**.

   The **New Start Menu Item** form appears.
4. Click in the **Name** field and 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.
5. Click in the **Description** field and enter a short description for the start menu item.
6. In the **Type** field, select **Search**.
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. In the **Sites** field, select which sites can use this start menu item. Use Ctrl + click to select more than one site.
9. Click **Save**.

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 CSEE system, there is a top-level **Start Menu** node on 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.

Configuring the Tree

You can configure the tree that is displayed in the left side of the Content Server interface in three ways:

- You can configure the tabs themselves, adding items or functionality to them, and determining which users have access to them based on their roles.
- You can determine whether the tree appears in the main window by default.
- You can specify the number of items that are displayed under a node on a tree tab.

There are three general categories of tree tabs:

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

Note

The Workflow Groups tab is a hybrid between a system default and a custom tab. Although it 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.

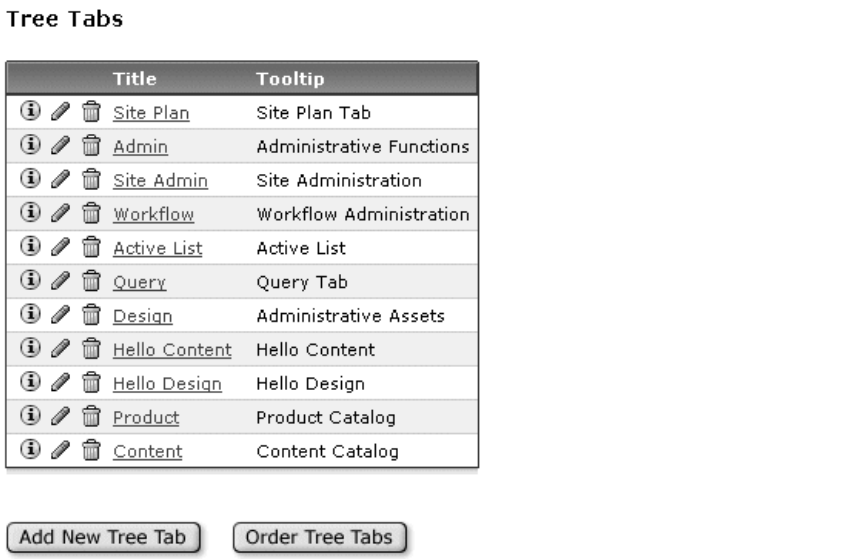
For information about creating custom tree tabs that implement new functionality, see the section called "Management System Features" in the *CSEE Developer's Guide*.

Creating Tree Tabs

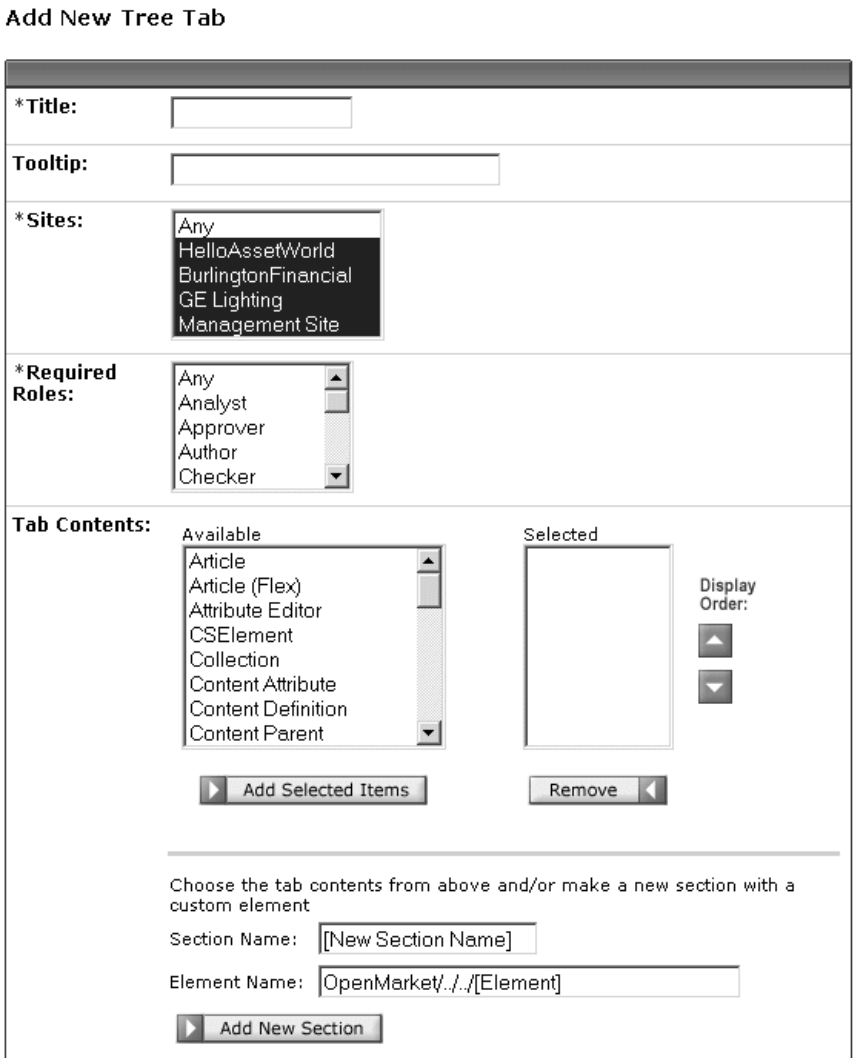
To create a new tree tab, complete the following steps:

1. In the Content Server interface, select **Admin** > **Tree**.

   CS-Direct displays a list of the tabs.



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



3. Click in the **Title** field and enter a short, descriptive name of up to 64 characters.
4. Click in the **Tooltip** field and enter a short, descriptive phrase using up to 255 characters. This phrase is displayed when the mouse hovers over the tab in the tree.
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** field, select the asset types that will be displayed on the tab and then click **Add Selected Items**.
8. (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 *CSEE Developer's Guide* for more information.
9. Click **Save**.

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 88. This tab does not appear by default. If you want to use it, you must create it.

To create the Workflow Groups tab, complete the following steps:

1. In the Content Server interface, select **Admin** > **Tree**.

   CS-Direct displays a list of the tabs.
2. In the **Tree Tabs** form, click **Add New Tree Tab**.
3. Click in the **Title** field and enter a short, descriptive name of up to 64 characters. For example: Groups
4. Click in the **Tooltip** field and enter a short, descriptive phrase using 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 CSEE sites, you must edit the following system default tabs to add your roles so that your users have access to them:

- Site Plan
- History
- Active List

To edit a tree tab, complete the following steps:

1. In the Content Server interface, select **Admin tab > Tree**.
2. In the tree tab list, click the **Edit** icon for the tab that you want to edit.
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, complete the following steps:

1. In the Content Server interface, select **Admin tab > Tree**.
2. In the tree tab list, click the **Delete** icon for the tab that you want to edit.

   The system displays a confirmation message.
3. Click **Delete Tree Tab.**

Changing the Sort Order of the Tree Tabs

You can change the order in which the tree tabs are displayed. To change their order, complete the following steps:

1. In the Content Server interface, select **Admin > Tree**.
2. At the bottom of the list of tree tabs, click **Order Tree Tabs**.

CS-Direct displays a form like this one:

3. Select a tab from the list and then use the arrow buttons to move the tab to the correct position.
4. Repeat step 3 for each tab that you want to reposition.
5. Click **Save**.

Configuring the Number of Items Displayed Under Nodes on Tabs

When your database holds many assets of any one type, it can take a long time for a tree tab to load all of those assets and to then display their names under a node on a tab. For this reason, you can limit the number of items that are displayed under the nodes. Then, when the number of assets or other items for a node exceeds that number, the Content Server interface prompts the user to enter search criteria that restricts the number of items.

For example, suppose that the tree is configured to limit the items displayed under an individual node to 100, there are more than 300 articles, and a user expands a node icon that represents the articles. At that point, the Content Server interface displays a small search criteria form that prompts the user to enter criteria that CS-Direct uses to limit the number of article assets that it returns.

To configure the number of items that the tree displays for a node, complete the following steps:

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. Select **File** > **Save**.

5. Select **File** > **Close**.
6. Stop and restart the application server.

Configuring Whether the Tree Is Displayed by Default

Any user can toggle the tree on and off whenever they need more space in the main window to display the assets that they are working on.

If you want, you can configure your CSEE 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 CS-Direct 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 whether the tree is displayed and who can toggle the tree on (if you have decided to suppress it), complete the following steps:

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 Content Server 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. Select **File** > **Save**.
6. Select **File** > **Close**.
7. Stop and restart the application server.

Configuring Other Asset Type Details

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

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 *CSEE Developer's Guide*.

Chapter 5

Workflow

Workflow is a feature provided by CS-Direct 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
- Planning Your Workflow Processes
- Sample Site Workflow Processes
- Configuring Your Workflow Processes
- Moving Your Work
- Clearing Workflow Assignments

Overview

The end goal for any asset is for it to be 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 CS-Direct 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 interface. 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 they fulfill 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 on the **My Work** form in the Content Server 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 CS-Direct 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

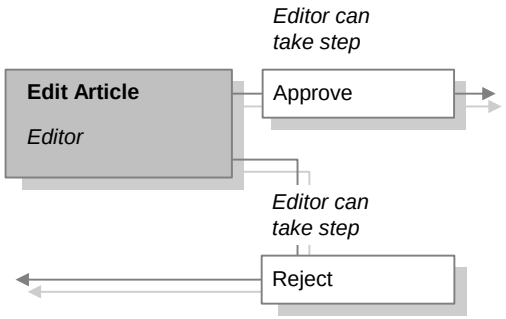
association fields filled—that an article has the associated images that it needs—before it progresses to the next state.

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:

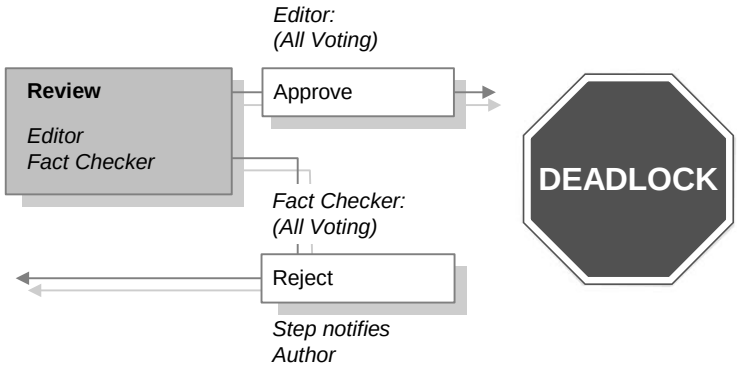
- Configure the steps so that the first participant to finish the assignment (vote) determines the direction the asset takes.
- Configure the steps so that all participants have to select the same step (vote the same way) in order for the asset to progress in either direction. A step that all the participants 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 CS-Direct 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 on 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 CS-Direct 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 CS-Direct completes when a step is taken.
- Actions that are triggered by a deadline. These actions are queued and are triggered only after CS-Direct 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 Content Server 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 interface, see Chapter 2, "User Management."

Planning Your Workflow Processes

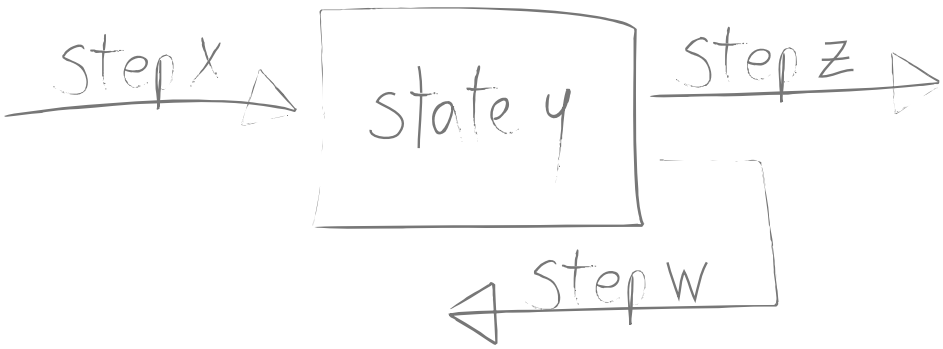
When you create a workflow process, you create steps that link together the states for that process. This means that you or someone else must create the workflow components—roles, e-mail messages, the various kinds of actions, step conditions, and states—before you can create a workflow process.

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 Content Server interface to create your workflow processes, described in “Configuring Your Workflow Processes” on page 116.

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 CSEE 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 2, "User Management."

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.

See “Using the Property Editor” on page 276 for instructions on how to start and use the Property Editor to verify or modify property values.

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 *CSEE Developer’s Guide* for information about customizing workflow.

CS-Direct 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 *CSEE 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 on the **Admin** tab in the Content Server 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 Content Server interface, select **Admin > 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 Content Server interface, select **Admin > Workflow Actions > 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 CS-Direct 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 Content Server interface, select **Admin > Workflow Actions > 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 Content Server interface, select **Admin > Workflow Actions > 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 CS-Direct 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 CS-Direct 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 CS-Direct 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.
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.
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.

Function	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.
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 CSEE 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 CSEE 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 102.
- Whether any of the CSEE 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 105.

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. (Available only when you have CS-Direct Advantage.)
- **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 you have CS-Engage.)
- **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 you have CS-Engage.)

The following sections describe the first three workflow processes: HelloArticle Process, BF: Normal Article Process, and GE Lighting Process.

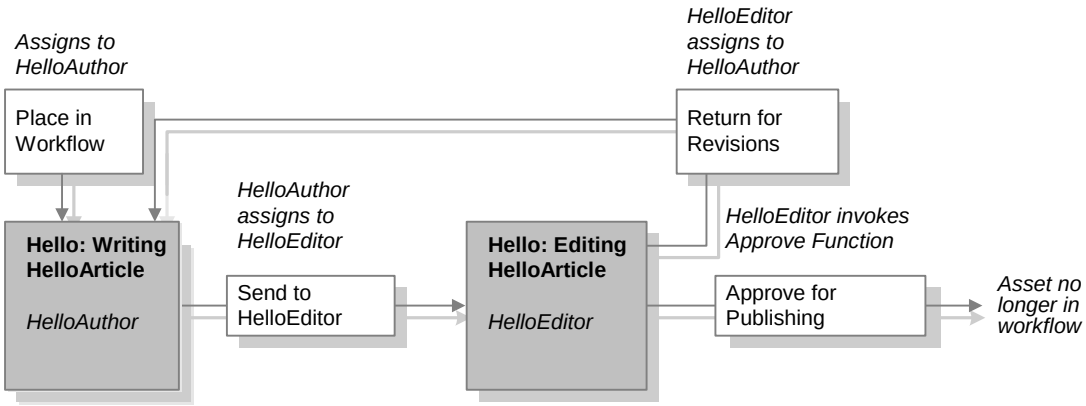
HelloArticle Process

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:



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.

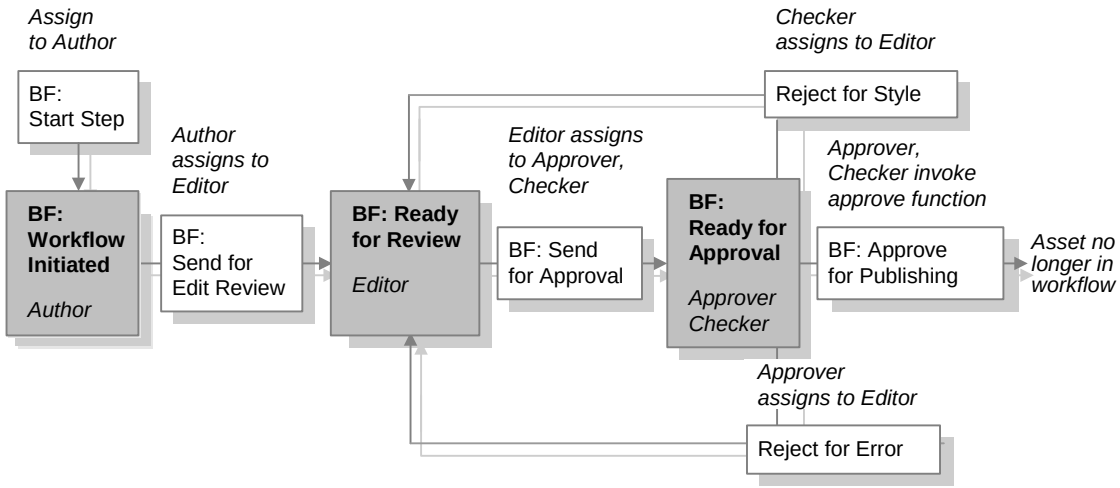
BF: Normal Article Process

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, CS-Direct 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.

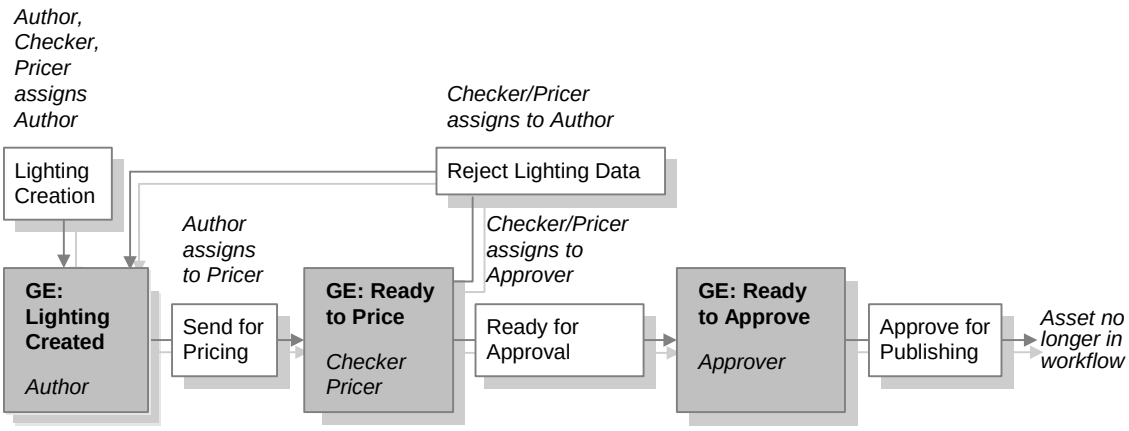
GE Lighting Process

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 93 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 2, "User Management" 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 94 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 "Set Up E-Mail Objects" on page 117.
4. Create the step actions, timed actions, deadlock actions, group deadlock actions, and delegate actions that you need. See "Set Up the Workflow Actions and Conditions" on page 119.
5. Create your states. See "Set Up the States" on page 123.
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. "Set Up the Workflow Processes" on page 125.
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. "Set Up the Timed Action Event" on page 122.
8. Test your workflow processes. See "Test Your Workflow Process" on page 134.

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 for Asset Types” on page 71.

Set 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:

- The properties on the **Email** tab in the `futuretense.ini` file must be configured to provide information about your e-mail server. See “Email” on page 314 for more information.
- The `xcelerate.emailnotification` property in the `futuretense_xcel.ini` file must be set to `true`.
- The workflow participants must have e-mail addresses specified in their user profiles. (For information about creating user profiles, see “Creating a New User” on page 40.)

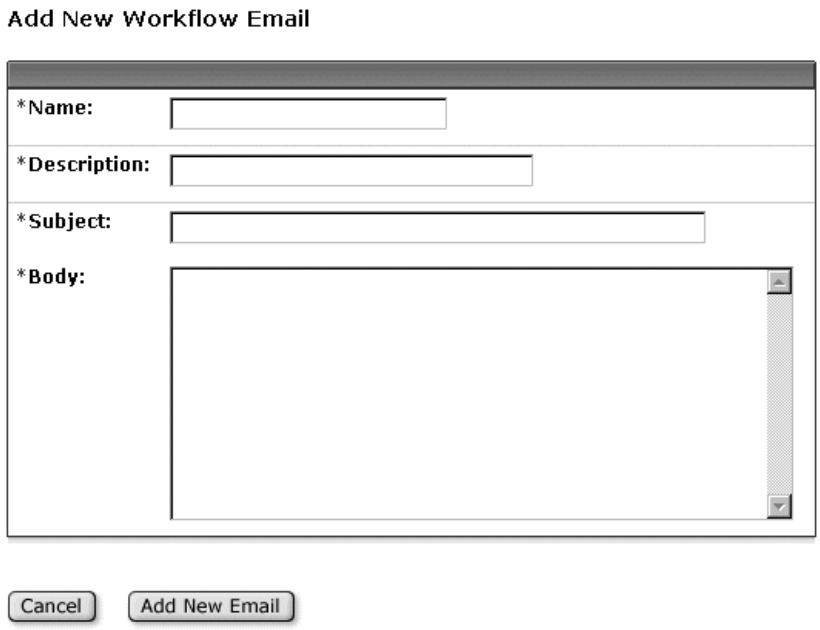
For information about using the Property Editor to configure these property, see “Starting the Property Editor” on page 276 and “Setting Properties” on page 276.

Creating E-Mail Objects

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Email** item.
2. Under **Email**, double-click on **Add New**.

   CS-Direct displays the **Add New Workflow Email** form:



3. In the **Add New Workflow Email** form, click in the **Name** field and enter a unique name of up to 36 characters. (This is the name that you use with the `emailname` variable when you use this e-mail object with an action).
4. In the **Description** field, enter a short, descriptive phrase of up to 36 characters.
5. In the **Subject** field, enter a phrase that describes the subject of the e-mail message. See "About E-Mail Objects" on page 96 for a list of variables that you can use.
6. In the **Body** field, enter the text for the message. Once again, see "About E-Mail Objects" on page 96 for a list of variables that you can use.
7. Click **Save**.

Editing E-Mail Objects

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Email** item.
2. Under **Email**, double-click on the e-mail object that you want to edit.
3. In the **Inspect** form, click the **Edit** icon.
4. In the **Edit Workflow Email** form, make the necessary changes. See "About E-Mail Objects" on page 96 for a list of variables that you can use.
5. Click **Save**.

Deleting E-Mail Objects

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Email** item.
2. Under **Email**, double-click on the e-mail object that you want to delete.
3. In the **Inspect** form, click the **Delete** icon.

   A confirmation message appears.
4. Click **Delete**.

Set 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. CS-Direct provides several default action elements and e-mail objects. Use the Content Server 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, “Set Up E-Mail Objects” on page 117 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.

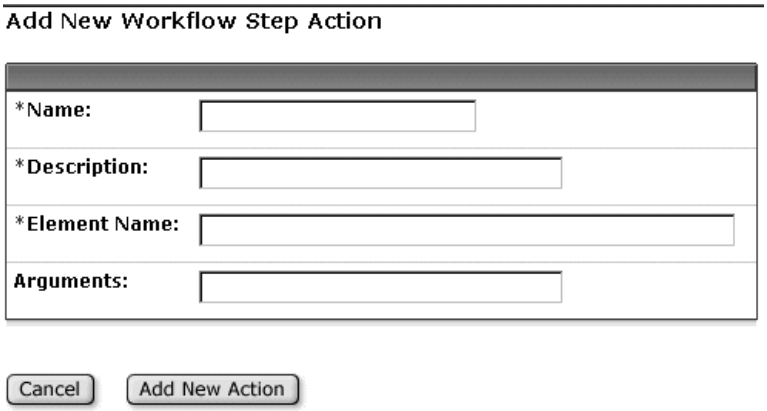
Your user account must give you access to the **Admin** tab in order for you to create actions or conditions.

Creating Workflow Actions and Conditions

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Workflow Actions** item.
2. Under **Workflow Actions**, expand the kind of item that you want to create: **Step Actions**, **Step Conditions**, **Timed Actions**, **Delegate Actions**, **Deadlock Actions**, or **Group Deadlock Actions**.
3. Under the item that you selected, double-click **Add New**.

   CS-Direct displays the **Add New** form for that type of action or condition. For example, this is the **Add New Step Action** form:



4. In the **Add New** form, click in the **Name** field and enter a unique name of up to 40 characters.
5. In the **Description** field, enter a short, descriptive phrase of up to 40 characters.
6. In the **Element Name** field, type the entire name of the element. For example, to use the default workflow element named `SendEmailToAssignees`, type: `OpenMarket/Xcelerate/Actions/Workflow/StepActions/SendEmailToAssignees.`
7. 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`
8. 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 CSEE 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

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Workflow Actions** item.
2. Under **Workflow Actions**, expand the kind of item that you want to edit: **Step Actions**, **Step Conditions**, **Timed Actions**, **Delegate Actions**, **Deadlock Actions**, or **Group Deadlock Actions**.
3. Under the item that you selected, double-click the name of the item that you want to edit.
4. In the **Inspect** form, click the **Edit** icon.
5. In the **Edit Action** (or **Condition**) form, make your changes.
6. Click **Save**.

Deleting Workflow Actions and Conditions

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Workflow Actions** item.
2. Under **Workflow Actions**, expand the kind of item that you want to delete: **Step Actions**, **Step Conditions**, **Timed Actions**, **Delegate Actions**, **Deadlock Actions**, or **Group Deadlock Actions**.
3. Under the item that you selected, double-click the name of the item that you want to delete.
4. In the **Inspect** form, click the **Delete** icon.

   A confirmation message appears.
5. Click **Delete**.

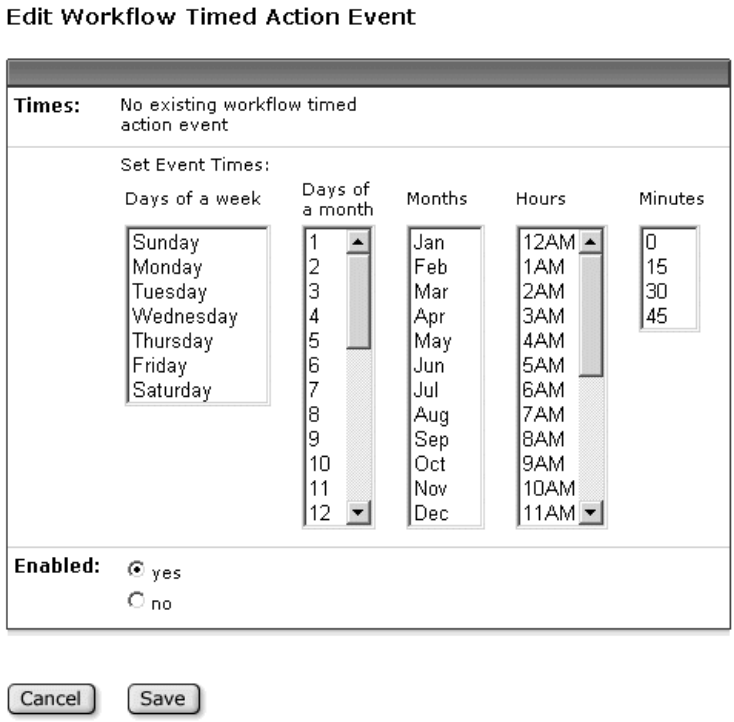
Set Up the Timed Action Event

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 for a CSEE system.

Complete the following steps:

1. In the Content Server interface, select **Admin > Timed Action Event**.
2. In the **Workflow Timed Action Event** form, click **Edit**.

   The **Edit Workflow Timed Action Event** form appears:



3. Set the schedule for how often the state deadlines should be calculated in terms of months, days, hours, and 15-minute increments.
4. Under **Enabled**, click **yes**.
5. Click **Save**.

   A summary of the schedule is displayed.

CS-Direct 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 242.

Set 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.

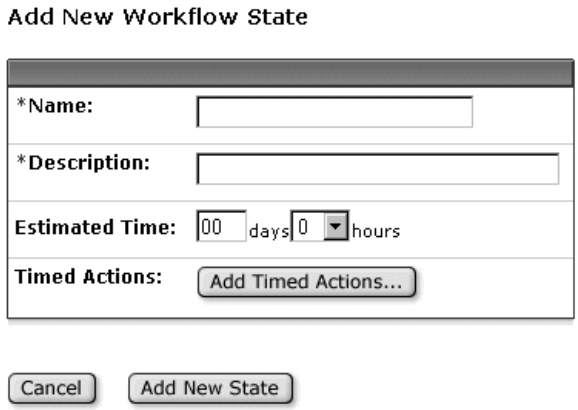
To work with workflow states, your user name must be assigned the WorkflowAdmin role for the site that you are working with.

Creating Workflow States

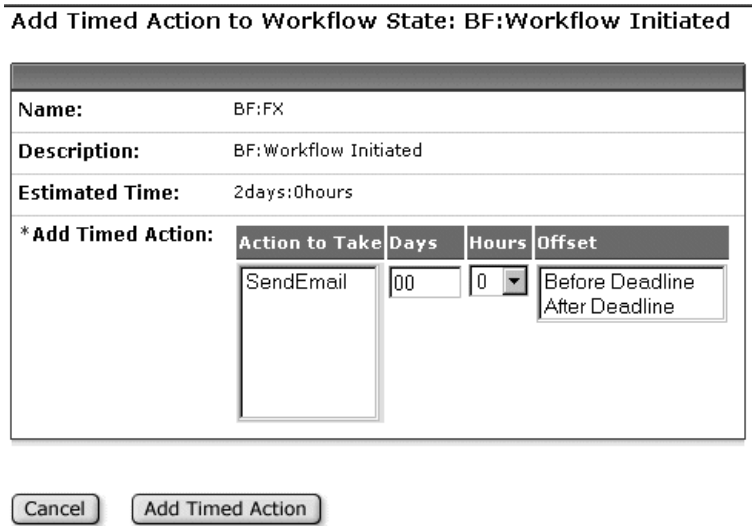
Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **States** item.
2. Under **States**, double-click on **Add New**.

   CS-Direct displays the **Add New Workflow State** form:



3. In the **Add New** form, click in the **Name** field and enter a unique, meaningful name of up to 40 characters.
4. In the **Description** field, enter a short, descriptive phrase of up to 40 characters.
5. (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.
6. (Optional) Click the **Add Timed Actions** button and complete the following steps:



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.

7. Click **Add New State**.

Editing Workflow States

Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **States** item.
2. Under **States**, double-click on the state you want to edit.
3. In the **Inspect** form, click the **Edit** icon.
4. In the **Edit** form, make the appropriate changes.
5. 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.
6. Click **Save**.

Deleting Workflow States

Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **States** item.
2. Under **States**, double-click on the state you want to delete.
3. In the **Inspect** form, click the **Delete** icon.

   A confirmation message appears.
4. Click **Delete**.

Set 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 78.

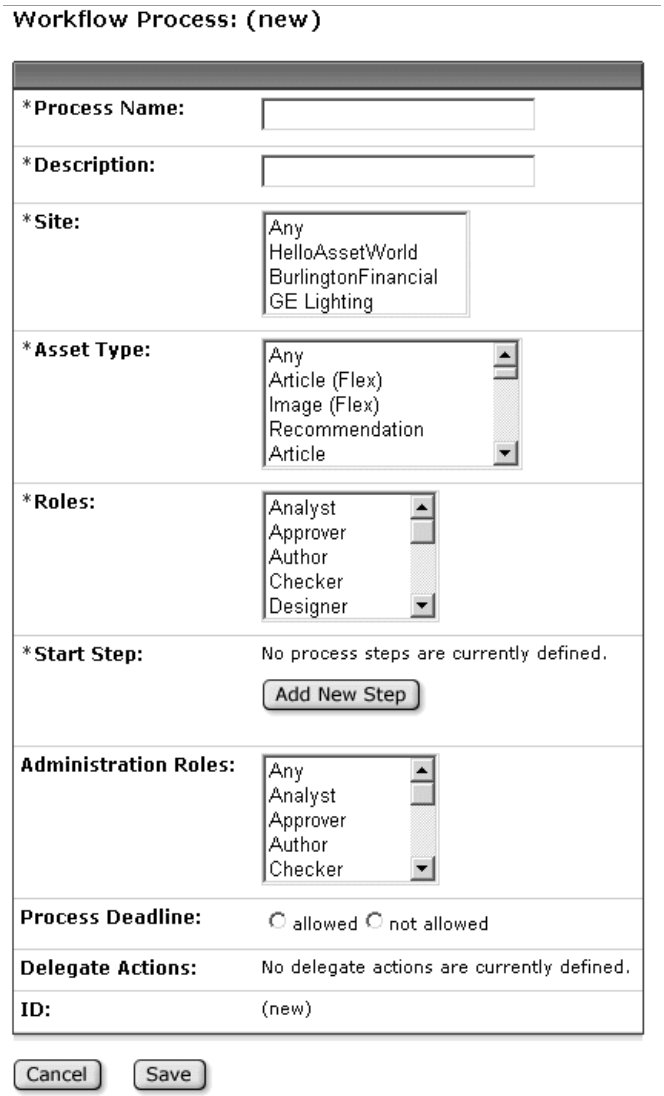
Creating Workflow Processes

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 Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on **Add New**.

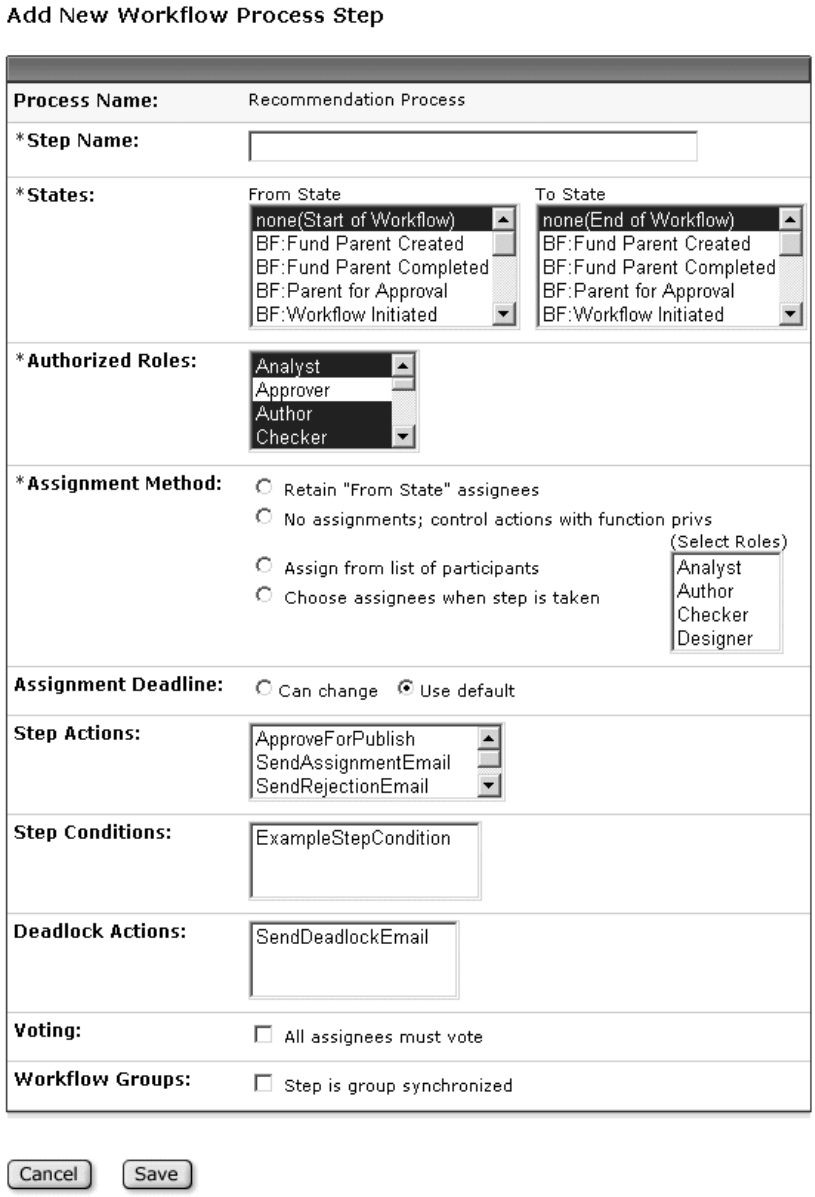
   The **Workflow Process: (new)** form appears:



3. In the **New Workflow Process** form, click in the **Name** field and enter a unique name of up to 25 characters.
4. In the **Description** field, enter a short, descriptive phrase of up to 64 characters.
5. In the **Sites** list, select the sites that can use this workflow process.
6. In the **Asset Type** list, select the asset type(s) that this workflow process is for.
7. In the **Roles** list, select the roles that will participate.
8. In the **Administration Roles** list, select the roles that can act as the administrator for this workflow process when an asset is using it.
9. 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.
10. If you have any delegate actions configured, select the appropriate actions in the **Delegate Actions** list.

11. In the **Start Step** section, click **Add New Step**.

Step B: Create the Start Step



1. In the **New Workflow Process Step** form, click in the **Step Name** field and 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. From 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 Step" 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, 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 102.)
5. 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.
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**.

   The process is saved and displayed in the **Inspect** form for the process. 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. From the **Inspect** form of the workflow process, select the **Edit** icon.
2. Click **Add/Edit Steps** (a button at the bottom of the form).
3. In the **Steps for Process** form, click **Add Step**.
4. In the **Add New Workflow Process Step** form, click in the **Step Name** field and 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. From 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, 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 102.)
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 86 and "About Delegate Actions" on page 98.
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**.

    The process is saved and displayed in the **Inspect** form for the process.
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 131.

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. From the **Inspect** form of the workflow process, select the **Edit** icon.
2. In the **Edit Process** form, click **Add/Edit Steps** (a button at the bottom of the form).
3. In the **Steps for Process** form, click **Add Step**.
4. In the **New Workflow Process Step** form, click in the **Step Name** field and 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. From 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 86 and "About Delegate Actions" on page 98.
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**.

    The process is saved and displayed in the **Inspect** form for the process.

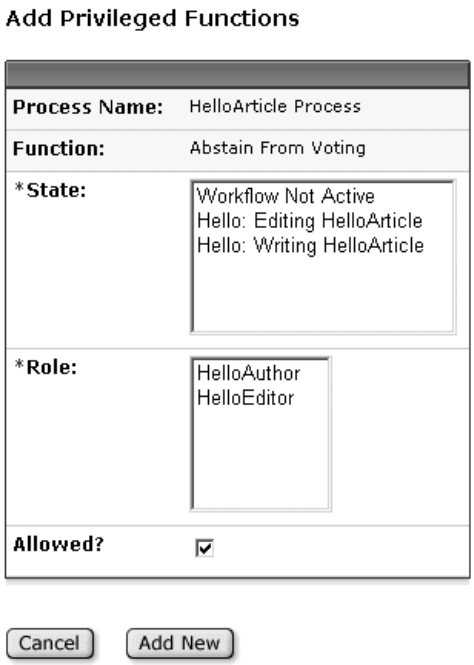
If you do not need to configure function privileges, this workflow process is completed. If you do need to configure function privileges, continue to the next procedures.

Step E: (Optional) Configure Function Privileges

Use the following procedure to set up function privileges:

1. From the **Inspect** form of the workflow process, select the **Edit** icon.
2. In the **Edit** form, click **Add/Edit Function Privs** (a button at the bottom of the form).
3. In the **Workflow Process Functions** form, scroll down to the function that you want to set a privilege for.
4. Click the **New** button next to the function.

   The **Add Privileged Function** form appears:



5. Select the appropriate state from the **State** list.
6. Select the roles that are allowed or not allowed to perform this function when an asset is in this state from the **Roles** list.
7. Do one of the following:
   - To allow users with those roles to perform the function, select the **Allowed** option.
   - To restrict users with those roles from performing the function, clear the **Allowed** option.
8. Click **Add New**.

   The privilege is set and the **Workflow Process Functions** form appears again.
9. Repeat steps 3 through 8 for each function privilege that you need to configure. For more information about function privileges, see “Determine the Function Privileges” on page 105.
10. After you have configured all of your function privileges, click **Save**.

    The privileges are saved and the **Inspect Process** form appears again.

The workflow process is complete.

Editing Workflow Processes

1. In the Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on the workflow process that you want to edit.
3. In the **Inspect** form, click the **Edit** icon.
4. To modify the name, description, or any of the other global settings, make your changes directly in the **Edit** process form.
5. To edit the steps, see "Editing Steps" on page 132
6. To edit the function privileges, see "Editing Function Privileges" on page 132.
7. When you are finished, click **Save**.

Editing Steps

Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on the workflow process that you want to edit.
3. In the **Inspect** form, click the **Edit** icon.
4. In the **Edit Process** form, click **Add/Edit Steps** (a button at the bottom of the form).
5. In the **Steps for Workflow Process** form, click **Edit** next to the step that you want to change.
6. In the **Edit Workflow Process Step** form, make your changes and then click Save.
7. The workflow process is saved and the **Inspect** form appears.

Editing Function Privileges

To edit a function privilege, you must delete it and then re-create it. Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on the workflow process that you want to edit.
3. In the **Inspect** form, click the **Edit** icon.
4. In the **Edit Process** form, click **Add/Edit FunctionPrivs** (a button at the bottom of the form).
5. In the **Functions for Workflow Process** form, click **Remove** next to the function that you want to change.
6. Then click **New** next to the same function.
7. In the **Add Privileged Functions** form, create the function privilege and then click **Save**.

8. In the **Functions for Workflow Process** form, click **Save**.

   The workflow process is saved and the **Inspect** form appears.

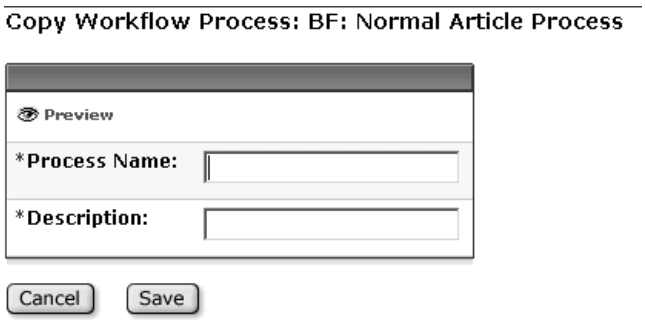
Copying Workflow Processes

You can copy a workflow process, which can save you some steps in configuring additional processes.

Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on the workflow process that you want to copy.
3. In the **Inspect** form, click in the drop-down list on the action bar and select **Copy Process**.

   The **Copy Workflow Process** form appears:



4. In the **Process Name** field, enter the name of the process.
5. In the **Description** field, enter a short, descriptive phrase.
6. Click **Save**.
7. Edit the process as necessary. See the following procedures for help:
   - “Editing Workflow Processes” on page 132.
   - “Editing Steps” on page 132.
   - “Editing Function Privileges” on page 132.

Deleting Workflow Processes

Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on the workflow process that you want to delete.
3. In the **Inspect** form, click the **Delete** icon.

   A confirmation message appears.
4. Click **Delete**.

Deleting Steps

Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on the workflow process that you want to edit.
3. In the **Inspect** form, click the **Edit** icon.
4. In the **Edit Process** form, click **Add/Edit Steps** (a button at the bottom of the form).
5. In the **Steps for Workflow Process** form, click **Remove** next to the step that you want to delete.
6. Click **Save**.

   The workflow process is saved and the **Inspect** form appears.

Deleting Function Privileges

Complete the following steps:

1. In the Content Server interface, select the **Workflow** tab and expand the **Processes** item.
2. Under **Processes**, double-click on the workflow process that you want to edit.
3. In the **Inspect** form, click the **Edit** icon.
4. In the **Edit Process** form, click **Add/Edit FunctionPrivs** (a button at the bottom of the form).
5. In the **Functions for Workflow Process** form, click **Remove** next to the function that you want to change.
6. Click **Save**.

   The workflow process is saved and the **Inspect** form appears.

Test Your Workflow Process

Before you move your workflow process to the management system and implement it there, test it on your development system:

- 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.

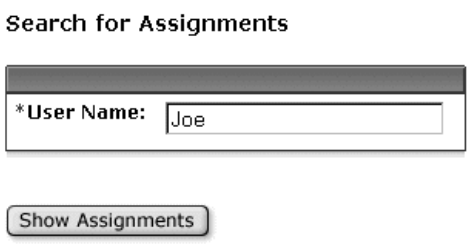
For information about this feature, see “Migrating a Site from One System to Another System” on page 232.

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, complete the following steps:

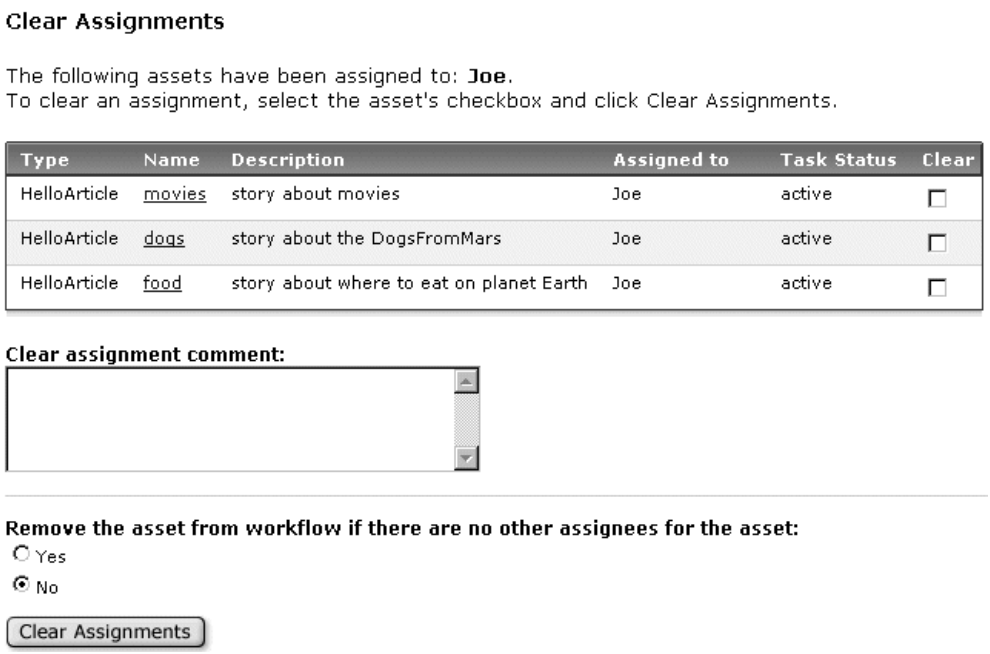
1. In the Content Server interface, select the **Admin** tab and double-click on **Clear Assignments**.
2. In the **Search for Assignments** form, enter a user name. For example:



3. Click **Show Assignments**.

   CS-Direct displays a list of all the assets that are currently assigned to that user.

For example:



4. In the **Clear Assignments** form, select the **Clear** option next to each assignment that you want to clear.
5. (Optional) Click in the **Clear assignment comment** field and enter a comment about why you are clearing these assignments. The text that you enter in this field appears in the **Action Taken** field in the **Workflow History** section on the **Status** form for the asset.
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**.

   CS-Direct displays a **Clear Assignments Report** that summarizes the changes.

Chapter 6

Sample Sites and Configuration

As described in Chapter 4, “Sites, Start Menu, and Tree Tabs,” in the CSEE content applications (CS-Direct, CS-Direct Advantage, and CS-Engage), a “site” is an object that administrators use to manage or control access to assets and that developers use as a design aid in the Content Server interface.

The CSEE content applications deliver four sample sites that demonstrate both sides of a CSEE site: the online site that is presented to the public from the delivery system and the content management site on the management system that the content providers interact with in order to supply the content for the online site.

The sample sites vary in complexity and purpose:

- **Hello Asset World** is an extremely simple site that illustrates the primary coding and design concepts behind extracting and displaying basic assets.
- **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 the CS-Direct product.
- **GE Lighting** is an online catalog of lighting products. It illustrates the power and flexibility of the flex asset model. When CS-Engage is installed, it also uses some of the CS-Engage features.
- **Burlington Financial Extension** provides an extension to the Burlington Financial site that illustrates the CS-Engage features, adds two families of flex asset types to the data model, and uses Analysis Connector.

In addition to providing code samples for your site designers, all four sites provide examples of how to configure the CSEE content management site that runs on a management system.

This chapter describes the configuration of the sample sites in the following sections:

- Hello Asset World
- Burlington Financial
- GE Lighting
- Burlington Financial Extension (for CS-Engage)

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 *CSEE 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)

Username	Password	ACLs	Roles
Flo	hello	Browser, ElementReader, PageReader, UserReader, xceleditor	HelloEditor
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 *CSEE 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	

Name	Asset Type	Start Menu Type	Roles	Default Values
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 Admin (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 109.

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 *CSEE 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
- admin, 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
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
admin	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 *CSEE 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

Asset Type	Number of Assets	Association Fields	Subtype	Categories
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	

Name	Asset Type	Start Menu Type	Roles	Default Values
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	
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	

Name	Asset Type	Start Menu Type	Roles	Default Values
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	admin
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	admin
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	admin

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 111.

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 *CSEE 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
- admin, 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 administive 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
user_pricer	user	Browser, ElementReader, PageReader, UserReader, xceleditor	Pricer
admin	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

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 *CSEE 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 CS-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

Name	Asset Type	Start Menu Type	Roles
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
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

Name	Asset Type	Start Menu Type	Roles
New Promotion	promotion	New	Marketer, Designer
Find Promotion	promotion	Search	Marketer, Designer
New Recommendation	recommend-ation	New	Marketer, Designer
Find Recommendation	recommend-ation	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	admin
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

Tab	Description	Roles	Users with Access
History (default)	Displays the assets that the user worked with during the current session.	All	All
Marketing (default for CS-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	admin
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	admin

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 113.

Burlington Financial Extension (for CS-Engage)

The CS-Engage extension to the Burlington Financial sample site illustrates the CS-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. Additionally, it illustrates Analysis Connector.

Users, ACLS, and Roles for Burlington Financial Extension

The CS-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 CS-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

Username	Password	ACLs	Roles
user_checker	user	Browser, ElementReader, PageReader, RemoteClient, UserReader, Visitor, xceleditor	Checker
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
admin	xceladmin	Browser, ElementEditor, PageEditor, RemoteClient, TableEditor, UserEditor, UserReader, Visitor, VisitorAdmin, xceladmin, xceleditor	GeneralAdmin, SiteAdmin, WorkflowAdmin

Asset Type Configuration for Burlington Financial Extension

The CS-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 *CSEE Developer's Guide* and then use Content Server Explorer to examine the database tables for these asset types.

The following CS-Engage and CS-Direct Advantage 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 CS-Engage extension to the Burlington Financial site adds no sources.

Start Menu Configuration for Burlington Financial Extension

The CS-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

Name	Asset Type	Start Menu Type	Roles
New History Attribute	history attribute	New	Designer
Find History Attribute	history attribute	Search	Designer
New History Defintion	history definition	New	Designer
Find History Defintion	History Defintion	Search	Designer
New PDF	PDF	New	Analyst, Expert, Designer, Marketer
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

Name	Asset Type	Start Menu Type	Roles
New Recommendation	recommenda-tion	New	Analyst, Expert, Designer, Marketer
Find Recommendation	recommenda-tion	Search	Analyst, Expert, Designer, Marketer
New Segment	segment	New	Analyst, Expert, Designer, Marketer
Find Segment	segment	Search	Analyst, Expert, Designer, Marketer
New Vistor Attribute	visitor attribute	New	Designer
Find Visitor Attribute	visitor attribute	Search	Designer

Tree Tab Configuration for Burlington Financial Extension

The CS-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	admin
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

Tab	Description	Roles Assigned	Users with Access
History (default)	Displays the assets that the user worked with during the current session.	All	All
Marketing (default with CS-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	admin
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	admin

Workflow for Burlington Financial Extension

The CS-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.

Section 4

System Configuration

This section describes the configuration options that affect all the sites on your CSEE system.

It contains the following chapters:

- Chapter 7, “Configuring the User Interfaces”
- Chapter 8, “Managing the CSEE Publishing System”
- Chapter 9, “Revision Tracking”
- Chapter 10, “Search Engines”
- Chapter 11, “Properties and Property Files”
- Chapter 12, “System Information: Content Server Database”

Chapter 7

Configuring the User Interfaces

There are several CSEE user interface features that must be configured before content providers can use them.

Some of these features are alternate interfaces to the Content Server interface: Content Server Desktop, Content Server DocLink, and the InSite Editor. Others change the Content Server interface: locale settings and Ektron's eWebEditPro. And one changes the behavior of the preview function.

This chapter contains the following sections:

- Setting the Locale for the Content Server Interface
- CS-Desktop
- CS-DocLink
- InSite Editor
- eWebEditPro
- Maintaining Separate Browser Sessions for Preview

Setting the Locale for the Content Server Interface

If your organization purchased and installed one of the CSEE language packs, you must configure a default locale for your CSEE system to use.

There are several options and properties that you use to set the locale:

- The **Locale** item on the **Admin** tab sets the default locale for the Content Server interface. 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 CSEE 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 a User Profile" on page 41 for information about how to set a user's locale preference.

CS-Direct 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 CSEE, CS-Direct obtains the value for the `locale` variable as follows:

- If the user's user profile has a value for the locale user attribute, CS-Direct obtains and uses it.
- If the user profile does not specify a preference, CS-Direct 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 CS-Direct cannot match that user's browser locale, CS-Direct uses the default locale that is set for the system.

To set the system default locale, complete the following steps:

1. On the **Admin** tab, select **Locale.**

2. In the **Locale Manager** form, click the **Edit** button next to the locale that you want to select.
3. In the **Edit Locale** form, select the **Default** option.
4. Click **Save**.

System Locale Properties

The properties in the following list should have been configured during the installation of the language pack on your CSEE 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.

See “Using the Property Editor” on page 276 for instructions on how to start and use the Property Editor to verify or modify property values.

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 CSEE 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-ASCI text, this value must be set to 4 or lower.
xcelerate.charset	futuretense_xcel.ini	Determines the HTTP header character set used for all CS-Direct 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 CS-Direct form can be displayed in any of the languages that you have installed on your CSEE system, items such as field names and asset type names can be displayed in one language only.

Following is a list of items in CSEE 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 CSEE

system. However, there are several core columns/fields that CS-Direct 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 CS-Direct displays them as fields in the Content Server interface, it uses the language that is set as the system default language.

For descriptions of the default columns for asset types, see either the “Data Design” section in the *CSEE Developers Guide* or the *CSEE Database Schema*.

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 Content Server interface.

When this column holds non-ASCI text, the value for this property should be set to 500.

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 that 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, CS-Direct 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.

Format Preserved	Word Content Marked to Be Saved in the Blob Attribute or Upload Field	How the Word Content Is Stored in the Database
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, you complete the following steps:

- 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.

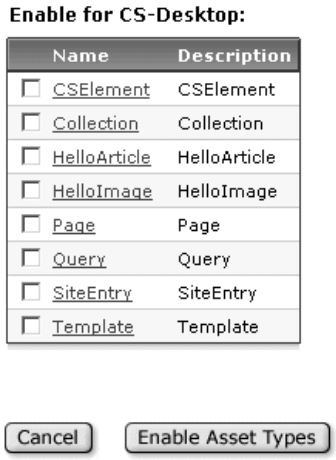
Enabling Asset Types for CS-Desktop

You enable an asset type for CS-Desktop by adding it to those that are selected for CS-Desktop.

Complete the following steps:

1. In the Content Server interface, do one of the following:
   - Select **Admin**, expand the **Sites** option, and expand the site that you are configuring CS-Desktop for.
   - 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. Select **CS-Desktop** > **Enable CS-Desktop**.

The **Enable Asset Types** form appears. It displays all the asset types that exist on this system that have not yet been enabled for CS-Desktop for this site:



3. Select the asset types that you want to enable for this site.
4. Click **Enable Asset Types**.
5. Continue to step 4 in “Specifying Which Fields Are on the Content Server Toolbar” on page 173.

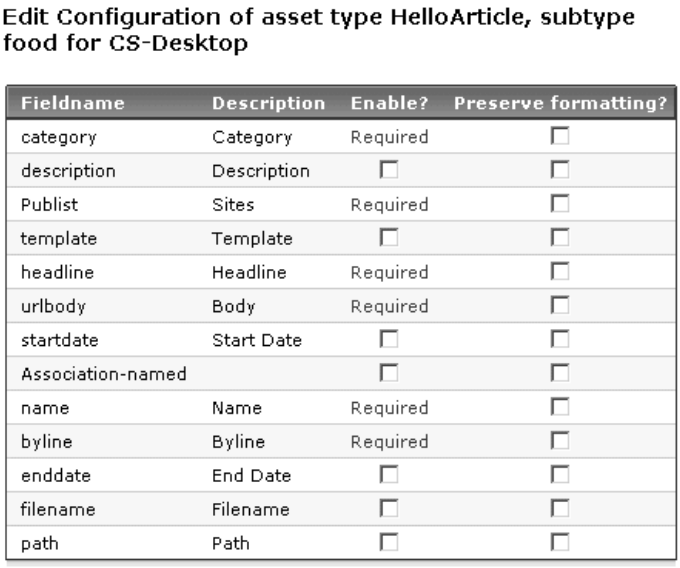
Specifying Which Fields Are on the Content Server Toolbar

Before you begin this procedure, be sure that you have enabled the asset type that you want to configure for CS-Desktop. If you have not yet enabled the asset type, see “Enabling Asset Types for CS-Desktop” on page 172.

To specify which of an asset type’s fields will be displayed on the Content Server toolbar within the Word interface, complete the following steps:

1. In the Content Server interface, do one of the following:
   - Select **Admin**, expand the **Sites** option, and expand the site you are configuring CS-Desktop for.
   - 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 the **CS-Desktop** option.
3. Expand the asset type that you want to configure.
4. Select the appropriate subtype (basic asset types) or definition (flex and flex parent asset types), and then click **Edit Configuration**.

CS-Direct displays a form similar to this one:



Edit Configuration of asset type HelloArticle, subtype food for CS-Desktop

Fieldname	Description	Enable?	Preserve formatting?
category	Category	Required	☐
description	Description	☐	☐
Publist	Sites	Required	☐
template	Template	☐	☐
headline	Headline	Required	☐
urlbody	Body	Required	☐
startdate	Start Date	☐	☐
Association-named		☐	☐
name	Name	Required	☐
byline	Byline	Required	☐
enddate	End Date	☐	☐
filename	Filename	☐	☐
path	Path	☐	☐

5. In the **Edit Configuration** form, select the **Enable** option for each field that you want to appear on the Content Server toolbar. As a general rule, select only text-entry fields.

   Note that any required fields are already selected.

6. (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.

7. Click **Save**.
8. Repeat steps 4 through 7 for each subtype or definition for this asset type.
9. Repeat steps 3 through 8 for each asset type.

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.

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab.
2. Expand **Asset Types** and then expand the asset type that you are interested in.
3. Select **Start Menu > Add New Start Menu**.
4. Click in the **Name** field and enter the name of the item. This is the name that is displayed in the list of asset types that are available from the Content Server toolbar.
5. Click in the **Description** field and enter a short description for the start menu item.
6. 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.

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. (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**.

      The system saves the workflow information and displays the **Start Menu** form again.
10. In the **Start Menu** form, click **Save**.
11. Repeat steps 2 through 10 for each of the Word asset types.

User Accounts and CS-Desktop

Like the content providers who use the Content Server interface, 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 CS-Engage installed)

For information about creating user accounts, see “Users and Required ACLs” on page 39 and “Creating a New User” on page 40.

Give the CS-Desktop users the following information:

- Their user name/password combination
- The appropriate URL for logging in to your CSEE management system

They need both pieces of information to log in to Content Server through CS-Desktop. Your Content Server URL uses one of the following conventions:

iPlanet Application Server (iAS)

`http://<server>:<port>/NASApps/cs`

where <*server*> is the name of the server that hosts your CSEE management system and <*Port*> is the port number.

All Other Application Servers, Including Sun ONE

`http://<server>:<port>/servlet`

where <*server*> is the name of the server that hosts your CSEE management system and <*Port*> is the port number.

Installing the CS-Desktop Client Application

Each of the content providers who will use the CS-Desktop feature must run the `csdesktop.exe` executable file on their client systems.

To install the client application, complete the following steps:

1. On a PC that has a supported operating system and a supported version of Microsoft Word, set the encoding either to UTF-8 or to another type of encoding that supports an even larger character set than does UTF-8.
2. Open Internet Explorer and navigate to the following URL:

   `http://yourserver:itsPortNumber/Xcelerate/DownloadPage.html`

   where *yourserver* is the name of the server that hosts your CSEE management system and *itsPortNumber* is the port number.
3. On the **Download** page, select **CS-Desktop**, and then click **Download**. Note that if there are any language packs installed on your CSEE system, the **Download** page presents CS-Desktop options in the languages that are installed on the system.
4. In the **File Download** dialog box, click **Open**.

   The Winzip application starts.
5. Click **I Agree**.

   Winzip unzips the zip file.
6. In the Winzip window, double-click on `csdesktop.exe`.

   The installation program begins.
7. Follow the instructions that appear on the screen.

Specifying Locale for the Client Application

The Download page is generated dynamically based on the language packs that are installed on your CSEE system. If your CSEE 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 CS-Direct. 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:

     iPlanet Application Server (iAS)

     `http://<server>:<port>/NASApps/cs`

     where *<server>* is the name of the server that hosts your CSEE management system and *<Port>* is the port number.

     All Other Application Servers, including Sun ONE:

     `http://<server>:<port>/servlet`

     where *<server>* is the name of the server that hosts your CSEE management system and *<Port>* is the port number.
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 6 through 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. On the button bar, select **Content Server** > **Save to Content Server**.
11. Open Internet Explorer and log in to the Content Server 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 interface.

Content providers can drag and drop Word files, Excel spreadsheets, graphic files, and so on—that is, single binary files—into the Content Server database by dropping the file on 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 asset types, you do the following:

- 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.

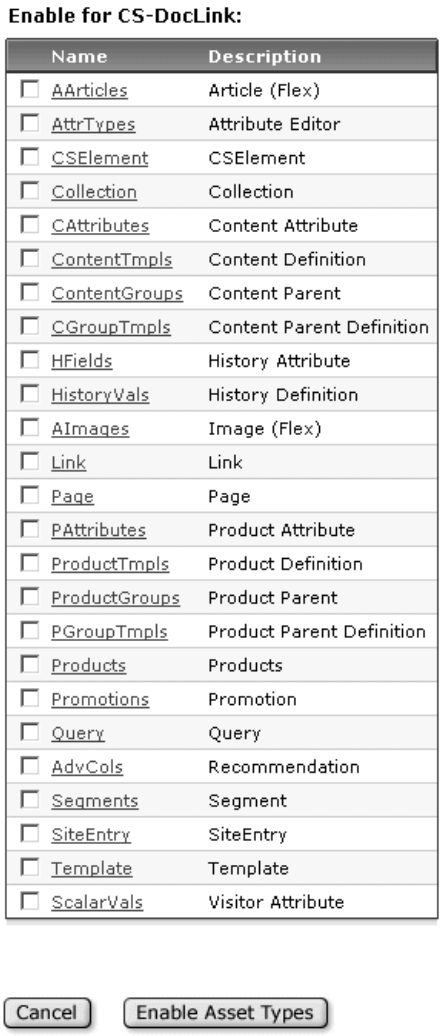
Enabling Asset Types for CS-DocLink

You enable an asset type for CS-DocLink by adding it to those that are selected for CS-DocLink.

Complete the following steps:

1. In the Content Server interface, do one of the following:
   - Select **Admin**, expand the **Sites** option, and expand the site that you are configuring CS-DocLink for.
   - 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. Select **CS-DocLink > Enable CS-DocLink**.

   The **Enable Asset Types** form appears. It displays all the asset types that exist on this system that have not yet been enabled for CS-DocLink for this site:



3. Select the asset types that you want to enable for this site.
4. Click **Enable Asset Types**.

   The asset types appear on the **Admin** tab under the CS-DocLink node for this site.
5. Continue to step 3 in "Specifying Which Fields Are on the Content Server Toolbar" on page 173.

Specifying Which Fields Are Available in CS-DocLink

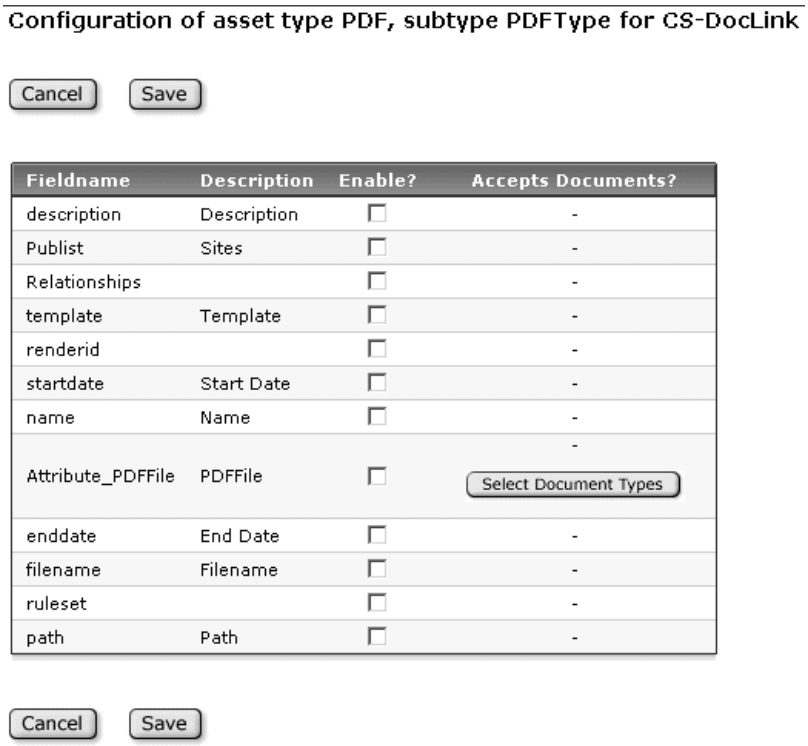
Before you begin this procedure, be sure that the following conditions are true:

- 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 179.
- 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, complete the following steps:

1. In the Content Server interface, do one of the following:
   - Select **Admin**, expand the **Sites** option, and expand the site that you are configuring CS-DocLink for.
   - 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 the **CS-DocLink option.**
3. **Under the CS-DocLink** option, expand the asset type that you want to configure and then select the appropriate definition.
4. Click **Edit Configuration**.

   CS-Direct displays a form similar to this one:



5. In the **Edit Configuration** 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.

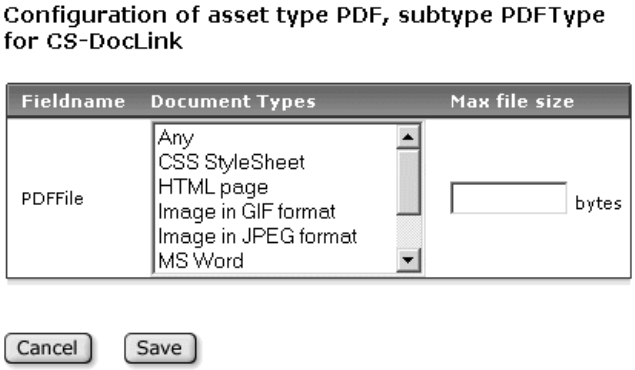
6. Next to the upload field (flex attribute) that you want to specify as the field that documents are to be stored in, click the **Select Document Types** button.

   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 *CSEE Developer's Guide* and talk to your developers.

   CS-Direct displays a form similar to this one:

7. 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 interface.

8. 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.
9. Click **Save**.
10. In the **Configuration** form, click **Save** again.
11. Repeat steps 3 through 10 for each definition for this asset type.
12. Repeat steps 3 through 11 for each asset type that 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.

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab.
2. Expand **Asset Types** and then expand the asset type that you are interested in.
3. Select **Start Menu** > **Add New Start Menu**.
4. Click in the **Name** field and enter the name of the item.
5. Click in the **Description** field and enter a short description for the start menu item.
6. 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.

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. (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**.

      The system saves the workflow information and displays the **Start Menu** form again.
10. In the **Start Menu** form, click **Save**.
11. Repeat steps 2 through 10 for each of the Word asset types.

Users and CS-DocLink

Like the content providers who use the Content Server interface, the content providers who use the CS-DocLink application must have Content Server user accounts as well. User accounts for the folks who use CS-DocLink must have the following ACLs:

- Browser
- ElementReader

- PageReader
- RemoteClient
- UserReader
- xceleditor

For information about creating user accounts, see "Users and Required ACLs" on page 39 and "Creating a New User" on page 40.

Give the CS-DocLink users the following information:

- Their user name/password combination
- The appropriate URL for logging in to the Content Server database

They need both pieces of information to log in to Content Server through CS-DocLink. Your CS-DocLink URL uses one of the following conventions:

iPlanet Application Server (iAS)

`http://<server>:<port>/NASApps/cs/ContentServer?pagename=OpenMarket/CSClient/InitialURLs`

where `<server>` is the name of the server that hosts your CSEE management system and `<Port>` is the port number.

All Other Application Servers, including Sun ONE:

`http://<server>:<port>/servlet/ContentServer?pagename=OpenMarket/CSClient/InitialURLs`

where `<server>` is the name of the server that hosts your CSEE management system and `<Port>` is the port number.

Installing the CS-DocLink Client Application

Each of the content providers who will use the CS-DocLink feature must download the application to their client systems.

To install the client application, complete the following steps:

1. Open Internet Explorer and navigate to the following URL:

   `http://yourserver:itsPortNumber/Xcelerate/DownloadPage.html`

   where *yourserver* is the name of the server that hosts your CSEE management system and *itsPortNumber* is the port number.
2. On the **Download** page, select **CS-DocLink**, and then click **Download**.
3. In the **File Download** dialog box, click **Open**.

   The Winzip application starts.
4. Click **I Agree**.

   Winzip unzips the zip file.
5. In the Winzip window, double-click on `setup.exe`.

   The installation program begins.
6. Follow the instructions that appear on the screen.

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 CS-Direct.

Complete the following steps:

1. Open Microsoft Explorer.
2. Double-click on the CS-DocLink node.
3. In the **Login** dialog box, enter the following information:
   - The CS-DocLink URL. See "Users and CS-DocLink" on page 183 for the appropriate URL for your system.
   - The username and password of a user who has the ACLs listed in "Users and CS-DocLink" on page 183.
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 Internet Explorer and log in to the Content Server interface as a user who has access to the type of asset that you just created.
7. Select the site that you created the asset for and then click **Search**.
8. Find the asset that you created through CS-DocLink.
9. Click **Inspect**.
10. 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 Editor

The InSite Editor is a CS-Direct 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 interface.

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 Editor control panel appears.

The control panel provides a subset of the CS-Direct features and functions. Your InSite Editor users interact with the CSEE system by using the control panel instead of the Content Server interface.

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 CSEE management system.
- Your developers code templates that invoke the InSite Editor 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 Editor control panel on the CSEE 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 CSEE users who need to use the InSite Editor.

Note

The InSite Editor 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 Editor cannot display it.

Enabling the InSite Editor

Complete the following steps:

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 Editor, verify that it is installed and confirm that the `xcelerate.ewebeditpro` property is set to `true`.

   For information about eWebEditPro, see "eWebEditPro" on page 187.

   Note that to use eWebEditPro within the InSite Editor, your developers must use the `EWEBEDITPRO` parameter with the `INSITE.EDIT` tags on your templates. For more information, see the *CSEE Developer's Guide*.
4. Select **File** > **Save**.
5. Select **File** > **Exit**.

After you enable the InSite Editor, CS-Direct displays the InSite Editor control panel whenever a CSEE 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 Editor help file.

User Accounts and the InSite Editor

Like the content providers who use the Content Server interface, the content providers who use the InSite Editor control panel must have Content Server user accounts as well. User accounts for the content providers who use the InSite Editor must have the following ACLs:

- Browser
- ElementReader
- PageReader
- RemoteClient
- UserReader
- Visitor (if you are using CS-Engage)
- xceleditor

For information about creating user accounts, see “Users and Required ACLs” on page 39 and “Creating a New User” on page 40.

Be sure that you give the InSite Editor 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 Editor

eWebEditPro

eWebEditPro is a third-party HTML editor from Ektron, Inc. that the CS-Direct and CS-Direct Advantage products support. Your developers can create basic assets whose text-entry fields use eWebEditPro as the input mechanism for the field. They can also create attribute editors for flex attributes that use eWebEditPro as the input mechanism.

Three versions of eWebEditPro are supported: v 3.0.0.7, v 4.0.0.14 (both used strictly as HTML editors), and eWebEditPro+XML. Only one version of eWebEditPro can be used by each Content Management installation. To obtain eWebEditPro or eWebEditPro+XML, contact your FatWire sales representative.

Configuring eWebEditPro

Installation and configuration instructions are included in the eWebEditPro release notes that accompany the version that you purchase through FatWire.

User Accounts and eWebEditPro

You do not need to add any extra ACLs to user accounts for CSEE users who work with eWebEditPro. If a user can log in to the Content Server interface, he or she can enter data in an asset field that has eWebEditPro as the input mechanism.

Maintaining Separate Browser Sessions for Preview

If the online site that you are serving from your CSEE 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 CSEE 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 CSEE 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 CSEE 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 CSEE 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 CSEE, 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 SatelliteServer servlet rather than the ContentServer servlet.

Start the Property Editor, open the `futuretense_xcel.ini` file, select the **xcelerate** tab, and then do one or both of the following:

- Set the `xcelerate.previewhost` property as follows:

  For most application servers, including Sun ONE:

  `http://servername:port/servlet/`

  For iPlanet Application Server (iAS) 6:

  `http://servername:port/NASApps/cs/`
- Change the value of the `xcelerate.previewservlet` property from `ContentServer` to `Satellite`

Chapter 8

Managing the CSEE Publishing System

You make content available to the visitors of your online site by moving it to your delivery system. Moving content from one CSEE system to another is called **publishing**.

Content Server provides two publishing APIs, called Export and Mirror. The Export API renders Content Server pages into files. The Mirror API copies rows from tables in one Content Server database to the corresponding tables in another Content Server database.

CS-Direct delivers a publishing system that uses these Content Server APIs. While the actual rendering of files or mirroring of rows is completed by the API, the CS-Direct publishing system includes an approval system that determines which content gets published, a schedule that determines when publishing occurs, and the ability to configure site-specific destinations. In addition, in dynamic CSEE systems, the publishing system communicates with the CacheManager on the destination server.

This chapter describes the CS-Direct publishing system, which is also used by the other CS content applications. It contains the following sections:

- Overview
- The Export to Disk Delivery Type
- The Mirror to Server Delivery Type
- The Export Assets to XML Delivery Type
- Configuring the Publishing Process for Your CSEE System
- Additional Publishing Procedures
- Troubleshooting

For information about the Content Server Export and Mirror APIs, see the *CSEE Java API Reference* and the *CSEE Developer's Tag Reference*.

Overview

The CS-Direct publishing system interacts with several underlying systems and APIs before, during, and after a publishing session.

Before CS-Direct can publish assets to a destination system, the following information must be determined:

- Which **delivery type** should CS-Direct use: Export to Disk, Export Assets to XML, or Mirror to Server?

  As an administrator, you specify the delivery type for the destination when you configure the publishing destination.

- Where should CS-Direct publish to? That is, what is the destination for the publishing session?

  As an administrator, you configure the **publishing destinations** for your system. When content providers mark their assets "approved," they approve them for a specific destination.

- Which assets are ready to be published to the current destination?

  The **approval** system manages this information. When an asset is ready to be published, a content provider marks it as "approved" for a specific destination. The publishing system checks with the approval system and then only the assets that are marked as approved are published, which protects the site from having any broken links. Connected parts of the site are held until they are all ready to be published.

- When should the assets 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.

During a publishing session, CS-Direct writes information about that session to log files. Because the publishing process runs in the background, you can use your browser to complete other administrative tasks while the session runs. And you can monitor the session through the **Publish Console**, which shows both the publishing history and the status of any publishing sessions that are currently running.

When the publishing session uses the Mirror to Server delivery type, the publishing system communicates and cooperates with the **CacheManager** on the delivery system. The CacheManager is a Content Server servlet that manages the page cache on a system. It ensures that pagelets or pages that refer to the assets that will be mirrored are cached. Then, after the publishing session concludes, CacheManager generates those pages again to display the updated content, and caches the new pages and pagelets.

Delivery Types

The term “delivery type” means the publishing method. There are three delivery types and they use either the Export API or the Mirror API. Which delivery type you use for your CSEE system depends on how you plan to deliver the content on your delivery system (statically or dynamically) or whether you are delivering your content to some other non-CSEE system:

- On a **static** CSEE delivery system, a web server delivers static files to your site visitors. You use the **Export to Disk** delivery type to create the files. During an export publishing process, CS-Direct instructs Content Server to render all approved assets according to their templates and then saves those files. You, the administrator, must then use FTP or another file transfer protocol to move those files to the web server that is your delivery system.
- On a **dynamic** CSEE delivery system, Content Server generates (renders) pages upon request. Content Server makes dynamic content decisions based on the request. You use the **Mirror to Server** delivery type to copy the rows for your approved assets from the Content Server database on the management system to the Content Server database on the delivery system.
- The **Export Assets to XML** delivery type is a data transformation method. Rather than creating pages that are ready to be displayed by a web server, it uses the Export API to create one XML file for each approved asset. You use this delivery type when your end goal is to publish assets to another (non-CSEE) delivery system or database and that system needs your content to be formatted in XML.

These sections describe each delivery type in detail:

- “The Export to Disk Delivery Type” on page 199.
- “The Mirror to Server Delivery Type” on page 208.
- “The Export Assets to XML Delivery Type” on page 213.

Publishing Destinations

A publishing destination defines the location where your content is published to. A destination is a named object that contains the following information:

- A delivery type (that is, Export to Disk, Mirror to Server, or Export Assets to XML)
- A location
  - For Export to Disk and Export Assets to XML, the location is the **root file directory** that the resulting HTML or XML files are saved to 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 Mirror to Server, 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.

    Note that before you can publish to a mirror destination, you must initialize it. For more information, see “The Initialize Mirror Destination Feature” on page 212.
- Various **publishing arguments**, depending on which delivery type you are using. For example, if you are using Export to Disk you could specify a URL prefix for your internal links (HREFs).

  - For information about Export to Disk publishing arguments, see "Publishing Arguments for Export to Disk" on page 201.
  - For information about Mirror to Server publishing arguments, see "Publishing Arguments for Mirror to Server" on page 211.
  - For information about Export Assets to XML publishing arguments, see "Publishing Arguments for Export Assets to XML" on page 215.
- The CSEE sites whose assets can be published to the destination.

You can configure more than one publishing destination for your CSEE 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**, which means the Content Server database that serves as the source for a publishing session. Because you can mirror assets from any CSEE system to any other CSEE system, the source is not always the CSEE management system.
- **Destination**, which means 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 2 (Export to Disk): Configure an Export Destination" on page 218 and "Step 4 (Mirror to Server): Configure a Mirror Destination" on page 224.

Publishing and the Approval Process

The publishing system and the approval system work very closely together. The approval system determines the answers to the following questions:

- Which assets are ready to be published (that is, are marked as approved)?
- Are there any assets that must accompany those approved assets or that must exist on the destination to ensure that all the links on the site will work correctly?

When an asset is ready to be published, a content provider marks it as approved for a specific destination. (For information about how you approve assets, see the chapter on publishing in the *CSEE User's Guide*.) CS-Direct then verifies that the asset is ready to be published by calculating the list of assets upon which approval depends.

For example, the Burlington Financial sample site has article assets with associated imagefiles. An approved article with an unapproved associated imagefile cannot be published until the imagefile is also approved.

It is typical to build an approval step into your workflow processes. For information about workflow, see Chapter 5, "Workflow."

Approval Dependencies

The approval status of an asset is determined by its dependency relationships, which include the approval status of all asset items associated with a particular asset item, as well as the dependency relationships of those associated items.

How are approval dependencies calculated? That depends on the publishing method:

- For **Export to Disk**, the approval system renders the asset according to the template that is assigned to it when the asset is approved, or, if there is one, according to the

default approval template for assets of that type. The tags in the template code set approval dependencies that determine the appropriate dependents for the approved asset. The dependent assets must be in an appropriate approval state before the current asset can be published.

- For **Mirror to Server** or **Export Assets to XML**, the approval process examines the data relationships between asset types. Basic assets have associations. Flex assets have family relationships. Both of these relationships create approval dependencies for these publishing methods. For example, if you approve a flex asset, it will be held from a publishing session unless its parent assets are in an appropriate approval state.

When the approval system calculates approval dependencies, it further categorizes those dependencies by the following types:

- **Exists** – some version of the dependent asset must exist on the destination so that links to it will work.
- **Exact** – the version of the dependent asset must exactly match the version of it that was approved. That version must either already exist on the destination or be approved for the destination so that it can be published with the asset that is being evaluated. The version of an asset is based on the timestamp in the `updateddate` column for that asset.

An asset can be published only if it meets all specified dependencies: that is, all associated assets are also in the correct approval state (either approved or previously published, as necessary). If not, the approved asset is held by CS-Direct until its dependent assets have been approved.

Note

When someone deletes an asset, its status is changed to Void and it is automatically approved for any destination that it was ever published to. Then, during the next publishing sessions for those destinations, that asset is published and its status becomes Void on those destinations.

Held Assets

A **held** asset is one that cannot be published because its approval dependencies are not yet met. When its approval dependencies are met, it will be published.

An asset will be **held** from a publishing session when any of the following conditions are true:

- It is not approved.
- It was approved, but someone edited it again and it has not yet been re-approved.
- It has a dependent asset with an exact dependency that is not approved.
- It has a dependent asset with an exists dependency and that dependent asset has never been published.
- It has a dependent asset with an exact dependency, the dependent asset has been published, but it has since been edited and is not yet approved.
- It is an exact dependent of another asset that was previously approved, but has since been edited and is not yet re-approved.
- It is checked out (revision tracking).

The approval system calculates dependencies for basic assets differently when the delivery type is Export to Disk than when it is Mirror to Server and Export Assets to XML.

Calculating Dependencies for Export to Disk

When an asset is approved to a destination that uses **Export to Disk**, the approval system determines the compositional dependencies for that asset by evaluating the code in the template that is assigned to the asset. It performs a "test-render."

- For **basic** assets, tags such as `ASSET.LOAD`, `RENDER.LOGDEP`, `ASSET.SITEPARENT`, and `RENDER.GETPAGEURL` mark dependencies between assets and the pages that they are rendered on.
- For **flex** assets, several tags in the `ASSETSET` family of tags, `RENDER.LOGDEP`, and the `RENDER.GETPAGEURL` tags mark dependencies between assets and the pages that they are rendered on.

When the delivery type is Export to Disk, the dependencies are determined as the code in the template is evaluated. If a default template has been assigned for 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 specified for the asset.

Your developers create default templates for asset types for the following reason: when Export to Disk actually publishes the 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 is used for 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 calculating approvals.

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 244.

For more information about how dependencies are created between assets when the delivery type is Export to Disk, see the chapters on template design in the *CSEE Developer's Guide*.

Calculating Dependencies for Mirror to Server and Export Assets to XML

When an asset is approved to a destination that uses **Mirror to Server** or **Export Assets to XML**, the approval system calculates the dependencies for that asset based on the relationships to other assets that are described in the database. That is, approval is based on data dependencies.

Basic Asset Types

For basic assets, the approval system follows these dependency rules:

- Either an exists or an exact dependency on dependent assets referenced through named asset associations, depending on how your asset associations are designed
- An exists dependency on dependent assets referenced through unnamed asset associations
- For page assets, an exists dependency for page assets that are lower than it in the hierarchy of the site plan (which is reflected in the `SitePlanTree` table)
- An exists dependency on assets referenced by an embedded link

- An exact dependency on assets referenced as an embedded pagelet

The following table summarizes this information:

Relationship to Approved Asset	Exists	Exact
Named association	X, by default, unless specifically configured to be exact	
Unnamed association	X	
For page asset only: another page asset at a lower level in site plan	X	
Embedded link	X	
Embedded pagelet		X

For information about embedded links and embedded pagelets, see the *CSEE Developer's Guide*.

CSElement and SiteEntry Assets

The root element for a SiteEntry asset is represented by a CSElement asset. There is an exists dependency on the CSElement asset that a SiteEntry asset refers to.

Flex Families

For the flex asset family member and the flex parent family member, the approval system follows these dependency rules:

- An exact dependency on its flex definition (or parent definition, for a parent asset)
- An exact dependency on its attributes
- An exists dependency on its parents
- An exists dependency on its template

The following rules are used for flex definitions and flex parent definitions:

- An exists dependency on its parent definitions
- An exists dependency on its attributes

The following rules are used for flex attributes:

- An exists dependency on its attribute editor (if one is assigned).
- 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.

The following table summarizes this information:

Approved Asset	Dependency Type of Dependent Assets					
	Flex Parent Definition	Flex Parent	Flex Definition	Flex Attributes	Flex Filter	Attribute Editor
Flex Parent Definition	exists			exists		
Flex Definition	exists			exists		
Flex Parent	exact	exists		exists		
Flex Asset		exists	exact	exists		
Flex Attribute					exists	exists

CS-Engage Visitor Data Assets

The approval system uses the following rules to calculate data dependencies for the CS-Engage visitor data asset types:

- 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)

The following table summarizes this information:

Approved Asset	Dependency Type of Dependent Assets				
	History Definition	History Attribute	Visitor Attribute	Recom-mendation	Related Flex Asset
History definition		exact			
Segment	exists		exists		
Recommendation					
Promotion				exists	
Flex Asset					exists

Approving Multiple Assets

The approval system provides a feature called **Approve Multiple Assets**. You use it to approve multiple assets in the following situations:

- You are upgrading from one version of CSEE to another and you have previously published assets that you want to be marked as approved.
- You add a new publishing destination and want to publish assets to it that you know are ready because they are approved for other destinations.
- The default approval template for an export destination is changed.

This feature is described in “Approving Multiple Assets” on page 237.

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 *CSEE Developer's Guide*.

The Publishing Schedule (Publish Events)

A publishing session (no matter what the delivery type) runs as a background, batch process called a **publish event**. As the administrator, you configure the schedule for the publishing events for your CSEE system.

You can schedule publishing events to occur daily, weekly, hourly, every 15 minutes, or in various combinations of these time increments.

Because each publishing session is a background event, there can be only one publishing session for the same destination at any given time. If a publishing session to a specific destination is still ongoing 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.

Note that you should 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 you publish from the management system to the delivery system. You should never run a publishing session from the development system to the management system while the management system is publishing to the delivery system.

Additionally, you can publish to more than one destination at the same time. You just set up events for the destinations and select the same time for them.

Be aware that the reverse is not true: that is, **you cannot have more than one source publish to the same destination**. If you do, there will be problems with data integrity as well as publishing errors.

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 CSEE system.

Note

The batch user is not the same as the mirror user. For more information, see “Users and Mirror to Server” on page 211.

What Happens During the Publishing Session?

When the publishing system begins publishing approved assets, CS-Direct 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 delivery type (Export to Disk or Mirror to Server), 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 delivery type in use. For information about what the Export to Disk, Export Assets to XML, and Mirror to Server delivery types do with the list of approved assets, see the following sections:

- "The Export to Disk Delivery Type" on page 199.
- "The Mirror to Server Delivery Type" on page 208.
- "The Export Assets to XML Delivery Type" on page 213.

When the publishing session concludes, the publishing system notifies the approval system of which assets were published and then it updates the page cache.

Note

The CS-Direct publishing system publishes assets that are held in the Content Server database.

If you are storing files directly on your web server or your site designers use links from assets to files that are not assets, you must implement an additional process to move those files to your web server and to update them when necessary.

Obtaining Information About a Publishing Session

CS-Direct 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.

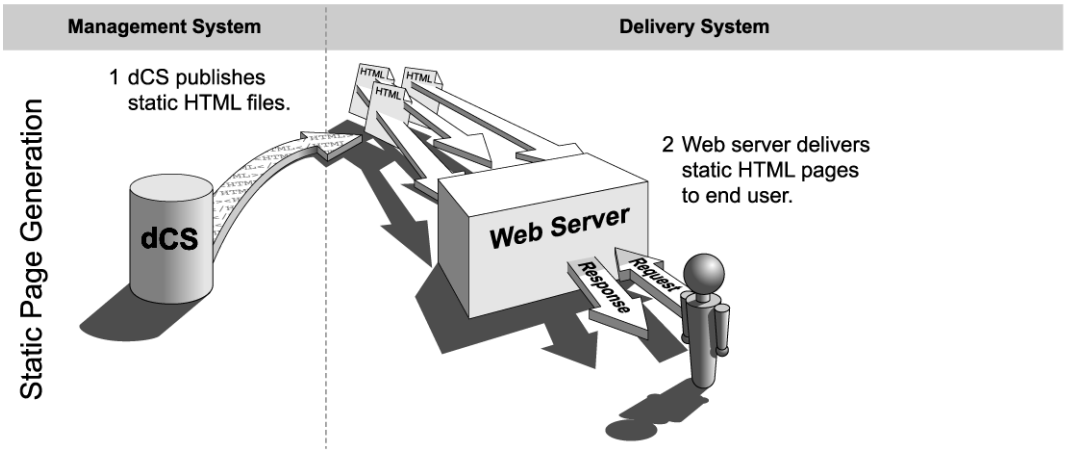
See also "Troubleshooting" on page 249.

The Export to Disk Delivery Type

In extremely simple terms, Content Server renders a page as follows:

1. A page name is submitted to Content Server (through HTTP or another protocol).
2. Content Server then locates the page in the SiteCatalog table, invokes the root element, and renders the page.

On a dynamic delivery system or when a user of the management system previews an asset, Content Server renders a page and posts it through HTTP to the visitor's or the user's browser. However, when the **Export** API is involved, Content Server **renders the page to a file**. You then move those files to the web server that is hosting your online site.

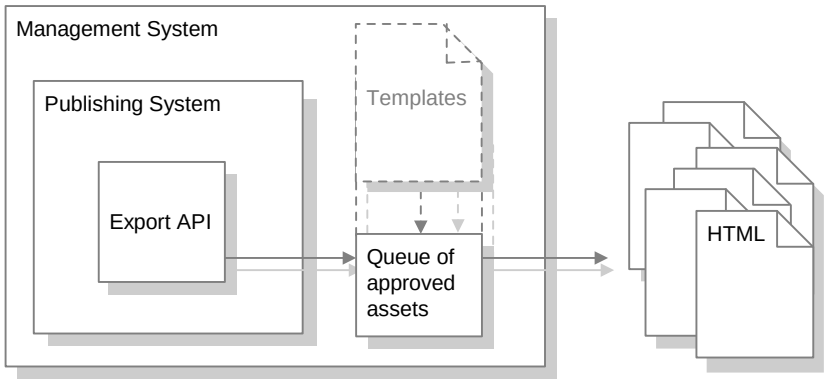


When you use the Export to Disk delivery type, typically you set up a quality assurance process to test the generated files before you move the files—either through FTP or another file transfer protocol—to the web server of your delivery system.

How Export to Disk Works

When you use the Export to Disk delivery type, the approval system, the publishing schedule, and your destination configurations all contribute to the process of exporting your approved assets into static files.

When an Export to Disk 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. CS-Direct 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:

   - An asset that has been designated as an **export starting point**. (See "Export Starting Points" on page 208.)

     There must be a starting point for the site and it 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, CS-Direct determines whether it has been approved:

   - If the asset that is the starting point has not been approved, the publishing session ends.
   - If the asset is approved, CS-Direct determines the page name of the template assigned to the asset.

4. CS-Direct 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 (“Publishing Arguments for Export to Disk” on page 201.)

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, CS-Direct 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 to step 3 in this description.
8. When the publishing session is successful, CS-Direct concludes the session by writing information about the published references to the `PubKey` and `PublishedAssets` tables.
9. CS-Direct 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.

Publishing Arguments for Export to Disk

When you configure publishing destinations that use the Export to Disk delivery type for your CSEE system, you must provide certain values that CS-Direct needs in order to export publish your assets correctly. You provide these values with **publishing arguments**.

Some export publishing arguments are used to name the files and some are used to generate the links (HREFs) within the files.

The following table lists the publishing arguments for Export to Disk:

Publishing Argument	Description
`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.

Publishing Argument	Description
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: value of cs.pgexportfolder/value of 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	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**.
SIMPLENAME	Optional. If set to true, this argument overrides the normal file-naming conventions. (For a description of the file-naming conventions, see "File and Directory Naming Conventions" on page 203.) When set to true, this argument instructs CS-Direct 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, CS-Direct uses the asset's ID, instead. 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. 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.

Publishing Argument	Description
SIMPLEDIR	Optional. If set to `true`, this argument overrides the normal directory-naming convention. (For a description of the file-naming conventions, see "File and Directory Naming Conventions" on page 203.) When set to `true`, this argument instructs CS-Direct to put the rendered HTML file directly into the default export directory when there is no path information available from the parent asset. (Normally, CS-Direct 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.
OLDTEMPLATE	Required in special cases. This argument instructs CS-Direct 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	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.

File and Directory Naming Conventions

Exported files are named according to specific file-naming conventions and are saved to file directories that are named according to specific directory-naming conventions. This section describes those conventions and provides examples that you can use to determine how your files and directories will be named, based on the information that you provide for a destination.

CS-Direct needs several bits of information that it uses to name the files that Export to Disk generates from your approved assets and to determine where to store them. This section presents each of the variables that affect file and directory names and describes the file-naming conventions that CS-Direct follows.

Directory Names

This section describes the properties, publishing arguments, and fields that contribute to directory names and then presents the directory naming conventions and provides some examples.

Values Used in Directory Names

The following values are used to create the directory names for the exported files:

- `cs.pgexportfolder` property in the `futuretense.ini` file

  This property sets the base directory for all exported files. It is a required property whose value applies to all publishing destinations that use the Export to Disk publishing method.

  Files are exported either directly into this directory, or to subdirectories within this directory if you use the `DIR` argument (described, above, in "Publishing Arguments for Export to Disk" on page 201).

  The directory specified by this property, or by the combination of this property and the `DIR` argument is the **destination directory**.

  If your system is running on either a Solaris or an AIX operating system, you can use symbolic links to map a local subdirectory to a remotely mounted file system.

  Typically, this value is set to a testing location (file system) where you or others test the exported files rather than to a location directly on the delivery system.

- `DIR` publishing argument (see "Publishing Arguments for Export to Disk" on page 201)
- `SIMPLEDIR` publishing argument (see "Publishing Arguments for Export to Disk" on page 201)
- **Path** information

  In general, the directory name for a file 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, CS-Direct uses the ID of the parent page asset instead.

  However, note the following about path information:

  - The path information entered for a page asset is used for its children assets only. When a page asset is exported, it receives the path set for its parent page, not the one set for that page asset.
  - For assets that are not page assets, if path information is entered in the **Path** field for that asset, the path is used instead of the path 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.

Directory Naming Conventions

Directory names for both page assets and other assets are created by concatenating the values from the items in the preceding list in the following order:

```
value_of_cs.pgexportfolder/value_of_DIR/parent page's path
```

If an asset other than a page asset has path information of its own, CS-Direct uses that path rather than the path provided by the parent page asset. In such a case, the directory is created like this:

```
value_of_cs.pgexportfolder/value_of_DIR/asset's path
```

If **no** path information is available from the parent page or the asset itself, the directory is created like this:

```
value_of_cs.pgexportfolder/value_of_DIR/ID of parent page
```

If **no** path information is available from the parent page or the asset, and `SIMPLEDIR` is set to `true`, the directory is created like this:

```
value_of_cs.pgexportfolder/value_of_DIR
```

The following table provides examples of how directory names are created for **page assets** during Export to Disk:

Page Asset	Value of cs.pgexport folder	Value of DIR	ID of Parent Page	Path of Parent Page	Directory for Exported File
Home	/export				/export
Home	/export	/Japan			/export/Japan
World News	/export		998877665		/export/Japan/998877665
World News	/export		998877665	/news/	/export/news
World News	/export	/Japan	998877665	/news/	/export/Japan/news

The following table provides examples of how directory names are created for **assets other than page assets** during Export to Disk:

Asset	Value of cs.pgexport folder	Value of DIR	Path from Asset	ID of Parent Page	Path of Parent Page	Directory for Exported File
OilPrice	/export			997766554		/export/997766554
OilPrice	/export	/Japan		997766554		/export/Japan/ 997766554
OilPrice	/export	/Japan		997766554	/news/	/export/Japan/news
OilPrice	/export		/energy	997766554	/news/	/export/energy
OilPrice	/export	/Japan	/energy	997766554		/export/Japan/energy
OilPrice	/export	/Japan	/energy	997766554	/news/	/export/Japan/energy

Paths for Links Within the Files

The 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 Names

This section describes the arguments and other items that contribute to file names and then presents the file naming conventions and provides some examples.

Values Used in File Names

The following values are used to create file names for the exported files:

- The `SiteCatalog` page entry of template used to render the asset

  All template assets have `SiteCatalog` page entries. The name of the `SiteCatalog` page entry of the template that is used to render the asset is included in the name of the file generated for the asset (unless you use the `SIMPLENAME` publishing argument).

- Packed arguments passed in from root element of page entry

  If any packed arguments are passed in from the root element of the template that is used to render the asset, 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 *CSEE Developer's Guide*.

- **Filename** field

  If there is a value set in an asset's **Filename** field, the name of the file uses the value from that field in the name of the exported file. If there is no value in the **Filename** field, CS-Direct uses the ID of the asset in the asset.

  Note that 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 it is exported to that destination.

- `SUFFIX` publishing argument (see "Publishing Arguments for Export to Disk" on page 201)

- `SIMPLENAME` publishing argument (see "Publishing Arguments for Export to Disk" on page 201)

File Naming Conventions

File names for all assets are created by concatenating the values from the items in the preceding list in the following order:

```
pagename_packedargs(if any)_filename.SUFFIX
```

If there is **no filename** information provided, the file name is created like this:

```
pagename_packedargs(if any)_assetID.SUFFIX
```

If `SIMPLENAME` is set to `true`, the file name is created like this:

```
packedargs(if any)_filename.SUFFIX
```

If `SIMPLENAME` is set to `true` but there is no file name information provided, the file name is created like this:

```
packedargs(if any)_assetID.SUFFIX
```

Note

Be sure to use `SIMPLENAME` carefully. If your assets are rendered by more than one template, do not use `SIMPLENAME`.

Note the following syntax changes in the file names:

- Slash characters (/) in pagenames are converted to hyphens (-) in file names.
- Equals signs (=) in packed arguments are converted to hyphens (-) in file names.
- Ampersand characters (&) in packed arguments are converted to underscores (_) in file names.

The following table provides examples of how file names are created during Export to Disk:

Asset Name	Asset ID	File Name	Value of SUFFIX	SiteCatalog Pagename	Arguments Passed with Pagename	File Name of Exported File
Home	123			BF/Page/Home	`cid=123 c=Page`	BF-Page-Home_123.html
Home	123		.htm	BF/Page/Home	`cid=123 c=Page`	BF-Page-Home_123.htm
Home	123	home.html		BF/Page/Home	`cid=123 c=Page`	BF-Page-Home_home.html
Home	123	home.html	.htm	BF/Page/Home	`cid=123 c=Page`	BF-Page-Home_home.html
Home	123	home.htm		BF/Page/Home	`cid=123 c=Page PACKEDARGS= “topicword=oil”`	BF-Page-Home_topicword_oil_home.htm

Note

CS-Direct needs both the asset's ID and the asset's type to determine whether there is a file name provided for it. If the identity of the asset's type is not provided, the file name cannot be used. In such a case, CS-Direct uses the asset's object ID in place of its file name.

Export Starting Points

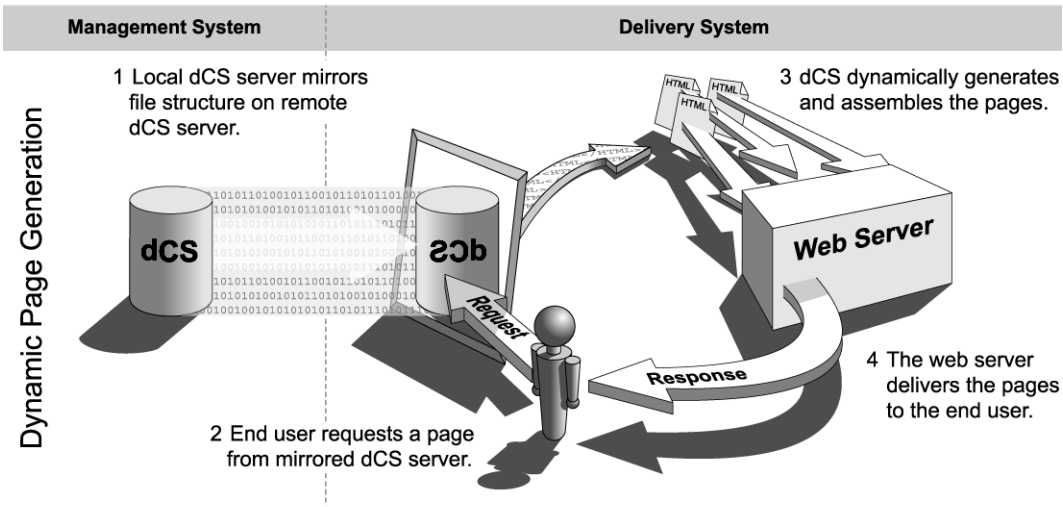
Export to Disk cannot begin rendering files unless it knows where to start. You tell it where to start by designating at least one page asset as an **export starting point** and specifying the template that should be used to render it. Content Server invokes the root element of the template's page name, 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.

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.

The Mirror to Server Delivery Type

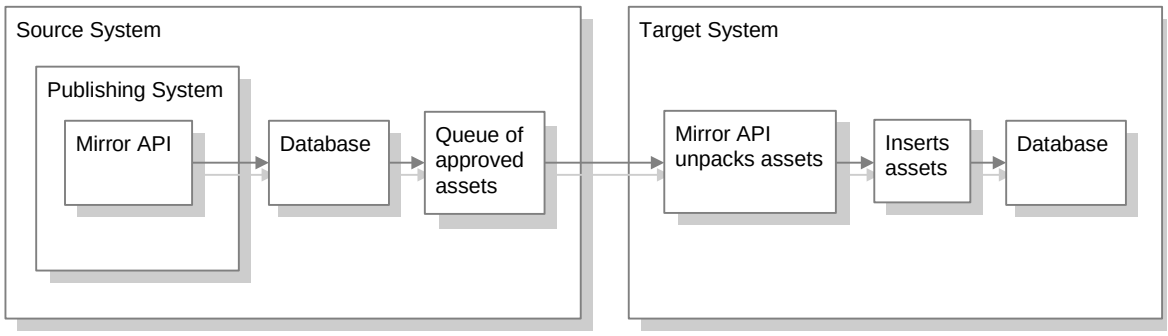
The CS-Direct Mirror to Server delivery type gathers information from the approval system, the publishing schedule, and the destination configurations, copies data to the destination destination, unpacks that data on the destination system, and then invokes the CacheManager servlet to refresh any pages that should be regenerated to take advantage of the new content.



Whether or not your CSEE delivery system is dynamic, you probably use the Mirror to Server delivery type to move data from your development systems to your management system. For information about that process, see “Troubleshooting” on page 249.

How Mirror to Server Works

When a Mirror to Server publishing session runs, this is what occurs:



1. On the source system, Mirror to Server 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. CS-Direct 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, Mirror to Server 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 10 (Mirror to Server): Turn Off Asset Invalidation on the Delivery System" on page 228 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 CS-Satellite 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 CSEE system. The CS-Satellite applications then use that information to refresh the CS-Satellite 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 Export to Disk delivery type, which writes to the publish log after each asset is exported, Mirror to Server 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, CS-Direct 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. CS-Direct 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.

Users and Mirror to Server

In addition to the batch user account on the source system that a publish event uses to process a publishing session, the Mirror to Server delivery type 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 Mirror to Server 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.

Publishing Arguments for Mirror to Server

There are three publishing arguments for Mirror to Server.

Because properties in the `futuretense.ini` file are global for all publishing destinations, there are two publishing arguments for Mirror to Server that enable you to specify destination-specific mirror users, in case you have more than one delivery system that you are publishing to from the same source and you do not want to name the mirror user account identically on each destination system.

Those publishing arguments are as follows:

- `REMOTEUSER`, which specifies the mirror user account on the destination system when it is different than the one specified by the `cs.mirroruser` property in the `futuretense.ini` file on the source.
- `REMOTEPASS`, which specifies the password for the user account designated by the `REMOTEUSER` argument. Note that value of this argument is not stored in an encrypted form, which means that anyone with access to the database on the source can see the password.

The final Mirror to Server publishing argument is one that it shares with Export to Disk:

- `VERBOSE`, which activates error logging during the publishing process. When set to 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 because it causes the publishing process to take longer than normal.

CacheManager

The CacheManager is a Content Server servlet that maintains the page cache on any dynamic CSEE 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 *CSEE Developer's Guide*.

The Initialize Mirror Destination Feature

The Mirror to Server delivery type copies information for approved assets from one CSEE database to another. The Initialize Mirror Destination feature 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 CSEE database to another.

When to Use Initialize Mirror Destination

You use the Initialize Mirror Destination 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.

Specifying Which Data to Mirror

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.
- Configuration details (but only when you are moving information from a development system to a management system):
  - Asset type subtypes, associations, and categories: the appropriate rows from the `AssocName`, `AssocNamed_Subtypes`, and `Category` tables are mirrored.
  - Start Menu items: the appropriate rows from the five Start Menu tables are mirrored.
  - Saved searches: the appropriate rows from the three save search tables are mirrored.
  - Workflow building blocks: the appropriate rows from the 27 workflow tables are mirrored.
- 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."

For More Information

Using the Initialize Mirror Destination feature is described in two places:

- In the context of setting up your system for publishing with the Mirror to Server delivery type—in the procedures “Step 1 (Mirror to Server): Set Up the Destination System” on page 222 and “Step 5 (Mirror to Server): Initialize the Destination” on page 226.
- In the section called “Troubleshooting” on page 249.

The Export Assets to XML Delivery Type

The CS-Direct **Export Assets to XML** delivery type is a hybrid between the Export to Disk and Mirror to Server delivery types. Assets are rendered into files, but this delivery type uses the Mirror to Server dependency calculation method rather than that of Export to Disk.

This delivery type differs from the other two in that it is really a data transformation method. While Export to Disk 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 delivery type.

Export to Disk creates static files that are to be delivered from a static CSEE delivery system. In contrast, Export Assets to XML creates **XML files** that are to be delivered to another **non-CSEE** system or database.

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 delivery type, typically you set up a quality assurance process to test the generated files before you move them to the external, non-CSEE 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:

value of `cs.pgexportfolder`/value of `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 delivery type does not use any information that is entered in an asset's **Path** or **Filename** field.

Configuring the Publishing Process for Your CSEE System

Configuring your system for publishing includes more than just creating destinations and the steps that you complete are different based on which delivery type you are using, Export to Disk or Mirror to Server.

The procedures in this section mention several properties in the `futuretense.ini`, `batch.ini`, and `futuretense_xcel.ini` files. In addition to the properties mentioned in this chapter, there are additional properties that you can use to fine-tune your system configuration. For a complete list of all properties, see Chapter 11, "Properties and Property Files."

Create the Batch User Account

There are certain steps that you must complete regardless of your delivery type. You must:

- Configure a **batch user** account for the publishing system on the source to use.
- Specify where the publish **logs** should be stored.

Complete the following steps:

1. Log in to the Content Server interface on your source system.
2. On the **Admin** tab, select **Content Server Management Tools > Users.**
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 CS-Engage)
   - VisitorAdmin (if your installation includes CS-Engage)
   - xceleditor
   - xceladmin

   If you need help with this step, see "Creating a New User" on page 40.
4. Start the Property Editor and open the `futuretense_xcel.ini` file for the source system. (If you need help with this step, see "Starting the Property Editor" on page 276.)
5. On the **Publishing** tab, set values for the following properties:
   - `xcelerate.batchhost`
     Set this property to the host name of **web server** (not the application server or the load balancer) 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 the settings by selecting **File** > **Save**.
7. Open the `batch.ini` file for the source system.
8. On 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 settings (**File** > **Save**) and then exit the Property Editor (**File** > **Exit**).
10. Stop and restart the application server.
11. Continue with the configuration procedure for the delivery type you are planning to use:
    - For Export to Disk, go to "Configuring Your System for Export to Disk Publishing," the next procedure in this section.
    - For Mirror to Server, go to "Configuring Your System for Mirror to Server Publishing" on page 222.
    - For Export Assets to XML, go to "Configuring Your System for Export Assets to XML" on page 229.

Configuring Your System for Export to Disk Publishing

The main steps when configuring a system to publish using the Export to Disk delivery type are these:

- Configure the export directory that stores the files.
- Create a publishing destination that points to the export directory.
- Map a URL prefix on the web server of the delivery system.
- Create an export starting point.
- Approve some test assets, including the export starting point.
- Publish the test assets and verify them.
- Set up the schedule.

Step 1 (Export to Disk): Specify the Base Export Directory

As mentioned previously in this chapter, the directory that CS-Direct 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, complete the following steps:

1. Start the Property Editor. If you need help with this step, see "Starting the Property Editor" on page 276.
2. Open the `futuretense.ini` file for the source CSEE system.
3. On the **Export/Mirror** tab select the `cs.pgexportfolder` property and set the value to the file directory (location) where all files should be exported to.

   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 "File and Directory Naming Conventions" on page 203.
4. Save the new value by selecting **File** > **Save**.
5. Select **File** > **Exit** to close the Property Editor.
6. Stop and restart the application server.

Note

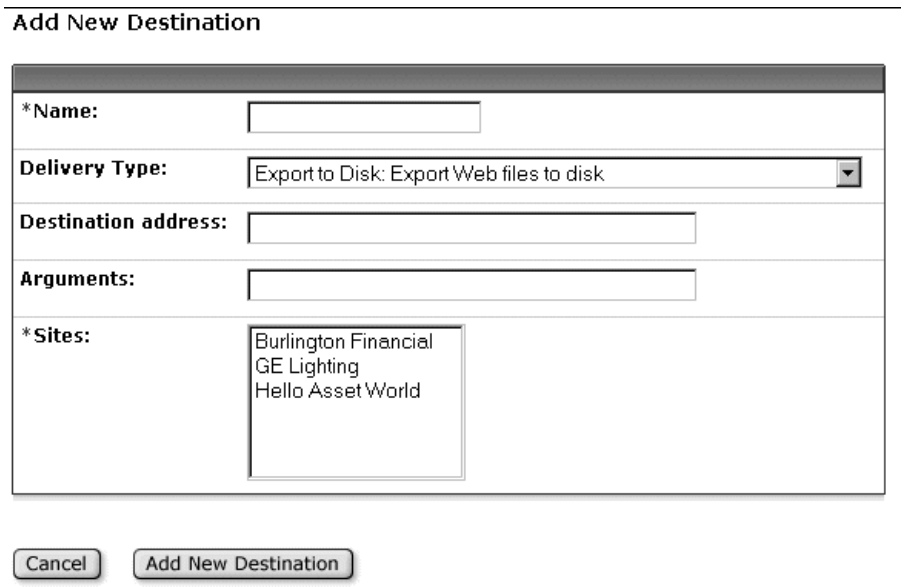
Before you run an export publish, make sure that this destination directory exists and that it has enough space for the HTML pages that will be created there.

Step 2 (Export to Disk): Configure an Export Destination

Create the export destination by completing the following steps:

1. Log in to the Content Server interface on the source CSEE system.
2. On the **Admin** tab, select **Publishing > Destinations**.
3. Select **Add New.**

   The **Add New Destination** form appears.



4. In the **Name** field, enter a unique name for the destination.
5. In the **Delivery Type** field, select **Export to Disk: Export Web Files to Disk**.
6. Leave the **Destination Address** empty. This field is for Mirror to Server only.
7. In the **Arguments** field, specify any arguments (`URLPREFIX`, `DIR`, `SUFFIX`, `SIMPLEDIR`, `SIMPLENAME`, `OLDTEMPLATE` or `VERBOSE`) that are appropriate for this destination, separated with the ampersand (&) character.

   For descriptions of the publishing arguments for this delivery type, see "Publishing Arguments for Export to Disk" on page 201.

   Here are some examples:

   ```
   URLPREFIX=/siteName&SUFFIX=htm
   URLPREFIX=/siteName&DIR=/Japan&SUFFIX=htm
   URLPREFIX=/siteName&DIR=/Japan&SUFFIX=htm&OLDTEMPLATE=true
   ```
8. In the **Sites** box, select the sites whose assets can be approved for and published to this destination.
9. Click **Add New Destination**.

   The information about this destination is written to the `PubTarget` table.
10. Repeat steps 2 through 9 for each additional export destination that you need to configure for your source system.

Step 3 (Export to Disk): 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 4 (Export to Disk): Create the Export Starting Points

To create an export starting point, complete the following steps:

1. Using the Content Server interface on the source system, find the asset that 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 publish approval form, select the destination that you are approving it for and click **Approve**.

The approval system calculates dependencies for all the assets linked to this asset and displays the results.

5. When CS-Direct 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 form; next to Path/Filename click the **Specify Path/Filename, Start points** link.
7. Configure the export starting point:
   a. Select the **Yes** option.
   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 "File and Directory Naming Conventions" on page 203.
   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.
8. Click **Save**.
9. Repeat steps 1 through 8 for each top-level page asset in the site.

Step 5 (Export to Disk): 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 237 for help with approving many assets at once.

Step 6 (Export to Disk): 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. In the Content Server interface on the source system, click the **Publishing** button from the main toolbar.
2. In the **Publish Console**, select your destination from the drop-down list and then click **Select Destination**.

   CS-Direct 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 Export to Disk 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 link to file that represents the top-level page in your site.

   The system displays the file in a browser window.
7. Scroll down that page, looking for errors, and click all the links in the file to verify that they work. Test all the links to all the other files that were generated.
8. In addition, test the results by directly entering the URL of the rendered home page in your browser and navigating the entire site. Note that to do this you must first set up a web server root on the CSEE management system that points to the export directory.
9. When you are satisfied that they the files are ready, use FTP or another file transfer protocol to move them to your CSEE delivery system.

Step 7 (Export to Disk): Set Up the Schedule

After you have ensured that your destination is configured correctly, that the file names and directory names are created according to your needs, and that 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 240.
- Plan how you will move the generated files to your web servers, for both the testing area and the delivery system. For example, you can set up a regular FTP transfer that automatically moves files to a testing area after a publishing session completes.

Configuring Your System for Mirror to Server Publishing

The main steps when configuring a system to publish using the Mirror to Server delivery type are as follows:

- Set up the destination system: create asset types and create a mirror user.
- Identify the mirror user on the destination to the publishing system on the source.
- If there is a firewall separating the source and destination systems, identify the proxy server.
- On the source system, create a publishing destination that points to the destination system.
- Initialize the destination destination.
- Approve some test assets.
- Publish the test assets and then verify the results.
- Set up the schedule.

Step 1 (Mirror to Server): Set Up the Destination System

To mirror publish assets from one CSEE system to another, the sites and asset types must be the same on both the source and the destination system.

When the delivery type is Mirror to Server, 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 source CSEE system and the destination CSEE system must have the exact same CSEE products and database tables installed on them. Additionally, the database properties, if not an exact match, must be compatible. Otherwise, your assets cannot be mirrored successfully.

To set up your destination system, complete the following steps:

1. If you have any flex asset types, log in to the Content Server interface on the destination system and use **Flex Family Maker** to create the flex asset types. Be sure that their names **exactly** match the names that you used on the development system.
2. If you have any custom basic asset type, use AssetMaker on the destination system to create them. Follow the steps that describe how to use AssetMaker in the *CSEE Developer's Guide* and be sure to set an appropriate value in the **Defdir** field on the **Create Asset Table** form. Be sure that the name of the asset types **exactly** match the names that you used on the development system.
3. 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.
4. On the destination system, create the mirror user. This user must have the following ACLs:
   - Browser

- ElementEditor
- PageEditor
- TableEditor
- Visitor (if CS-Engage is installed)
- VisitorAdmin (if CS-Engage is installed)
- xceladmin
- xceleditor

Additionally, because the CacheManager uses the mirror user account 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, if ACLs are assigned to any page entries in the `SiteCatalog` table or database tables that are holding data that needs to be rendered, the mirror user account must be assigned those ACLs, too.

If the destination system has any of the sample sites installed, a user named "mirroruser" already exists. For security reasons, if you decide to use this user as your mirror user, be sure to **change the password** for this user. If you decide to create a different mirror user, be sure to delete the sample site mirroruser.

Note

If you plan to mirror publish from any source system to more than one destination system, use either 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 one only 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.

If you need help with this step, see "Creating a New User" on page 40.

Step 2 (Mirror to Server): 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.

Complete the following steps:

1. Start the Property Editor and open the `futuretense.ini` file for the source CSEE system. If you need help with this step, see "Starting the Property Editor" on page 276.
2. On 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. Select **File** > **Save** to save the settings.
4. Do one of the following:
   - If there is a firewall separating your source system from the destination system, go to "Step 3 (Mirror to Server): Identify the Proxy Server (If There Is a Firewall)" on page 224.
   - If there is not a firewall separating your source and destination systems, select **File** > **Exit**. Stop and restart the application server. Then go to "Step 4 (Mirror to Server): Configure a Mirror Destination" on page 224.

Step 3 (Mirror to Server): 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:

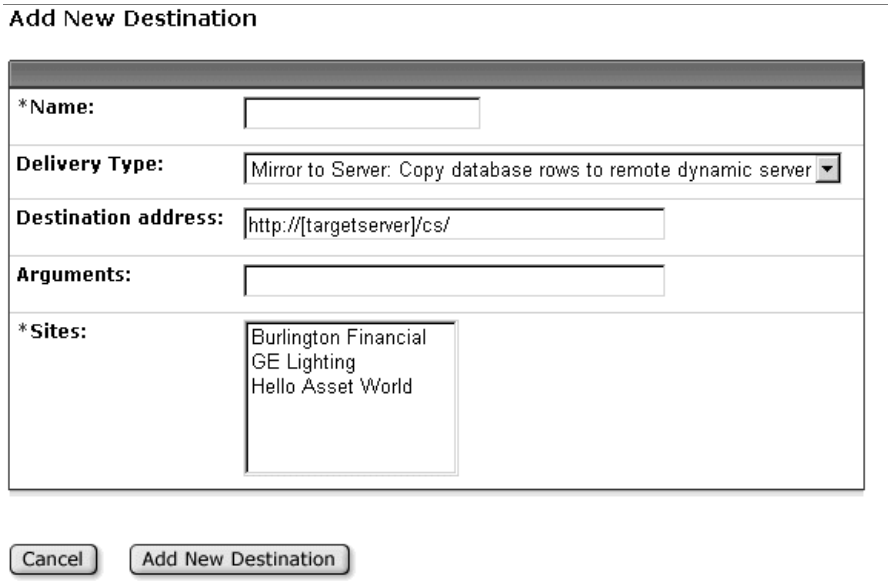
1. On the **Export/Mirror** tab, specify values for these properties:
   - `cs.mirrorproxyserver`
     Set this property to either the name or the IP address of the firewall's proxy server.
   - `cs.mirrorproxyserverport`
     Set this property to the port number of the firewall's proxy server.
2. Select **File** > **Save** to save the values.
3. Select **File** > **Exit** to close the Property Editor.
4. Stop and restart the application server.

Step 4 (Mirror to Server): Configure a Mirror Destination

Create the mirror destination by completing the following steps:

1. Log in to the Content Server interface on the source CSEE system.
2. On the **Admin** tab, select **Publishing** > **Destinations**.
3. Select **Add New.**

The **Add New Destination** form appears.



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.

   Example: iPlanet Application Server (iAS)

   `http://your_Webserver_name/NASApp/cs/`

   Example: WebLogic, WebSphere, and Sun ONE Application Server

   `http://your_Webserver_name/servlet/`

   Note

   A slash is required at the end of the URLs because these URLs are appended dynamically.

7. In the **Arguments** field, specify any appropriate arguments for this destination, separated by the ampersand (&) character. For more information, see “Publishing Arguments for Mirror to Server” on page 211.
8. In the **Sites** box, select the sites whose assets can be approved for and published to this destination.
9. Click **Add New Destination**.

   The information about this destination is written to the `Pubdestination` table.
10. Repeat steps 2 through 9 for each additional mirror destination that you need to configure for your source system.

Step 5 (Mirror to Server): Initialize the Destination

You must initialize the destination system before you can publish to it.

Complete the following steps:

1. In the Content Server interface on the source system, select **Admin** > **Publishing** > **Destinations**.
2. Double-click on the destination that you want to initialize.
3. In the **Publish Destination** form, select the **Initialize Mirror Destination** link.

   The **Initialize Mirror Destination** screen appears.
4. Select the sites whose assets are published to this destination.
5. If you are migrating a site from a development system to a management system, select the appropriate options in the Configuration section. For more information, see the procedures in the section "Migrating a Site from One System to Another System" on page 232.
6. Enter the names of any required auxiliary tables. For help with this step, see the procedures in the section "Migrating a Site from One System to Another System" on page 232 and talk to your developers.
7. Click **Mirror**.

   If the initialization is successful, a verification message appears on the screen. If not, an error message appears.
8. Log in to the destination system, select **Admin** > **Sites**, and then select the site you just mirrored.
9. Enable the asset types for this site.

Step 6 (Mirror to Server): 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 237 for help with approving many assets at once.

If you want to perform a simple test of your configuration, you could temporarily create a home page asset with just a few dependents. For information about approving individual assets, see the *CSEE User's Guide*.

Step 7 (Mirror to Server): Publish the Assets

After you have approved assets that can be published to your destination, you can run a test publishing session:

1. In the Content Server interface on the source system, click the **Publishing** button from the main toolbar.
2. In the **Publish Console**, select your mirror destination from the drop-down list and then click **Select Destination**.

   CS-Direct displays information about the assets that are ready to be published to this destination.
3. Click **Publish**.

The publishing system mirrors all the approved assets for this destination to the Content Server database on the destination system.

Step 8 (Mirror to Server): Test the Results

To test the results, use your browser to navigate to the URL of the home page asset on the destination system and examine the site.

First, you must determine what that URL is. A Content Server URL is constructed by concatenating the following values:

- The name of the destination system's server and sometimes the port number as well.
- The CGI path, which it 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/` and for iPlanet Application Server, this path is `/NASApp/cs/`.
- The string `ContentServer?pagename=`
- A page name from a `SiteCatalog` entry.
- Additional information that is passed in with the CS-Direct page criteria variables, `c`, `cid`, `tid`, and `p` (see the *CSEE 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 then use it to 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 "Starting the Property Editor" on page 276.)
2. Select the **App Server** tab.
3. Select the `ft.cgipath` property and write down the value.
4. Select the **Compatability** tab.
5. Select the `ft.approot` property and write down the value.
6. Select **File** > **Exit** to close the Property Editor.
7. Open a text editor. Type the server name, a slash (/), and then the cgipath. Precede the server name with the proper protocol—http:// or https://—as in the following examples:

   iPlanet Application Server (iAS):

   `http://serverABC/NASApp/cs/`

   WebLogic, WebSphere, and Sun ONE Application Server

   `http://serverABC/servlet/`
8. 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 iAS:

```
http://serverABC/NASApp/cs/ContentServer?pagename=ExampleSite/
Page/Home
```

Example for WebLogic, WebSphere, and Sun ONE:

```
http://serverABC/servlet/ContentServer?pagename=ExampleSite/
Page/Home
```

9. Open a web browser and enter the URL for the home page in the address field.
10. Scroll from the top to the bottom of the page and check for errors.
11. Click on all links to verify that they are working properly.

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 9 (Mirror to Server): 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 240.
- 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 *CSEE 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 or other process to move those uploaded image files from the management system to the delivery system. Because these files are not assets, they are not mirrored to the destination.

Step 10 (Mirror to Server): 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 "Starting the Property Editor" on page 276.
2. On the **Publishing** tab, select the `xcelerate.publishinvalidate` property.
3. Set the value to `false`.
4. Save the file and exit the Property Editor.

Configuring Your System for Export Assets to XML

The main steps when configuring for the Export Assets to XML delivery type are as follows:

- Configure the export directory that stores the files.
- Create a publishing destination that points to the export directory.
- Approve some test assets.
- Publish the test assets and test them.
- Set up the schedule.

Step 1 (Export to XML): Specify the Base Export Directory

As mentioned previously in this chapter, the directory that CS-Direct exports files to is determined by the `cs.pgexportfolder` property in the `futuretense.ini` file.

To set the base directory for the exported files, complete the following steps:

1. Start the Property Editor. If you need help with this step, see "Starting the Property Editor" on page 276.
2. Open the `futuretense.ini` file for the source CSEE system.
3. On the **Export/Mirror** tab select the `cs.pgexportfolder` property and set the value to the file directory (location) where all files should be exported to.

   This is a global setting for the system. If you have multiple destinations or are publishing with both the Export to Disk and Export Assets to XML delivery types 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 new value by selecting **File** > **Save**.
5. Select **File** > **Exit** to close the Property Editor.
6. Stop and restart the application server.

Note

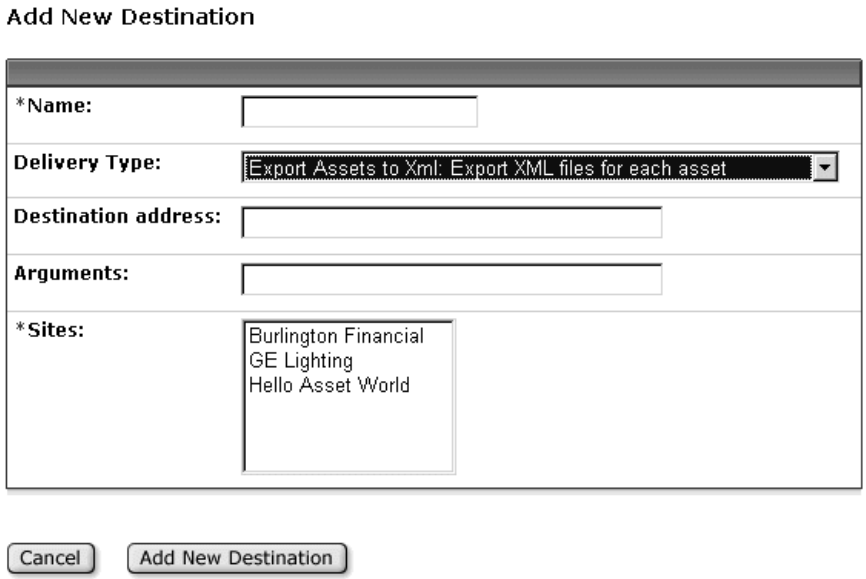
Before you run an Export to XML publish, make sure that this destination directory exists and that is has enough space for the XML files that will be created there.

Step 2 (Export to XML): Configure an Export Assets to XML Destination

Next, create the Export to XML destination by completing the following steps:

1. Log in to the Content Server interface on the source CSEE system.
2. On the **Admin** tab, select **Publishing > Destinations**.
3. Select **Add New.**

   The **Add New Destination** form appears.



4. In the **Name** field, enter a unique name for the destination.
5. In the **Delivery Type** field, select **Export Assets to XML**.
6. Leave the **Destination Address** empty. This field is for Mirror to Server only.
7. (Optional) If you want to use a different directory than is specified by the `cs.pgexportfolder` property, click in the **Arguments** field and set an `DIR` argument.

   For example:

   ```
   DIR=/export/XML
   ```

8. In the **Sites** box, select the sites whose assets can be approved for and published to this destination.
9. Click **Add**.

   The information about this destination is written to the `PubTarget` table.

10. Repeat steps 2 through 9 for each additional export destination that you need to configure for your source system.

Step 3 (Export to XML): 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 237 for help with approving many assets at once.

Step 4 (Export to XML): 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. In the Content Server interface on the source system, click the **Publishing** button from the main toolbar.
2. In the **Publish Console**, select your destination from the drop-down list and then click **Select Destination**.

   CS-Direct displays information about the assets that are ready to be published to this destination.
3. Click **Publish**.

   The publishing system exports all the approved assets for this destination into files.
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 5 (Export to XML): 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 240.
- Plan how you will move the generated files to your external, non-CSEE systems. For example, you can set up a regular FTP transfer that automatically moves files to a testing area after a publishing session completes.

Additional Publishing Procedures

In addition to the configuration steps described in preceding sections of this chapter, administrators and developers also perform the following publishing procedures whenever necessary:

- Migrating a Site from One System to Another System
- Approving Multiple Assets
- Creating Destinations
- Editing Destinations
- Deleting Destinations
- Creating Export Starting Points
- Scheduling Publish Events
- Reading the Schedule Abbreviations
- Editing Publish Events
- Overriding the Schedule
- Assigning Approval or Preview Templates
- Monitoring Sessions in the Publishing Console
- Verifying Publishing Readiness
- Managing Publishing History Information
- Publishing All Approved Assets

Migrating a Site from One System to Another System

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 **Initialize Mirror Destination** feature to migrate it to the management system and the delivery system.

Note

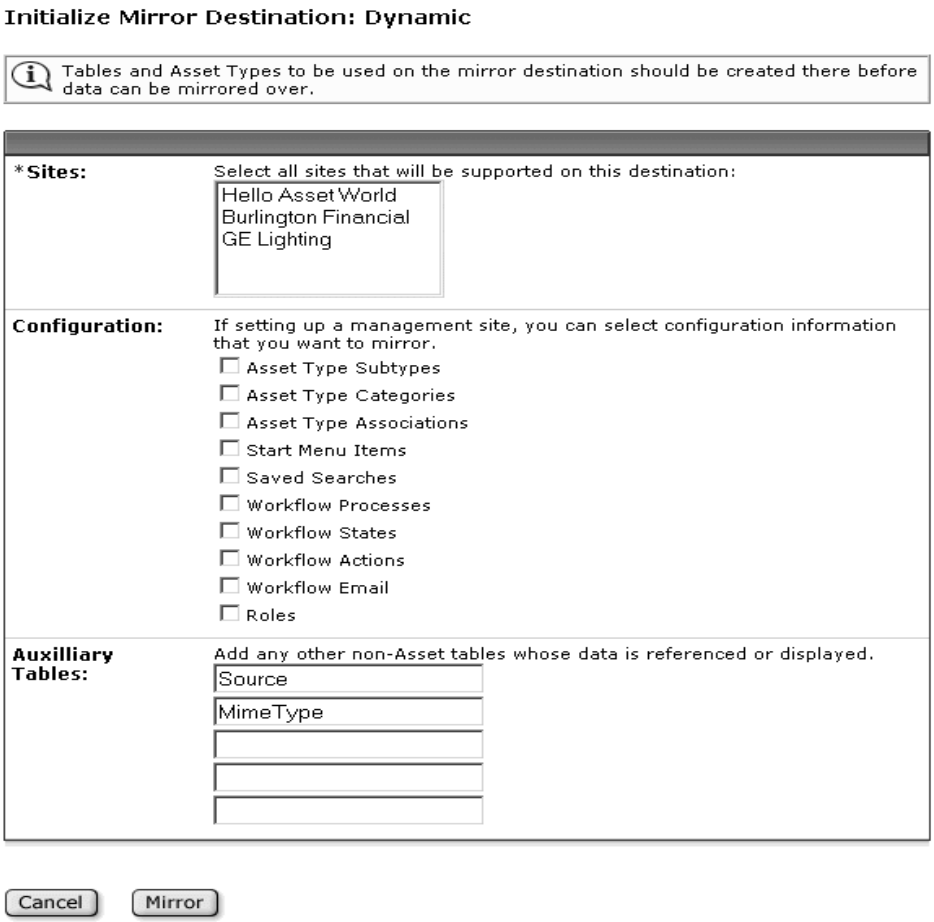
You can use this feature to move your configuration data only when **none** of the **sample sites** exist on the **destination** system.

Moving a Site from a Development System to a Management System

Complete the following procedure **once**:

1. If you have any flex asset types, log in to CS-Direct on the management (destination) system as a user who has access to the **Admin** tab and use **Flex Family Maker** to create the flex asset types. Be sure that their names **exactly** match the names that you used on the development system. For help with this step, see the *CSEE Developer's Guide*.
2. If you have any custom basic asset types, use AssetMaker to create them on the destination system. For help with this step, see the *CSEE Developer's Guide*. Be sure that you complete the following additional steps:
   - Be sure that the value in the **Defdir** field is appropriate for the management system.
   - Clear the **Add "General" Category** check box.
3. 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.
4. On the management system, delete the named associations from the collection asset type. Complete the following steps:
   a. Select the **Admin** tab **> Asset Types > Collection > Associations**.
   b. Double-click on Query 1.
   c. In the form on the right, click the **Delete** icon.
   d. Click **Delete Association**.
   e. Repeat steps b through d for Query 2 and Query 3.
5. On the management system, create all the users.
6. On the source system, log in to the Content Server interface.
7. Create a **Publishing Destination** for the destination system that uses the Mirror to Server delivery type. For help with this step, see "Step 4 (Mirror to Server): Configure a Mirror Destination" on page 224.
8. In the **Publish Destination** form for this destination, select **Initialize Mirror Destination**.

The **Initialize Mirror Destination** form appears:



9. In the **Sites** list, select the names of the sites that you want to enable on the management system.
10. If your asset types have subtypes, categories, or associations, select the appropriate options from the list in the **Configuration** section.
11. If you or the developers designed saved search items or start menu items for your content providers to use, select those options, too.
12. If there are workflow processes for your asset types, select the appropriate workflow items.
13. 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. Note that if you have created any additional auxiliary tables that AssetMaker or CS-Direct does not provide, you must create those tables on the destination before proceeding to the next step.

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 8, 9, and 13 for the remaining tables.

14. Click **Mirror**.

    Based on the sites that you selected in step 9, appropriate rows from the `Publication`, `SitePlanTree`, and `AssetType` tables are copied to the destination.

    Based on which asset type configuration options you selected in step 10, 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 11, appropriate rows from the workflow tables are copied to the destination.

    Additionally, all the rows in the tables that you specified as Auxiliary Tables are copied to the destination.

15. Log in to the destination system as the **admin** user and configure the tree tabs. Complete the following steps:

    a. On the **Admin** tab, select **Tree**. Then edit the Admin tree tab and select all the sites that you just mirrored.

    b. Edit the rest of the default tree tabs so they are enabled for your sites and assign the appropriate roles to them. For help with this step, see “Editing Tree Tabs” on page 79.

    c. Create the other tree tabs that you need and assign the appropriate roles to them.

16. Enable all the asset types for the sites that you mirrored. For help with this step, see “Enabling an Asset Type for a Site” on page 69.

17. Configure the users for the site. For help with this step, see “Granting Users Access to Sites (Assigning Roles to Users)” on page 49.

18. 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 237.

19. 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.

20. 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 *CSEE 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. If you have any flex asset types, log in to CS-Direct on the delivery (destination) system as a user who has access to the **Admin** tab and use **Flex Family Maker** to create the flex asset types. Be sure that their names **exactly** match the names that you used on the development system. For help with this step, see the *CSEE Developer's Guide*.
2. If you have any custom basic asset types, use AssetMaker to create them on the delivery system. For help with this step, see the *CSEE Developer's Guide*. Be sure that you complete the following additional steps:
   - Be sure that the value in the **Defdir** field is appropriate for the management system.
   - Clear the **Add "General" Category** check box.
3. 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.
4. On the delivery system, delete the named associations from the collection asset type. Complete the following steps:
   a. Select the **Admin** tab > **Asset Types** > **Collection** > **Associations**.
   b. Double-click on Query 1.
   c. In the form on the right, click the **Delete** icon.
   d. Click **Delete Association**.
   e. Repeat steps b through d for Query 2 and Query 3.
5. On the source system, log in to the Content Server interface.
6. Create a **Publishing Destination** for the delivery system that uses the Mirror to Server delivery type. For help with this step, see "Step 4 (Mirror to Server): Configure a Mirror Destination" on page 224.
7. In the **Publish Destination** form, select **Initialize Mirror Destination**.
8. In the **Initialize Mirror Destination** form, select the names of the sites that you want to enable on the destination system.
9. In the **Configuration** section, select any or all of the following options, as necessary:
   - Asset Type Subtypes
   - Asset Type Categories
   - Asset Type Associations

   Do **not** select start menu items, saved searches, or any of the workflow options.

10. 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 this table for any of your custom asset types.
    - Any other auxiliary tables (such as lookup tables) for your asset types. Note that if you have created any additional auxiliary tables that AssetMaker or CS-Direct does not provide, you must create those tables on the destination before proceeding to the next step.

    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 7, 8, and 10 for the remaining tables.
11. Click **Mirror**.
12. Log in to the Content Server interface on the delivery system and enable all the asset types for the sites that you mirrored. For help with this step, see "Enabling an Asset Type for a Site" on page 69.
13. 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 237.
14. 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.
15. 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 *CSEE Developer's Guide.*
16. 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

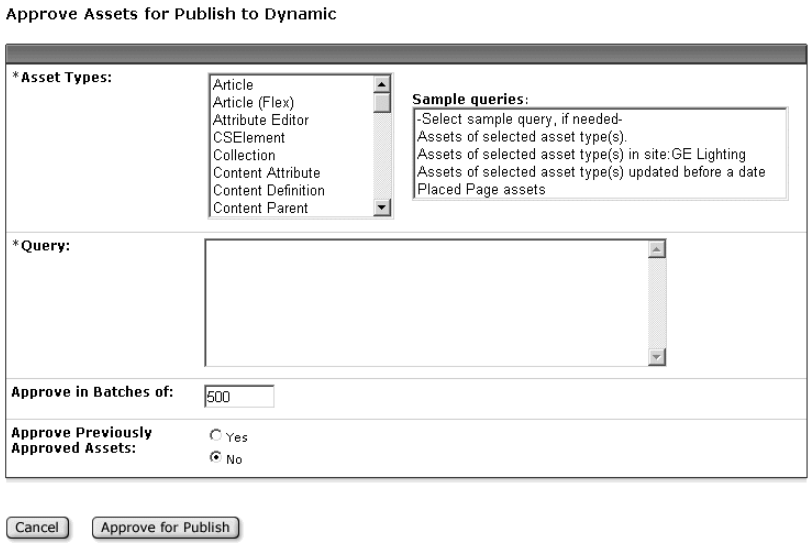
Each publishing destination has an option to **Approve Multiple Assets**. It is especially useful for upgrades and for the first publishing session to a destination.

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 *CSEE 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 item **Approve Multiple Assets** option.

The **Approve Assets** form appears:



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.

   CS-Direct 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 Chapter 11, "Properties and Property Files.")
- 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 CS-Direct 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 CS-Direct 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

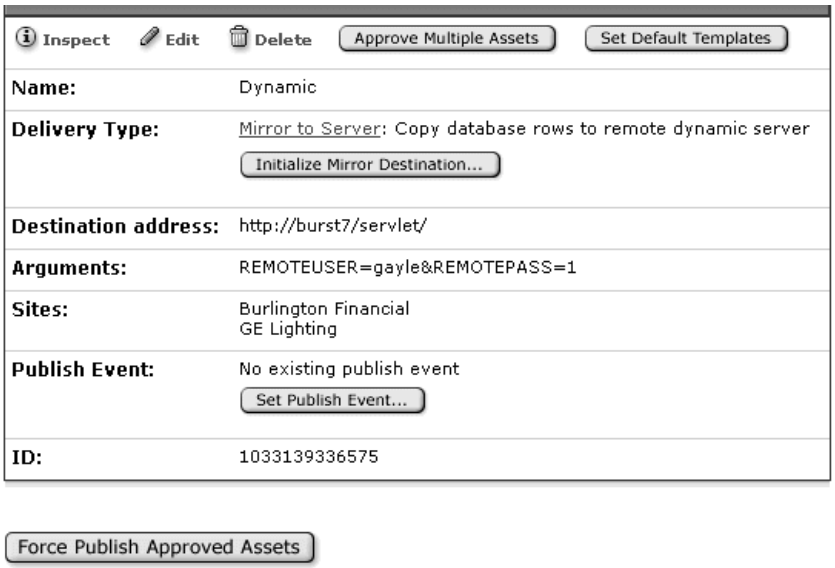
The procedures for creating destinations are described in the previous section. For information about creating new destinations, use one of the procedures in the following list, as appropriate:

- To create an export destination, see "Step 2 (Export to Disk): Configure an Export Destination" on page 218.
- To create a mirror destination, see "Step 4 (Mirror to Server): Configure a Mirror Destination" on page 224.
- To create an export assets destination, see "Step 2 (Export to XML): Configure an Export Assets to XML Destination" on page 230.

Editing Destinations

To edit a destination:

1. Select **Admin > Publishing > Destinations**,
2. Double-click on the destination that you want to edit.



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 2 (Export to Disk): Configure an Export Destination" on page 218.
- For a Mirror to Server destination, see "Step 4 (Mirror to Server): Configure a Mirror Destination" on page 224.
- For an Export Assets to XML destination, see "Step 2 (Export to XML): Configure an Export Assets to XML Destination" on page 230.

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.

   CS-Direct displays a verification message.

Creating Export Starting Points

Export starting points are defined in "Export Starting Points" on page 208. For information about creating them, see "Step 4 (Export to Disk): Create the Export Starting Points" on page 219.

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.

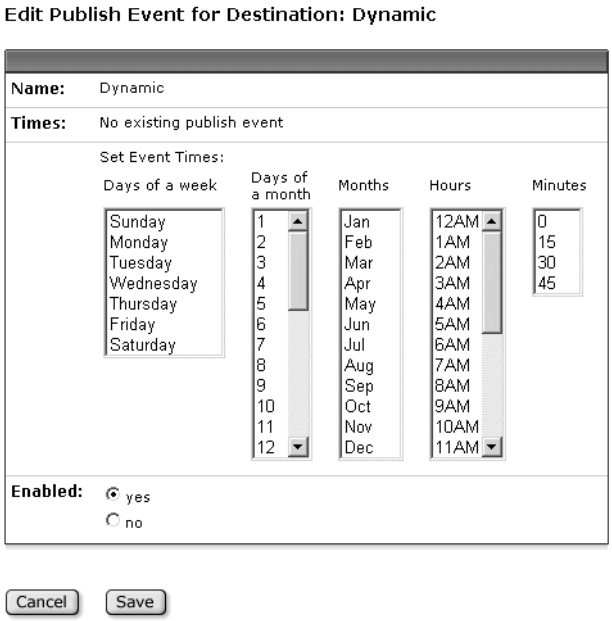
Note the following:

- 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.

CS-Direct displays the **Publish Event** form:



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 CS-Direct 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 241), 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.

   CS-Direct 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.

   CS-Direct determines whether there are any assets that can be published to the destination and displays a summary form like this one:

   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**.

   CS-Direct 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 CS-Direct 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

As mentioned earlier in this chapter (see "Calculating Dependencies for Export to Disk" on page 194), when assets are approved for a publishing destination that uses the Export to Disk publishing method, the approval system examines the template assigned to the asset to determine its dependencies.

However, when Export to Disk actually 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.

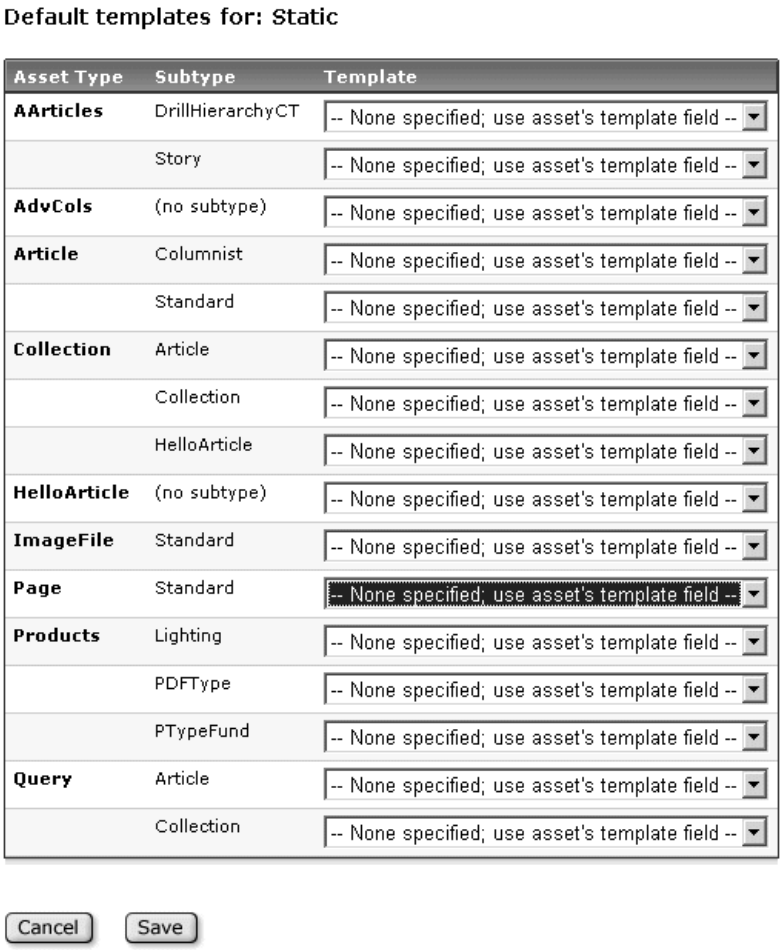
What if the template that contains the most representative set of dependencies is not the template that you want to assign to the asset? Set it as the Default Approval Template for assets of that type.

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. CS-Direct displays a **Default Templates** form.
4. Click **Edit**.

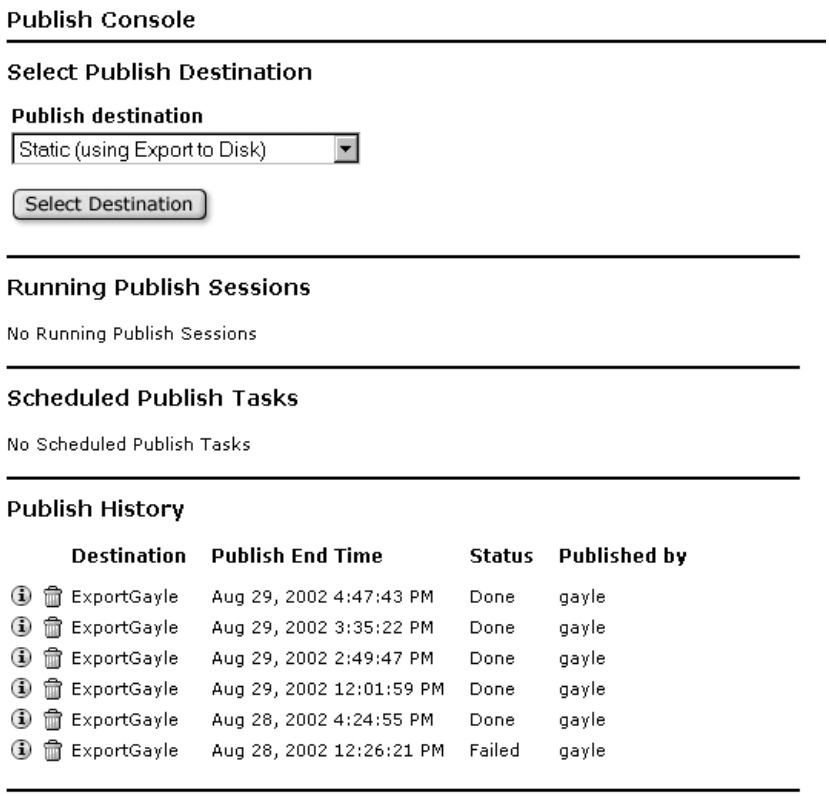
   The system displays a form like this one:



5. For each asset type that you need to configure, select the template that you want CS-Direct 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:



The console displays a summary of all the publishing activity, including any sessions that are currently running, scheduled, or completed.

Verifying Publishing Readiness

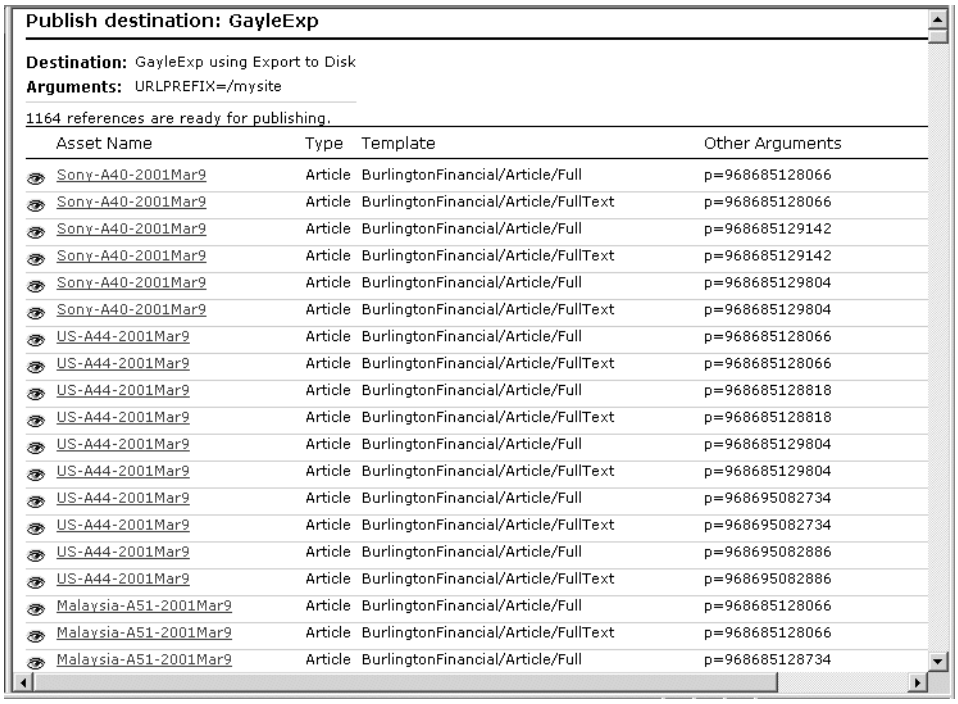
To determine whether a publishing session is ready or whether there are any problems with unapproved assets, follow these steps:

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:

3. To see the details, click on the link.

CS-Direct 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 “Held Assets” on page 193.
- For procedures that describe how to work with assets, see the *CSEE 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, CS-Direct 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 CS-Direct 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 249.

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 delivery type:

- For Mirror to Server, a reference is an **asset**. Each asset that is published is listed by its asset type and its ID.
- For Export to Disk 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 an Export to Disk 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 249 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 2 (Mirror to Server): Identify the Mirror User to the Source System" on page 223 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 1 (Mirror to Server): Set Up the Destination System" on page 222 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 4 (Mirror to Server): Configure a Mirror Destination” on page 224.

-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.
- If you are using the AltaVista search engine on the source system, you see this problem if you have not yet configured the search engine on the destination or if you have not configured it correctly

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.

The following messages appear when your AltaVista search engine is configured incorrectly:

- "-806 Search remove index failed" in the `futuretense.txt` file indicates that the search engine is not installed on the destination system.
- "-823 Native method mismatch" in the `futuretense.txt` file indicates that the `doc_converters` path is not included in `LD_Library_PATH` in the startup script. See the AltaVista release notes for configuration instructions.

Other Indicators of System or Configuration Issues

This section describes symptoms your CSEE 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 "Create the Batch User Account" on page 216 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 Export to Disk and Export Assets to XML, you need enough disk space in your file system to store all the files that are being generated.
- For Mirror to Server, 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 CS-Satellite—either the co-resident Satellite on a management or development system or a remote CS-Satellite 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 CS-Satellite 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 CS-Satellite and your remote CS-Satellite 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.

Chapter 9

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.

The CS-Direct application 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 *CSEE 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
- Enabling Revision Tracking
- Disabling Revision Tracking
- Unlocking Revisions
- Additional Revision Tracking Functions for Non-Asset Tables

Overview

Content Server provides revision tracking functionality through its revision tracking API. CS-Direct uses this API 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.

CS-Direct'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 (defdir) that you specify. 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, CS-Direct 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 approriate 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 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 is 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 on 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 to lock (check out) a row and then unlock it (check it back in) when you are finished with it or the Content Server Management Tools.

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 and revision tracking XML or JSP tags. For information, see the *CSEE Developer's Tag Reference* and the *CSEE 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

You enable revision tracking differently for asset types than you do for tables that do not hold assets.

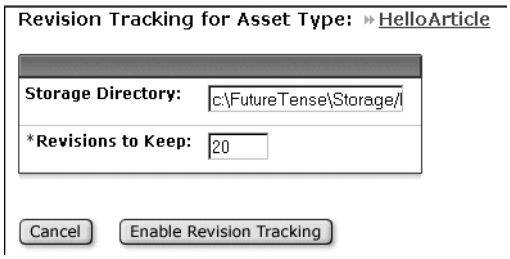
- For asset types, you use the nodes under the **Asset Type** item on the **Admin** tab.
- For non-asset tables, you use the **Revision Tracking** node under the **Content Server Management Tools** item on the **Admin** tab.

Enabling Revision Tracking for Asset Types

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Asset Types** item.
2. Expand the asset type for which you want to enable revision tracking.
3. Select **Revision Tracking** > **Track**.

   The **Track Asset Type** form appears.



4. Click in the **Storage Directory** field and enter the path name to the directory where you want the supporting text files for versions of assets of this type to be stored. Use the full pathname and do not add a slash character after the directory name.
5. Click in the **Revisions to Keep** field and enter the number of revisions that you want the system to keep stored. Once this number of revisions is stored, the oldest version is deleted when a new revision is created. 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.
6. 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** to set revision tracking. Note that in older versions of the product, database tables were called “catalogs” and the **Content Server Management Tools** still use the old terminology in some places.

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and then expand the **Content Server Management Tools**.
2. Double-click the **Revision Tracking** option.
3. In the **Revision Tracking** form, select **Track Tables** and click **OK.**

4. In the **Track Tables** form, click in the **Enter root storage directory** and enter the pathname to the directory where you want versions of rows to be stored. Use the full pathname and do not add a slash character after the directory name.
5. In the field **Enter number of revisions to keep** field, enter the number of revisions to keep in storage for rollback purposes.
6. Select the **Track?** option next to the name of the table for which you are enabling revision tracking.
7. Click **Track Catalogs**.

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, select **Admin** tab **>** **Asset Types** **>** the asset type you want to modify **>** **Revision Tracking** **>** **Set Revisions** and then increase the value.
- For non-asset tables, select **Admin** tab **>** **Content Server Management Tools** **>** **Revision Tracking** **>** **Set Table Revisions** and then increase the value.

Decreasing

Althought you might need to decrease the number of versions while you are testing configuration settings on a CSEE development system or while you are fine-tuning the CSEE managment 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:
   - If you are decreasing the number of versions for an asset type, select **Admin** > **Asset Types** > **your asset type** > **Revision Tracking** > **Set Revisions** and decrease the number of versions stored for that asset type.
   - If you are decreasing the number of versions for a non-asset table, select **Admin** > **Content Server Administrator Tools** > **Revision Tracking** > **Set Catalog Revisions** and decrease the number of versions stored for that table.

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, complete these steps:

1. Select the **Admin** tab and then expand **Content Server Administrator Tools**.
2. Double-click the **Revision Tracking** option.
3. In the **Revision Tracking** form, click in the **Enter Table Name** field and enter the name of the table.
4. Click in the **Enter Value for Key** field and enter the ID of the asset or the row whose versions you want to delete. (You can obtain the ID of an asset by inspecting it in the Content Server interface. You can obtain the ID of any row by examing the table with Content Server Explorer.)
5. Select **Delete Revisions** and then click **OK**.
6. 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.
7. 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:

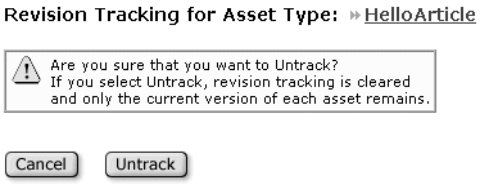
1. Use the Content Server Management Tools revision tracking forms to delete all the versions stored for the table. See the procedure "Deleting Revisions" on page 263.
2. Disable revision tracking for the table. See the procedures in this section.
3. Manually delete all the text files from the storage directory for that asset type or database table.

Disabling Revision Tracking for Asset Types

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and expand the **Asset Types** item.
2. Expand the asset type for which you want to disable revision tracking.
3. Select **Revision Tracking** and double-click **Untrack**.

   A confirmation message appears.



4. Click **Untrack.**

Disabling Revision Tracking for Non-Asset Tables

Complete the following steps:

1. On the **Admin** tab, expand the **Content Server Management Tools** item.
2. Double-click the **Revision Tracking** option.
3. In the **Revision Tracking** form, select **Untrack Tables** and click **OK**.

4. In the **Untrack Tables** form screen appears, select the **Untrack?** option next to the table that you want to disable revision tracking for.
5. Click **Untrack Catalogs**.

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** item on the **Admin** tab.
- For non-asset tables, use the **Content Server Management Tools** revision tracking forms.

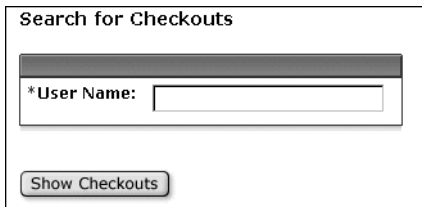
Clearing Checkouts for Assets

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.

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and then double-click on **Clear Checkouts**.

   The **Search for Checkouts** form appears:



2. Enter the name of the user who has the asset locked and click **Show Checkouts**.
3. In the list of assets checked out to that user, select the **Clear** option for each asset that you want to check back in.
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.

When you know the object ID of the item, complete the following steps:

1. In the Content Server interface, select the **Admin** tab and then expand the **Content Server Management Tools** item.
2. Double-click the **Revision Tracking** option.
3. In the **Revision Tracking** form, enter the name of the table, select **Unlock Rows.**

4. In the **Unlock Rows** form, select the option next to the object ID for each row that you want to unlock. You can use the **All** button to check all of the boxes and the **None** button to clear all of the boxes.
5. When you have only the boxes checked for rows that you want to unlock, click **Unlock Records**.

Additional Revision Tracking Functions for Non-Asset Tables

The Content Server 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 261)
- Untrack Tables (see "Disabling Revision Tracking" on page 264, above)
- Set Table Revisions (see "Editing Revision Tracking Settings" on page 262)
- Delete Revisions (see "Deleting Revisions" on page 263
- Unlock Rows (see "Unlocking Revisions" on page 265)

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.

Chapter 10

Search Engines

The native search feature provided with CS-Direct for searching for assets in the Content Server interface is a SQL-based search that performs database searches for assets. For rich-text searches based on an index, you must use one of the supported search engine modules.

If your organization purchased one of the search engine modules (Verity or AltaVista) to use with your CSEE systems, it was installed when your CSEE system was installed.

This chapter describes how to configure your asset types so that your assets are indexed correctly and can be found by your search engine. This chapter contains the following sections:

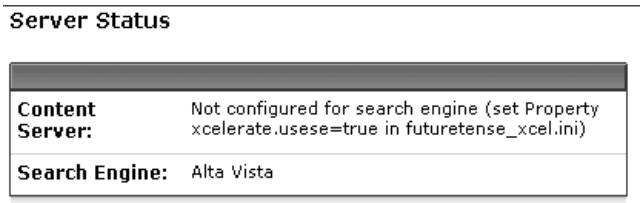
- Overview
- Configuring Asset Types for Your Search Engine
- Rebuilding an Index

Overview

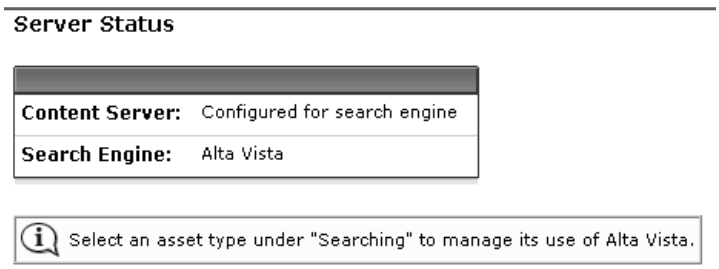
As the administrator, setting up the indexes for asset types is simple: you select the **Searching** option on the **Admin** tab and select which of your asset types should be enabled.

The Searching Option

The **Searching** option on the **Admin** tab is active if only if a search engine module is installed. When a search engine moduel has not been installed, CS-Direct displays the following message when you select **Admin > Searching**:



If a search engine module has been installed and configured, CS-Direct displays the following message when you select **Admin > Searching**:



Search Engine Properties

Although your search engine module was configured when your CSEE system was installed, you might need to fine-tune its configuration. There are several search properties in the `futuretense.ini` file and the `futuretense_xcel.ini` file.

For information about these properties, see "Search" on page 328.

Asset Types and Search Engines

Different kinds of asset types are indexed differently by the search engine modules.

Default (Core) Asset Types

When you enable any of the core asset types for the **Searching** option, each of the input fields of the asset is indexed. That is, drop-down fields are not indexed.

The core asset types are made available as follows:

- CS-Direct delivers the template, query, collection, link, CSElement, SiteEntry, and page asset types.
- CS-Direct Advantage delivers the attribute editor asset type.

- CS-Engage delivers the visitor attribute, history attribute, history definition, segment, recommendation, and promotion asset types.

AssetMaker Asset Types

Basic asset types that are created with AssetMaker are defined by an asset descriptor file. There are two kinds of fields in an asset type created by AssetMaker:

- Custom fields – these are the fields that are described in the asset descriptor file. For example, custom fields for the ImageFile asset type include Artist, Alt Text, Height, Width, and so on.
- Default fields – in addition to the custom fields specified in the asset descriptor file, AssetMaker asset types have the same default asset type fields that all asset types have: Name, Description, Status, Updated By, and so on.

When you enable an AssetMaker asset type for the **Searching** option, all of the default fields that are not drop-down fields are indexed. (For example, Category and Source are not indexed.) However, to specify which of the custom fields are indexed, your developers must customize certain elements for the asset type.

As the administrator, you specify which asset types can be found by the search engine, but it is the developers who determine which of the custom fields for assets of that type are indexed.

For information about designing basic asset types, see the “Data Design” section in the *CSEE Developer’s Guide*.

Flex and Flex Parent Assets

Flex and flex parent assets also have two kinds of fields:

- Flex attributes – the “custom” fields for a flex asset are flex attributes. Your developers create the flex attributes they need in order to define flex and flex parent assets.
- Default fields – in addition to the flex attributes that hold values for flex and flex parent assets, the asset types have the same default fields that all asset types have: Name, Description, Status, Updated By, and so on.

When you enable a flex or flex parent asset for the **Searching** option, all of their default fields that are not drop-down fields are indexed. However, to specify which of their fields—that is, flex attributes and their values—are indexed, your developers must use the **Search Engine** field on the flex attributes’ **New** and **Edit** forms to specify which attribute values should be indexed for the flex and flex parent assets that use them.

Other Flex Family Members

For all the other members in a flex family—flex attribute, flex definition, flex parent definition, flex filter—when you enable the **Searching** option, all of their fields that are not drop-down fields are indexed.

Location of Indexes

Indexes are stored in two different locations:

- The indexes for flex attribute values of flex assets are stored in the directory specified by the `mwb.searchdir` property in the `gator.ini` file.
- For all other asset types (including flex attributes when they are indexed as assets), indexes are stored in the directory specified by the `xcelerate.sePath` property in the `futuretense_xcel.ini` file.

Configuring Asset Types for Your Search Engine

To specify that assets of a specific type should be enabled for the search engine, you select the asset type and invoke the search engine to build the index for existing assets of that type. From that point on, each asset of that type is indexed when it is saved.

Enabling Asset Types for Search

Complete the following steps:

1. In the Content Server interface, select the **Admin** tab and then expand the **Searching** item.
2. Double-click on the asset type.

   Note

   Before continuing to the next step, examine the form and verify that this asset type has not been enabled for the search engine yet.

3. In the search form that is displayed, click **Enable Search Engine**.

   A message lists the directory where the index for this asset type will be created. You can change the directory name if you want.
4. Click **Enable Search Engine**.

   Note

   Building the index can take a long time, depending on the number of assets of that type in the database.

5. Repeat steps 2 through 4 for each asset type that you want to enable.

Disabling Asset Types for Search

To disable an asset type, complete the following steps:

1. In the Content Server interface, select the **Admin** tab and then expand the **Searching** item.
2. Double-click on the asset type.
3. In the search form, click **Disable**.

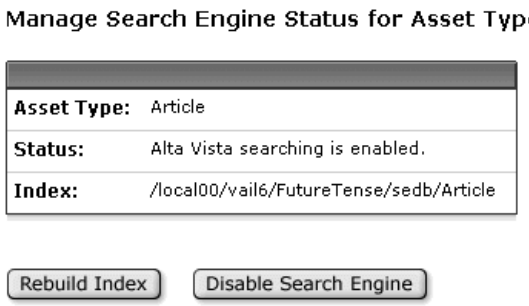
Searches for assets of this type will now use the native SQL-based database search method rather than the rich-text search from the search engine.

Rebuilding an Index

If the index that the search engine uses for an asset type has been damaged or corrupted, you can rebuild it by completing the following steps:

1. In the Content Server interface, select **Admin tab > Searching**.
2. Under **Searching**, double-click the name of the asset type whose index you want to rebuild.

   CS-Direct displays the following form:

3. Click the **Rebuild Index** button.
4. In the confirmation message, click **Rebuild Index** again.

   The search engine module deletes the existing index and builds a new one. Note that this can take a long time, depending on how many assets exist of this type.

Chapter 11

Properties and Property Files

Many of your configuration tasks for Content Server and the CSEE applications require you to set or modify values for properties found in initialization or property files. These files have the suffix `.ini`. Each CSEE system has a set of these files, typically located in its `$HOME/`*`Installation`* directory.

There are one or more `.ini` files associated with each of the CSEE applications. Content Server provides a utility called the Property Editor that you can use to open these files and modify their property values.

This chapter lists all of the properties in all of the property files that are delivered with the applications in Content Server Enterprise Edition. It contains the following sections:

- Using the Property Editor
- Property Files by Product
- Analysis Connector Property Files
- Content Server Property Files
- CS-Direct Property File
- CS-Direct Advantage Property Files
- CS-Engage Property File
- CS-Satellite Property Files

Using the Property Editor

Content Server provides a utility called the Property Editor. You use it to set or modify the values of properties in property files.

FatWire recommends that you always use the Property Editor to set values in the CSEE property files, because using other means to edit property files can create problems, especially with properties that have encrypted fields.

Starting the Property Editor

To start the Property Editor, execute the following scripts at the MS DOS prompt or in a UNIX shell:

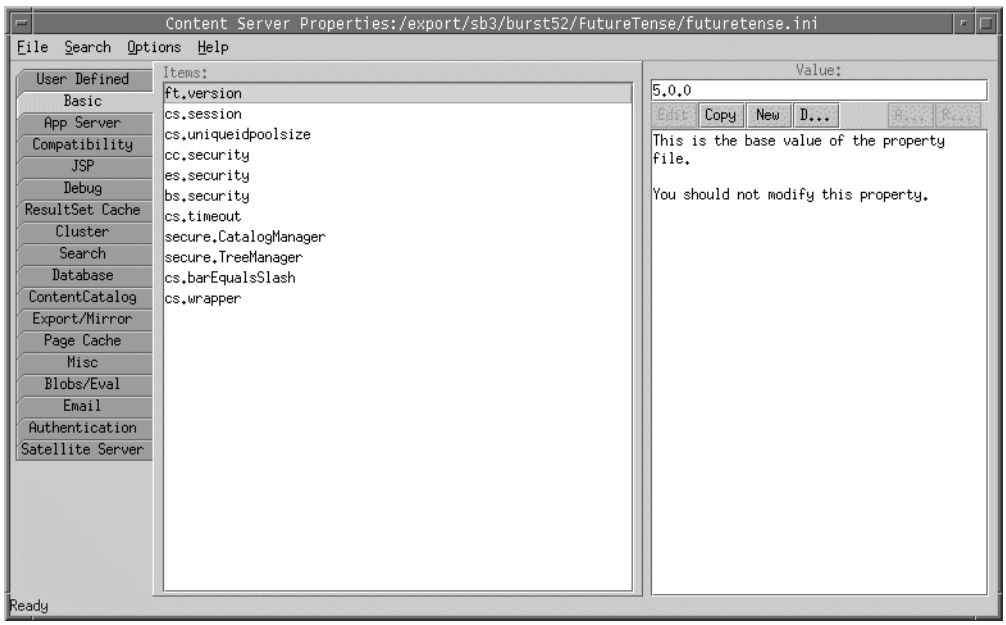
- Windows: `propeditor.bat`, which is usually located in *`installdir/`*
- Solaris: `propeditor.sh`, which is usually located in `$HOME/`*`installDir`*

Setting Properties

To set properties for the CSEE products, complete the following steps:

1. Start the Property Editor, as previously described in "Starting the Property Editor."
2. Select **File** > **Open**.
3. Browse to the `.ini` file that you need to modify and then select it.

   The Property Editor opens the file. For example, this is what the `futuretense.ini` file looks like when it is first opened in the Property Editor:



   The tabs on the left represent functional groups of properties.

   The **Items** pane lists the properties in the selected tab (functional group).

The **Value** pane lists the current value for the selected item, a brief description of the item, and the acceptable values for it.

4. Select the tab that represents the functional group that contains the property that you want to configure.

   The Property Editor displays the properties from that tab in the **Items** pane.

5. Select the property from the **Items** pane.

   The Property Editor displays the value that is currently set for that property value and a provides a brief description of the property in the **Values** pane.

6. In the **Values** pane, enter the value for the property in the text field at the top.
7. Click **Accept**.
8. Repeat steps 4 through 7 for each property that you want to configure.
9. When you are finished, select **File > Save**.
10. Select **File > Close**.
11. Stop and restart the application server so the new values can take effect.

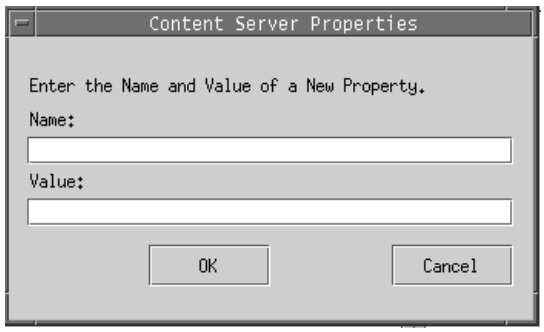
Adding Properties

Some configuration tasks require you to add new, custom properties for your system. For example, when you implement resultset caching, you can create up to three properties for any Content Server database table that you want to set caching values for. (For information about resultset caching, see the *CSEE Developer's Guide*.)

To add a property, complete the following steps:

1. Start the Property Editor, previously described in "Starting the Property Editor" on page 276.
2. Select **File > Open**.
3. Browse to the `.ini` file that you need to add a property for.
4. (Optional) Select the **User Defined** tab. (This step is optional because any property that you create is displayed on the **User Defined** tab the next time you open this `.ini` file in the Property Editor no matter which tab you select.)
5. In the **Values** pane, just under the text entry field, click **New**.

   The Property Editor displays the **Content Server Properties** dialog box:



6. Click in the **Name** field and enter the name of the new property.

7. Click in the **Value** field and enter the value for the new property.
8. Click **OK**.

   The new property appears in the **Items** pane and the value appears in the **Values** pane.

   Note

   If you did not select the **User Defined** tab in step 4, the new property might appear on the tab that was selected when you clicked **New** in step 5. This is temporary. The next time that you open this `ini` file in the Property Editor, the property is displayed on the **User Defined** tab.

9. Select **File** > **Save**.

Deleting Properties

It is unlikely that you will ever have to delete a property. However, if you do have to delete a property, complete the following steps.

Caution

Never delete a required property.

1. Start the Property Editor. (See "Starting the Property Editor," above.)
2. Select **File** > **Open**.
3. Browse to the .ini `file` that you need to modify by deleting a property and select it.
4. Select the tab that represents the functional group that holds the property that you want to delete.
5. Take note of the current value for this property just in case you need to restore it for any reason.
6. In the **Values** pane, just under the text entry field, click **Delete**.

   The Property Editor displays a confirmation message.
7. Click **Yes**.
8. The property is deleted from the `.ini` file.
9. Select **File** > **Save**.

Property Files by Product

This table lists all the CSEE property files, organized by the product that they are associated with:

Product	Files

Analysis Connector	`commercedata.ini`
	`databaseloader.ini`
	`writequeue.ini`

Content Server, which includes CS-Satellite (co-resident), CS-Bridge XML, and the user manager plug-ins	`batch.ini`
	`dir.ini`
	`futuretense.ini`
	`jsprefresh.ini` (WebLogic only)
	`ldap.ini`
	`logging.ini`
	`satellite.ini`
	`xmles.ini`

CS-Direct	`futuretense_xcel.ini`

CS-Direct Advantage	`assetframework.ini`
	`catalog.ini`
	`gator.ini`
	`transact.ini`
	`visitor.ini`

CS-Engage	`ms.ini`
CS-Satellite, remote server version	`resin.conf` `satellite.ini`

Analysis Connector Property Files

Analysis Connector installs the following property files:

- commercedata.ini
- databaseloader.ini
- writequeue.ini

commercedata.ini

The `commercedata.ini` file holds properties that Analysis Connector uses to define and manage its logging tables. By default, Analysis Connector installs five logging tables: `cart_event`, `orders`, `order_item`, `shopper`, and `shopper_map`. The `commercedata.ini` file contains the properties that govern those tables as well as global properties that specify settings for Analysis Connector.

Note

If you are not using the default Analysis Connector logging tables, remove their names from the `cd.tablenames` property (described below).

Properties that are specific for a logging table have that table's name as a prefix. Global properties have the prefix `cd`.

Following is a list of these properties in alphabetical order:

Property	Description
`cd.ageinterval`	Age data collected by Analysis Connector can be stored in intervals. This property specifies the interval space of the age interval. Default: 10
`cd.agelowest`	Specifies the lower limit of the age intervals. Default: 5
`cd.CDAliasName`	The name of the persistent cookie to be set. Default: `AnalConnCookie`
`cd.incomeinterval`	Income data collected by Analysis Connector can be stored in intervals. This property specifies the interval space of the income interval. Default: 10000
`cd.incomelowest`	Specifies the lower limit of the income intervals. Default: 0

Property	Description

`cd.maritalID`*n*	There are several Marital ID properties, one for each type of Marital ID. The number of these properties is specified by the value for the `cd.totalmaritalIDs` property, which is 7 by default.
	When the default for the `cd.totalmaritalIDs` property is 7, the following properties and the corresponding default values are required in this property file:
	`cd.maritalID1 Single`
	`cd.maritalID2 Married`
	`cd.maritalID3 Common Law`
	`cd.maritalID4 Separated`
	`cd.maritalID5 Divorced`
	`cd.maritalID6 Widowed`
	`cd.maritalID7 Other`

`cd.money`	The database-independent datatype to store the `MONEY` type data. Default: `NUMERIC(20,3)`
`cd.productatr`	The name of the attribute to use for a one-tier parent hierarchy. Default: `AnalConnAtr`
`cd.tablenames`	A list of the names of all the logging tables that will be created when you invoke the `CreateTable` element. Each table name should be separated by a space. If the system has the Burlington Financial Extension sample site installed, the following tables were created during the installation and the value for this property is: `clickstream searchpref recpromo` See also the *`table`*`#columns` property. Be sure to remove the names of the default Analysis Connector tables from this list if you do not plan to use them. Those table names are: `cart_event`, `orders`, `order_item`, `shopper`, and `shopper_map`.
`cd.totalmaritalIDs`	The total number of Marital IDs. Default: 7 For each ID, there is a property named `cd.maritalID`*n* , where *n* ranges from 1 to the number set with this property.
`comdata.path`	The directory where CommerceData is installed. The default is the installation directory. Do **not** change the value of this property.

Property	Description
table#columns	For each of the tables specified by the cd.tablenames property, a corresponding property that specifies the number of columns. When you run the CreateTables element, it uses this data to create the table. For example, the property for the shopper table for the Burlington Financial Extension sample site is shopper#columns and it is set to 23. Then, for each of the 23 columns, there are two more properties: *table*colname*n* and *table*colvalue*n* , where *n* ranges from 1 to the number set with this property.
*table*colname*n*	Specifies the name of a column in a logging table. For each column in a table, this property must specify the name of that column, where *table* is the name of the table and *n* is the column number. For example, for the shopper table, there are 23 of these properties: shoppercolname1 through shoppercolname23. There is a corresponding *table*colvalue*n* property and a *table*collnul*n* property for each *table*colname*n* property. Therefore, for the shopper table, there are 23 corresponding *table*colvalue*n* properties (shoppercolvalue1 through shoppercolvalue23) and 23 corresponding *table*colnull*n* properties (shoppercolnull1 through shoppercolnull23). See Table , "Example tablecolnamen and tablecolvaluen Properties for the shopper Table" on page 283 for examples of the default properties for the shopper table.
*table*colnull*n*	Specifies whether a column in a logging table can have a null value, using the convention *table*colnull*n* where *table* is the name of the table and *n* is the column number. Possible values: NULL | NOTNULL

Property	Description
*table*colvalue*n*	Specifies the field type of a column in a logging table. For each column identified with a *table*colname*n* property, there is a corresponding *table*colvalue*n* property. For example, for the shopper table, there are 23 of these properties: shoppercolvalue1 through shoppercolvalue23. See Table , "Example tablecolnamen and tablecolvaluen Properties for the shopper Table" on page 283 for examples of the default properties for the shopper table.

The following table lists sample values the default shopper table:

Example *table*colname*n* and *table*colvalue*n* Properties for the shopper Table

Name Property	Default	Value Property	Default
shoppercolname1	shopper_id	shoppercolvalue1	BIGINT
shoppercolname2	shopper_name	shoppercolvalue2	VARCHAR(80)
shoppercolname3	shopper_type_id	shoppercolvalue3	INT
shoppercolname4	gender	shoppercolvalue4	CHAR(1)
shoppercolname5	age	shoppercolvalue5	VARCHAR(16)
shoppercolname6	income	shoppercolvalue6	VARCHAR(16)
shoppercolname7	marital_status	shoppercolvalue7	VARCHAR(16)
shoppercolname8	nbr_children	shoppercolvalue8	INT
shoppercolname9	hhold_size	shoppercolvalue9	INT
shoppercolname10	company	shoppercolvalue10	VARCHAR(255)
shoppercolname11	area_code	shoppercolvalue11	VARCHAR(255)
shoppercolname12	zip_code	shoppercolvalue12	VARCHAR(255)
shoppercolname13	city	shoppercolvalue13	VARCHAR(255)
shoppercolname14	state	shoppercolvalue14	VARCHAR(255)
shoppercolname15	region	shoppercolvalue15	VARCHAR(255)
shoppercolname16	country	shoppercolvalue16	VARCHAR(255)
shoppercolname17	email	shoppercolvalue17	VARCHAR(255)
shoppercolname18	transactions	shoppercolvalue18	INT

Name Property	Default	Value Property	Default
shoppercolname19	total_sales	shoppercolvalue19	NUMERIC(20,3)
shoppercolname20	lastvisit	shoppercolvalue20	BIGINT
shoppercolname21	lastorder	shoppercolvalue21	BIGINT
shoppercolname22	date_hour	shoppercolvalue22	BIGINT
shoppercolname23	last_update	shoppercolvalue23	BIGINT

databaseloader.ini

The properties in this file set values for properties that the DatabaseLoader process uses. You can set up DatabaseLoader to run at regularly scheduled times or you can invoke it manually.

Property	Description
dbl.cdfile	The full path to the commercedata.ini file that contains commercedata table definitions. Default: c:/FutureTense/commercedata.ini
dbl.loginpassword	The password for database access. Default: FutureTense Be sure to change the default password.
dbl.loginurl	The URL to post data from disk files to load in to the database. Default: http://localhost/servlet/CatalogManager
dbl.loginuser	The user account to use for database access. Default: ContentServer Be sure to change the default user account.
dbl.qfile	The full path to the writequeue.ini file that contains necessary information about data files. Default: c:/FutureTense/writequeue.ini

writequeue.ini

The properties in this file configure the queue that Analysis Connector uses to write visitor or commerce data to files on the disk.

Property	Description
`q.debug`	The debug setting on the queue. Default: `true`.
`q.fileextention`	The file extention for files created by the queue. Default is `.bkt`
`q.numminutestillstale`	Specifies when the queue should be automatically written to the disk. The value is in minutes. Default: 10
`q.numrotatingfiles`	Specifies the number of files to hold in the queue. Default:10
`q.numrowsinfile`	Specifies the number of rows of data that can be held in each file in the queue. Default: 1
`q.storagefilepath`	Specifies the path to the storage directory in which the queue puts files. Default: `c:/FutureTense/queue/`

Content Server Property Files

Content Server installs the following property files:

- batch.ini
- dir.ini
- futuretense.ini
- jsprefresh.ini (WebLogic Only)
- ldap.ini
- logging.ini
- satellite.ini
- xmles.ini

This section defines all of the properties in these files.

batch.ini

A CSEE system uses background, batch processes for various reasons (publishing, for example). The `batch.ini` file provides configuration information for batch processes. It has the following tabs:

- Configuration (batch.ini)
- Debug (batch.ini)
- Results
- Security (batch.ini)

Configuration (batch.ini)

The **Configuration** tab holds properties that configure settings for the threads that the batch processes use.

Property	Description
`thread.count`	Specifies the number of dispatcher threads to allocate and manage in the pool. Default value: 32
`thread.growcache`	Specifies whether additional dispatcher threads (in excess of the number specified by `thread.count`) can be allocated to the pool if they are needed. Possible values: `true` and `false` Default value: `false`
`thread.idle`	Applies only when `thread.growcache` is set to `true`. Specifies the number of seconds a dispatcher thread can remain idle before it is released by the pool. Default value: 10
`thread.wait`	Applies only when `thread.growcache` is set to `false`. Specifies the number of seconds that a batch process waits for a free dispatcher thread before it reports an error because it cannot complete its task. Default value: 15

Debug (batch.ini)

Property	Description
`debug`	Specifies whether debugging for batch processes is enabled or disabled. If you set this value to `true`, messages about the status of batch processes are written to the `futuretense.txt` file. Default value: `false`

Results

Property	Description
`request.folder`	Specifies the location of the file that stores information about the results of batch processes. For example, the CS-Direct publishing system uses this directory to hold the publishing log files. Default value: `./dispatcher/`

Security (batch.ini)

Property	Description
`security.class`	Specifies the name of the class file that is used for security checks. The default is provided for reference only: `com.openmarket.Batch.DefaultSecurity` Do **not** change the value of this property.

dir.ini

Content Server delivers a directory services API that enables your CSEE system to connect to directory servers that contain authentication information, user information, and so on.

CSEE provides three directory services options (implemented through the directory services API) for managing user information, one of which your CSEE system is configured to use:

- The Content Server directory services plug-in, which uses the native Content Server user management tables (`SystemUsers` and `SystemUserAttrs`).
- The LDAP plug-in. When you use this option, user names and attributes are stored in your directory server rather than in the Content Server database.
- The NT 4.0 plug-in, which uses the Contnet Server user management tables, but authenticates users through NT authentication.

The `dir.ini` file contains the properties that configure the directory services plug-ins. It has the following tabs:

- Attribute Names
- Global Data
- Interface Implementations
- JNDI SPI Env.
- Naming Syntax
- Schema Defaults
- Search Controls

Although the `dir.ini` file is the main configuration file for the directory services API, there are additional user manager/directory services properties in two other property files. See also the following sections in this chapter:

- "ldap.ini" on page 335, which describes the properties in the `ldap.ini` file
- "Authentication" on page 297, which describes the properties on the `authentication` tab of `futuretense.ini`

Attribute Names

The **Attribute Names** tab holds attribute-mapping properties. You use these properties to specify how a user attribute that CSEE uses is identified in the directory server.

Property	Description
`cn`	Specifies the name of the attribute in the directory server that serves as the group name attribute. Possible values: • Content Server and NT: `cn` • LDAP, iPlanet: `cn` • LDAP, Active Directory: `cn` **Note**: If you are using LDAP, be sure that the value that you enter here exactly matches the value set for the `cn` property in the `ldap.ini` file.
`password`	Specifies the name of the attribute in the directory server that serves as the password attribute. Possible values: • Content Server and NT: `password` • LDAP, iPlanet: `userPassword` • LDAP, Active Directory: `password`
`loginattribute`	Specifies the attribute that a user logs in with. Possible values: • Content Server and NT: `uid` • LDAP, iPlanet: `uid` • LDAP, Active Directory: `cn`

Property	Description
uniquemember	Specifies the name of the attribute in the directory server that serves as the group assignment attribute. Possible values: • Content Server and NT: uniquemember • LDAP, iPlanet: uniquemember • LDAP, Active Directory: member **Note**: If you are using LDAP, be sure that the value that you enter here exactly matches the value set for the uniquemember property in the ldap.ini file.
username	Specifies the name of the attribute in the directory server that serves as the user name attribute. Possible values: • Content Server and NT: username • LDAP, iPlanet: uid • LDAP, Active Directory: sAMAccount **Note**: If you are using LDAP, be sure that the value that you enter here exactly matches the value set for the username property in the ldap.ini file.

Compatibility (dir.ini)

The Compatibility tab holds properties that determine how any strings that are extracted from the directory server and stored in the Content Server database are treated.

Property	Description
cleandns	Specifies how the strings for distinguished names are stored in the Content Server database. If set to true, the Directory Services API extracts distinguished names from the directory server, removes extra spaces from the names, and then changes all the upper-case letters to lower-case letters before storing the strings in the Content Server database. Possible values: true | false Default value: false Note: do not set this value to true if you are upgrading from an earlier version of Content Server. If you do, you must manually change any existing dns strings that are stored in the Contet Server tables. Also, if you set it to true, you must also verify that the syntax.ignorecase property is also set to true.

Global Data

The Global Data tab holds properties that determine global values for all users.

Property	Description
`baseDN`	Specifies the distinguished name for the root to use by default for searches and for prepending to the names for attribute values that require a DN type. Default value: blank Do **not** change the value of this property. Because the authentication module sets the currentUser session variable to a fully qualified name, CS-Direct assumens that all names returned from search are fully qualified.
`groupparent`	Specifies the entry to use as the parent of all Content Server entries of type `Group`. Possible values: • Content Server and NT: `ou=groups` • LDAP, iPlanet: `ou=groups,dc=`*`companyname`*`,dc=com` • LDAP, Active Directory: `cn=groups,dc=`*`companyname`*`,dc=com` **Note**: If you are using LDAP, be sure that the value that you enter here exactly matches the value set for the `LDAPGroupsBase` property in the `ldap.ini` file.
`peopleparent`	Specifies the entry to use as the parent of all Content Server entries of type `User`. Possible values: • Content Server and NT: `ou=people` • LDAP, iPlanet: `cn=people,dc=`*`companyname`*`,dc=com` • LDAP, Active Directory: `cn=users,dc=`*`companyname`*`,dc=com` **Note**: If you are using LDAP, be sure that the value you enter here exactly matches the value set for the `LDAPUserBase` property in the `ldap.ini` file.

Interface Implementations

The **Interface Implementations** tab holds two properties that determine which user manager module your CSEE system is using. The values of the rest of the properties on the tab should never be modified.

Property	Description
`className.Attribute`	Specifies the name of the concrete class to implement the interface Attribute. Do **not** change the value of this property.
`className.Attributes`	Specifies the name of the concrete class to implement the interface Attributes. Do **not** change the value of this property.
`className.IDir`	With the `className.IName` property, specifies which user manager module your system is using. Possible values: • Content Server: `com.openmarket.directory.cs.CSDir` • LDAP: `com.openmarket.directory.jndi.JNDIDir` Do **not** change the value of this property after the installation.
`className.Ifactory`	Specifies the name of the concrete class to implement the interface Ifactory. Do **not** change the value of this property.
`className.IName`	With the `className.IDir` property, specifies which user manager module your system is using. Possible values: • Content Server: `com.openmarket.directory.cs.CSName` • LDAP: `com.openmarket.directory.jndi.NameWrapper` Do **not** change the value of this property after the installation.
`className.JNDIName`	Specifies the name of the concrete class to implement the interface JNDIName. Do **not** change the value of this property.

JNDI SPI Env.

The properties on the **JNDI SPI Env** tab are used only if your CSEE system is configured to use the LDAP user manager module.

Property	Description
`java.naming.factory.initial`	System property. Do **not** change the value of this property.
`java.naming.security.authentication`	Specifies the security level to use. Its value is one of the following strings: `none`, `simple`, `strong`. If this property is unspecified, the security level is determined by the service provider. Default value: `simple`
`jndi.baseURL`	Specifies the server name and port number of the directory server. The value uses the following format: `ldap://`*`hostname`*`:`*`port`*
`jndi.connectAsUser`	Specifies whether Content Server needs a designated user account to query the directory server for user attribute information. If set to `true`, Content Server can query the directory server for information as the user who is logged in to CSEE and is making the inquiry. For example, when an administrator examines user information in the Content Server interface, Content Server can make the inquiry as that user (admin, for example.) If set to `false`, there must be a valid username/password combination specified for the `jndi.login` and `jndi.password` properties; Content Server uses that user account to make inquiries.
`jndi.custom`	System property. Do **not** enter a value for this property.
`jndi.login`	Applies only when `jndi.connectAsUser` is set to `false`. Specifies the fully qualified, fully distinguished name of the user account that Content Server uses to query the directory server.
`jndi.password`	Applies only when `jndi.connectAsUser` is set to `false`. Specifies the password of the user account that Content Server uses to query the directory server. This value is encrypted.

Naming Syntax

The **Naming Syntax** tab holds properties that determine how strings for user attributes and their values are interpreted.

Property	Description
`syntax.beginquote`	Specifies the string that delimits the beginning of a quoted string.
`syntax.beginquote2`	Specifies an alternative to the value specified for the `syntax.beginquote` property.
`syntax.custom`	If your system is using a custom JNDI implementation, specifies additional variables that can be passed to the `classIName` constructor. The syntax follows the `x-www-form-urlencoded` format.

`syntax.direction`	Specifies the direction that the components in a designated name are read in.
	Possible values:
	`left_to_right`
	`right_to_left`
	`flat`
	Default value: `left_to_right`

`syntax.endquote`	Specifies the string that delimits the end of a quoted string.
`syntax.endquote2`	Specifies an alternative to the value specified for `syntax.endquote`.
`syntax.escape`	Specifies the escape string for overriding separator, escapes, and quotes.
`syntax.ignorecase`	Specifies whether strings are case-sensitive or not. Set to `false` if the uppercase and the lowercase version of a letter character should be considered as different characters. (That is, “admin” and “Admin” should be interpreted as different words.) Set to `true` if you want the uppercase and the lowercase version of a letter character to be considered as the same character. (That is “admin” and “Admin” should be interpreted as the same string.) Default value: `true` **Note**: if you need to set the `cleandns` property on the **Compatiblity** tab to `true`, you must also set this property’s value to `true`.

Property	Description
`syntax.separator`	Specifies the separator character used between atomic name components. This property is required unless `syntax.direction` is set to a value of `flat`.
`syntax.separatorava`	Specifies the separator character used to separate multiple attribute/value pairs. Typically the comma character (,) is used.
`syntax.separatortypeval`	Specifies the separator character used to separate an attribute from its value. For example, the equals symbol (=) is used.
`syntax.trimblanks`	Specifies whether spaces and whitespace characters are significant or should be ignored (trimmed) when evaluating a string. Set to true if spaces should be ignored. Set to false if spaces should be considered when evaluating a string.

Schema Defaults

The **Schema Defaults** tab holds properties that identify to Content Server the directory server attributes that users must have values for in order to be valid users as well as any attribute values that are assigned to users by default (if any).

Property	Description
`defaultGroupAttrs`	Specifies the attribute name/value pairs that are set for every descendent of the entry specified by the `groupparent` property. CS-Direct uses this information to create the default users that it needs during the installation, which means that this property must be set **before** you install CS-Direct. Values must be entered in the x-www-form-urlencoded format.
`defaultPeopleAttrs`	Specifies the attribute name/value pairs that are set for every descendent of the entry specified by the `peopleparent` property. CS-Direct uses this information to create the default users that it needs during the installation, which means that this property must be set **before** you install CS-Direct. Values must be entered in the x-www-form-urlencoded format.

Property	Description
objectclassGroup	Specifies the name of the base object that signifies a Content Server group. The DIR.GROUPMEMBERSHIPS tag uses the value set for this property to differentiate group entries from user or other entries. Possible values: • Content Server: groupofuniquenames • LDAP, iPlanet: groupofuniquenames • LDAP, Active Directory: group
objectclassPerson	Specifies the name of the base object that signifies a Content Server user (person). The DIR.LISTUSERS tag uses the value set for this property to differentiate user entries from group or other entries Value for Content Server or LDAP: person
requiredGroupAttrs	Specifies the attributes that every descendent of the entry specified by the groupParent property must have values for. Values must be entered in the x-www-form-urlencoded format.
requiredPeopleAttrs	Specifies the attributes that every descendent of the entry specified by the peopleParent property must have values for. Values must be entered in the x-www-form-urlencoded format.

Search Controls

The **Search Controls** tab holds properties that constrain the queries that the user manager plug-in makes to the directory server.

Property	Description
search.returnLimit	Specifies the maximum number of entries to return. To obtain all the entries that satisfy the search criteria, set the value to 0.
search.scope	Specifies to what depth in the hierarchy a search reaches: search just the specified or current node, or search the nodes under that node. Default value: 2, which means search all nodes under the stated node.

Property	Description
`search.timeoutVal`	Specifies the number of milliseconds to wait for results before returning an error. If the value is 0, it means to wait indefinitely (that is, wait until the network timeout limit ends the wait).

futuretense.ini

The `futuretense.ini` file is the main property file for Content Server. This section presents the properties in the `futuretense.ini` file grouped by the tab that represents their functional group. The tabs are documented here in alphabetical order:

- App Server
- Authentication
- Basic
- Blobs/Eval
- Cluster
- Compatibility
- ContentCatalog
- Database
- Debug (futuretense.ini)
- Email
- Export/Mirror
- JSP
- Misc
- Page Cache
- ResultSet Cache
- Satellite Server (futuretense.ini)
- Search
- User Defined (futuretense.ini)

App Server

The **App Server** tab holds the `futuretense.ini` properties that supply information to Content Server about the application server.

The following table lists all of the properties on the **App Server** tab in alphabetical order:

Property	Description
`cs.eventhost`	The host string for running the event engine on application servers. Enter the host and port number, as in the following example: `http://localhost:80`
`ft.cgipath`	The web server CGI directory where Content Server objects are installed. Used in the constructions of URLs and form actions. Be sure the value ends with a backslash (/). Possible values: `/NASApp/CS/` - when using the iPlanet application server (iAS). `/servlet/` - in http servlet application environments (such as WebLogic, WebSphere, and Sun ONE).

Authentication

The **Authentication** tab holds user authentication properties that are configured during installation based on the user manager plug-in in use on your CSEE system. Some of these properties apply no matter which user management module you are using while others apply only if you are using NT authentication.

See also, the following:

- The main configuration file for the user manager plug-ins is the `dir.ini` file. See “dir.ini” on page 287
- If you are using the LDAP user manager plug-in on your CSEE system, see also “ldap.ini” on page 335.

The following table presents the user authentication properties in alphabetical order:

Property	Description
`cs.manageACL`	Specifies the class that replaces the default Content Server ACL name-to-privilege mask function. Do **not** change the value of this property.

Property	Description
`cs.manageproperty`	Specifies the name of the appropriate property file that configures the NT authentication plug-in or the LDAP plug-in, depending on whether you are using either of these user manager modules. • For NT, set the value to: `futuretense.ini` • For LDAP, set the value to: `ldap.ini`
`cs.manageUser`	Specifies which user manager plug-in to use with this CSEE system. • If you are using the default Content Server plug-in, the value is blank. • If you are using LDAP, the value is: `COM.FutureTense.LDAP.ValidateLogin.LDAPLogin` • If you are using NT authentication, the value is: `COM.FutureTense.NTUserGroups.ValidateLogin.NTUserGroupsLogin` This property was set when your CSEE system was installed. Do **not** change it after installation.
`cs.manageUserAccess`	Specifies the class that replaces the default Content Server user-to-privilege-by-resource lookup functionality. Do **not** change the value of this property.
`cs.manageUserSystem`	Applies only when your CSEE system is using NT authentication (that is, `cs.manageUser` points to the NT plug-in). Specifies a comma-separated list of NT domain names that Content Server uses to authenticate users. Authentication is done in the order specified by the list of domains. A user is declared a valid user if his username/password combination is found in any of those domains. Specify the local system with a period (.) character. If the `cs.manageUser` points to the NT plug-in, but there are no domain names set for this property, Content Server attempts authentication on the local NT domain only. This value was set during installation.
`ntlogin.DefaultACL`	NT user manager plug-in only. A comma-separated list of any ACLs that should be assigned to all users by default. By default, this value is blank.

Property	Description
`ntlogin.DefaultReaderACL`	NT user manager plug-in only. The ACL list to be assigned to the account that is used as the default reader account. By default, this value is blank.
`ntlogin.DefaultReaderID`	NT user manager plug-in only. The user name to be used as the default reader account. By default, it is set to `DefaultReader`. This value can be blank.
`ntlogin.DefaultReaderPW`	NT user manager plug-in only. The password for the user name that the NT authentication module uses as the default reader account. Required if there is a value for `ntlogin.DefaultReaderID`. By default, it is set to `SomeReader`. The value is encrypted.
`ntlogin.LogFile`	NT user manager plug-in only. The complete path to the file where debug information from the NT authentication module should be written. (Used only when `ntlogin.Logging` is set to `true`.)
`ntlogin.Logging`	NT user manager plug-in only. Enables or disables debugging for the NT authentication module. Possible values are `true` | `false`.

Basic

The **Basic** tab holds the `futuretense.ini` properties that control such things as security settings, session timeouts, and Global Unique Identifiers that the Content Server servlets use.

The following table lists all of the properties on the **Basic** tab in alphabetical order:

Property	Description
`bs.security`	Specifies whether the BlobServer servlet checks security before allowing database access and image retrieval. If security is on, images cannot be cached in memory. If you enable BlobServer security, the BlobServer servlet serves the data only if the `csblobid` parameter exists in the URL and its value matches a session variable of the same name, as in the following example: `<IMG SRC="BlobServer?blobtable=MovieImages&#38;blobcol=urlimage&#38;blobkey=id&#38;csblobid=SessionVariables.blobid&#38;blobwhere=25"/>` Possible values: `true` | `false`.
`cc.security`	Specifies whether Content Server checks security before allowing database access. This property should always be set to `true` except in special cases. Possible values: `true` | `false`.
`cs.barEqualsSlash`	Specifies whether an Internet Explorer browser should interpret the bar (|) character as a forward slash (/) when it is included in a page name. Possible values: `true` | `false` For example, when set to `true`, Internet Explorer interprets `pagename=folder|subfolder|page` as the same page as `pagename=folder/subfolder/page` Default value: `false`
`cs.session`	Specifies whether Content Server starts and maintains a browser session for each user. Possible values: `true` | `false`. Cannot be set to `false` when the `cc.security` property is set to `true`.

Property	Description
`cs.timeout`	Specifies the number of seconds a connection can remain idle before the application server logs out this connection, which ends a browser session. Idle time is the time between Content Server http requests. Default value: 300 (5 minutes) **Note**: When the approval system approves assets, it is not a background process. Therefore, if you use the **Approve Multiple Assets** feature, be sure to set this property to a value that is greater than the amount of time it takes to approve a batch of asset so that the browser session does not time out. You will have to experiment with this setting, but you can start by setting it to 1800 seconds (30 minutes).
`cs.uniqueidpoolsize`	Specifies the number of unique and cluster-safe ID numbers that are cached at one time. (Content Server generates unique IDs for every row in any database table.) Default value: 100
`cs.wrapper`	Specifies whether the Content Server HTML wrapper pages should (can) be used. Default value: `true` Set this value to `false` on a CSEE system in which the application server does not have HTTP access to the web server, or, if you have removed the directory that holds the wrapper pages for security reasons. See also "Properties That Configure Security Settings" on page 56.
`es.security`	**Deprecated**. Specifies whether the EvalServer servlet checks security before allowing database access. This property exists for backward compatibility. Leave it set to its default value of `false`.
`ft.version`	Specifies the version number of the Content Server application. Do **not** modify this value.

Property	Description
`secure.CatalogManager`	Specifies whether the DefaultReader user can access the CatalogManager servlet. Possible values: `true` or `false` During installation, this property is set to `false`. Be sure that this value is changed to `true` after the installation. For more information, see "DefaultReader, secure.CatalogManager, and secure.TreeManager" on page 55.
`secure.TreeManager`	Specifies whether the DefaultReader user can access the TreeManager servlet. Default value: `true` For more information, see "DefaultReader, secure.CatalogManager, and secure.TreeManager" on page 55.

Blobs/Eval

The Blobs/Eval tab holds properties that configure the BlobServer and EvalServer servlets:

- BlobServer serves blobs. It gathers a blob from a table and performs all the necessary security checks. When BlobServer serves a blob, it caches it in both the Content Server and Satellite Server.

  BlobServer properties have the prefix "`bs`".

- EvalServer is a little-used servlet that still remains in the Content Server product for backwards compatability. In general, you should leave all EvalServer properties set to their defaults.

  EvalServer properties have the prefix "`es`".

The following table presents the properties on the **Blobs/Eval** tab in alphabetical order:

Property	Description
`bs.bCacheSize`	Specifies the default number of blobs that can be cached (to memory). Default value: 100
`bs.bCacheTimeout`	Specifies the number of seconds that a blob is kept in the memory cache since it was last requested before it is flushed. Note that the memory cache is cleared whenever the BlobServer servlet is restarted. Default value: 300 (5 minutes)

Property	Description
`es.cacheCriteria`	**Deprecated** A parameter list that defines how the EvalServer identifies cached content. Leave the value for this property blank.
`es.cacheSize`	**Deprecated**. The number of cached memory-based data items, created by the EvalServer. Default: 100 Do not change this value.
`es.cacheTimeout`	**Deprecated**. Represents the duration, in seconds, of memory-based cached data created by the EvalServer. Default: 300 Do not change this value.

Cluster

The **Cluster** tab holds the properties that Content Server uses to communicate with all the servers in a cluster when a CSEE system is installed in a cluster.

These properties are presented here in alphabetical order:

Property	Description
`cc.cacheNoSync`	Specifies whether the system allows the database query transaction data cached by Content Server to persist even if a cluster member updates the table. Default value: `false` You can set table-specific values for individual tables by adding a property in the following format: `cc.<sometable>CSync=<true or false>`

Property	Description
ft.sync	An identifier or value that defines the synchronizer key for clustered servers; for example, the DSN that cluster members use for their shared database. Each cluster member must have the same ft.sync value. It is valid to leave this field blank; the field should be left blank during the installation. **Note:** You should not turn off ft.sync on systems with muliple Java virtual machines. Instead, you can set cc.cacheNoSync=false, or you can set a specific table as follows: cc.<sometable>CSync=false. For example, cc.ElementCatalogCSync=false turns off ft.sync for the ElementCatalog table.
ft.usedisksync	Specifies a shared file system folder to synchronize data across a cluster. Set this property to a valid folder when synchronization is turned on with the ft.sync property. For example, set it to a directory where you have read/write access.

Compatibility

The **Compatibility** tab holds properties that set values necessary for backward compatibility with earlier versions of Content Server. These properties are documented here in alphabetical order:

Property	Descriptions
cs.httpvariables	Specifies whether the Content Server variables that contain HTTP header information are created on each page request, as was necessary in pre-4.0 versions of the product. Starting with 4.0, Content Server provides built-in variables that perform the same function. Default value: false For best performance, leave this value set to false. If developers need to retrieve an http variable for a site page, they can retrieve the value of the ones they need by using the built-in variables.

Property	Descriptions
`cs.pgcachefolder`	For backwards compatibility. In previous versions, when cache information is specified in the `cacheinfo` column for a `SiteCatalog` page entry, it pointed to this property. This property specified the default directory location where Content Server pages would be cached. Starting in version 5.0, Content Server pages are cached in the database, not to a directory. The value in the `cacheinfo` column now starts with a value of `true` or `false`, which determines whether the page is cached or not. For backwards compatibility, if the value in the `cacheinfo` column of a SiteCatalog page entry follows the old syntax, the CacheManager caches Content Server pages using the old methodology. Therefore, if the `SiteCatalog` page entries on your system have not yet been updated to use the new syntax (which invokes the new caching behavior), this property must be set to a valid directory.
`ft.approot`	This property is no longer used. In some cases, it may be required for backward compatibility.
`ft.catalogmanager`	Defines the Global Unique Identifier (GUID) for the CatalogManager service. It is provided for reference only. Value: {40DD4E30-8DE2-11D1-8599-0080C7D07E91} Do **not** modify this value.
`ft.contentserver`	Defines the GUID for the Content Server servlet. It is provided for reference only. Value: {29434AD0-8DE2-11D1-8599-0080C7D07E91} Do **not** modify this value.
`ft.evalserver`	Defines the GUID for the EvalServer servlet. It is provided for reference only. Value: {834A7BD0-B2CD-11D1-86BC-0080C77014DF} Do **not** modify this value.
`ft.treemanager`	Defines the GUID for the TreeManager servlet. Do **not** modify this value.

Property	Descriptions
`security.checkpagelets`	Specifies whether Content Server checks security before allowing a user to view a pagelet that is nested in an enclosing page. Note that the `cc.security` property must also be set to `true` for security to be implemented. Default value: `true` If set to `false`, the following occurs: • On systems with co-resident CS-Satellite, security is not checked on any pages or pagelets, even with `cc.security` set to `true`. • On systems with stand-alone CS-Satellite or with Content Server only, security is checked on the first or outermost page but security is not checked on nested pagelets.

ContentCatalog

The **ContentCatalog** tab holds one property that specifies the default primary key column for all of the content tables (as opposed to object tables) in your Content Server database.

If you or your developers create any content tables to support your online sites, you can specify that a column other than the one specified by the default is the primary key column for those content tables by creating table-specific properties. Use the following format:

`cc.<name of table>Key=<name of column>`

For example, when CS-Direct installs the `Category` table (which is used by basic asset types), it creates a property named `cc.CategoryKey`. The `cc.CategoryKey` property and any new property that you create appears on the **User Defined** tab rather than the **ContentCatalog** tab.

Note

Be sure that you do not change the key value specified for any of the CS-Direct content tables.

The following table describes the content table property:

Property	Descriptions
`cc.contentkey`	Specifies the name of the column that serves as the primary key for content tables in the Content Server database. This is a default setting that applies to any content table that does not have a table-specific property that sets a different primary key for it. Value set during installation of the CSEE content applications: `id` **Caution:** Do **not** change the value of this property. If you change it, the CSEE content applications will not function. To specify a different primary key for an individual content table, create a table-specific key property as described in the paragraphs preceding this table.

Database

The **Database** tab holds the both the general database configuration properties, properties, such as database name and user access properties, as well as vendor-specific properties, such as how the database interprets date/time values.

The database properties must be set to the same values on each of the systems in your CSEE system—development, management, and delivery—so that you can move assets and other work from one system to another.

Note

Database properties are set during Content Server installation.

Do **not** change the values of these properties after CSEE is installed.

If you are unsure about how or why values were determined, check with your database administrator or whoever was responsible for installing Content Server

These properties are documented here in alphabetical order:

Property	Description
`cc.bigint`	Specifies the SQL string for defining a 64-bit integer field. Possible values: • Oracle: `NUMBER`(20) • SQL Server: `BIGINT` • DB2: `BIGINT` Do **not** change the value of this property.

Property	Description
cc.bigtext	Specifies the SQL string for defining a large text field. Possible values: • Oracle: CLOB • SQL Server: TEXT • DB2: LONG VARCHAR Do **not** change the value of this property.
cc.blob	Specifies the SQL string for defining a BLOB (binary large object) field. Possible values: • Oracle: BLOB • SQL Server: IMAGE • DB2: LONG VARCHAR FOR BIT DATA Do **not** change the value of this property.
cc.char	Specifies the SQL string for defining a CHAR data type. Possible values: • Oracle: CHAR • SQL Server: CHAR • SQL Server, multi-lingual Unicode: NCHAR • DB2: CHAR Do **not** change the value of this property.
cc.datepicture	Specifies how Content Server creates a date/time literal. Value: {ts '$date'} Do **not** change the value of this property.
cc.datetime	Specifies the SQL string for defining a date/time field. Possible values: • Oracle 8: DATE • Oracle 9: TIMESTAMP • SQL Server: DATETIME • DB2: TIMESTAMP Do **not** change the value of this property.

Property	Description
`cc.double`	Specifies the SQL string for defining a double field. Possible values: • Oracle: `NUMBER(38,10)` • SQL Server: `NUMERIC(28,10)` • DB2: `FLOAT` Do **not** change the value of this property.
`cc.forcelower`	Specifies whether the column names for the tables that Content Server creates have all lowercase letters. Possible values: • Oracle: `true` • SQL Server: `false` • DB2: `true` Do **not** change the value of this property.
`cc.ignoreTblCase`	Determines whether Content Server ignores case when assessing table names. Possible values: `yes` | `no` For example, if “tablename” and “TABLENAME” would be considered different tables in your database, set this value to `no`. Possible values: • Oracle: `yes` • SQL Server: `yes` • DB2: `yes` Do **not** change the value of this property.
`cc.integer`	Specifies the SQL string for defining a 32-bit integer field. Possible values: • Oracle: `NUMBER(10)` • SQL Server: `INT` • DB2: `INTEGER` Do **not** change the value of this property.
`cc.maxvarcharsize`	Specifies the maximum size of a varchar column for your database. Possible values: • Oracle: `2000` • SQL Server: `8000` • DB2: `4000` Do **not** change the value of this property.

Property	Description
`cc.null`	Specifies the SQL string for defining a field which allows NULL values; this is nonstandard, though most databases support NULL. Possible values: • Oracle: `NULL` • SQL Server: `NULL` • DB2: `blank` Do **not** change the value of this property.
`cc.numeric`	Specifies the SQL string for defining a numeric field. Possible Values: • Oracle: `NUMBER` • SQL Server: `NUMERIC` • DB2: `NUMERIC` Do **not** change the value of this property.
`cc.primary`	Specifies the SQL string that defines a primary key. Possible values: • Oracle: `PRIMARY KEY NOT NULL` • SQL Server: `PRIMARY KEY NOT NULL` • DB2: `PRIMARY KEY NOT NULL` Do **not** change the value of this property.
`cc.rename`	Specifies the SQL string that renames a table in the database, as required by your database vendor. Possible values: • Oracle: `rename %1 to %2` • SQL Server: `execute sp_rename %1,%2` • DB2: `rename %1 to %2` Do **not** change the value of this property.
`cc.smallint`	Specifies the SQL string for defining a 16-bit integer field. Possible values: • Oracle: `NUMBER(5)` • SQL Server: `SMALLINT` • DB2: `SMALLINT` Do **not** change the value of this property.

Property	Description
cc.stringpicture	Specifies how Content Server creates a string literal. Possible values: • Oracle: '$string' • SQL Server: '$string' • SQL Server, multi-lingual Unicode: N '$string' • DB2: '$string' Do **not** change the value of this property.
cc.unique	Specifies the SQL string for defining a unique field. Possible values: • Oracle: UNIQUE NOT NULL • SQL Server: UNIQUE NOT NULL • DB2: UNIQUE NOT NULL Do **not** change the value of this property.
cc.varchar	Specifies the SQL string for defining a VARCHAR data type. Possible values: VARCHAR, for all supported databases except SQL Server, multi-lingual Unicode which is set to NVARCHAR Do **not** change the value of this property.
cs.dbconnpicture	Specifies the format of the database connection string used by JNDI datasources: • WebLogic 6.1: $dsn • SunOne: jdbc/$dsn • WebSphere 4.0: $dsn Do **not** change the value of this property.
cs.dbtype	Defines the type of database you are connecting to. Do **not** change the value of this property.
cs.dsn	Contains the database JNDI data source name for connecting to your database. Do **not** change the value of this property.
cs.privpassword	Specifies the password for the database account name used for read/write access (cs.privuser). The value is encrypted.

Property	Description
cs.privuser	Specifies the database account name to use for read/write access to the database. The default value is ftuser, which is set during installation. For security reasons, be sure that your system is not using the default user name/password combination.

Debug (futuretense.ini)

The **Debug** tab holds the properties that enable various kinds of Content Server debug logging (and one property for the DebugServer servlet). The ContentServer servlet writes various error and status messages to the futuretense.txt file when the ft.debug property is set to yes.

The futuretense.txt file is located in the Content Server installation directory.

If you enable debug logging, note the following:

- Delete or archive the futuretense.txt file frequently because a large log file can affect Content Server performance.
- Because enabling any of the debug logging options can affect performance, you should not enable these options on a management or delivery system that is live.
- By default, all debug log messages go into a single log file, which can make debugging more difficult. To put debug messages into a separate log file, set the ft.dbl property to yes, which enables browser-based debugging. When this property is set to yes, Content Server creates a log file for each browser IP address. Each file is stored in the same directory as the futuretense.txt file and is created using the following naming convention: futuretense.IPaddress.txt.

  You can then use the exportlog argument of the CATALOGMANAGER tag to retrieve the log file for the IP address of the browser that you are using.

These properties are documented here in alphabetical order:

Property	Description
ft.cachedebug	Specifies whether Content Server writes information about the resultset cache to the log files. Possible values: yes | no
ft.dbdebug	Specifies whether Content Server writes information about database activity to the log files. Possible values: yes | no (default)

Property	Description
ft.dbl	Specifies whether you can retrieve and examine the log files in a browser. Possible values: yes | no If set to no (default), all debug messages are written to the futuretense.txt log file, which you can examine in any text editor. If set to yes, Content Server creates a separate log file based on the IP address of the browser that you are using.
ft.debug	Specifies whether debug messages are written to the ContentServer log file, futuretense.txt. Possible values: yes | no (default) **Note:** Enabling debug logging can affect system performance.
ft.debugport	Specifies the port that DebugServer uses to communicate with the template debugger utility. The port number must be greater than 1024. Default value: 1025
ft.evaldebug	**Deprecated**. Specifies whether Content Server writes diagnostic messages about the EvalServer servlet to the log files. Possible values: yes | no (default) Leave this property set to no.
ft.eventdebug	Specifies whether Content Server writes information about event management processing to the log files. Possible values: yes | no
ft.jspdebug	Specifies whether Content Server writes information about serving JSP pages to the log file. Possible values: yes | no
ft.logsize	Specifies the maximum size of the output log file in bytes When it reaches this size, Content Server empties the file and starts over. Default value: 1000000 For example, 50000 indicates that the maximum size of the log file is 50 KB. If you enable a large number of debugging options, set this value to a high number (such as 2,000,000) to prevent losing useful entries before you have a chance to examine them.

Property	Description
ft.pgcachedebug	Specifies whether Content Server writes information about page cache management to the log files. Possible values: yes | no
ft.ssdebug	Specifies whether Content Server writes information about sessions to the log file. Possible values: yes | no (default)
ft.syncdebug	Specifies whether Content Server writes information about data cache synchronization processing to the log file. Possible values: yes | no
ft.timedebug	Specifies whether Content Server writess information about evaluation timing to the log file. Possible values: yes | no (default)
ft.xmldebug	Specifies whether Content Server writes messages about XML template evaluation to the log file. Possible values: yes | no (default) If set to yes, comments that are written in elements are also written to the rendered HTML file displayed in the browser. If set to no, comments are stripped out of the element when it is rendered and they do not appear in the file displayed in the browser.

Email

The **Email** tab holds the properties that configure the Content Server e-mail system features. Note that there is an additional property on the **Preference** tab in futuretense_xcel.ini that enables the CS-Direct workflow e-mail system that sends notices to workflow participants when they are assigned assets through a workflow process.

These properties are documented here in alphabetical order:

Property	Description
cs.emailaccount	Specifies the user account name to be used for sending outgoing mail. This is the account name on the SMTP server. If SMTP authentication is required, you must set a value for this property.

Property	Description
cs.emailauthenticator	Specifies the class that is used as the authenticator for mail operations. Default value: com.openmarket.framework.mail.ICSAuthenticator
cs.emailcharset	Specifies the default character set that is used for the text in the subject of an e-mail message. Examples: • Latin1: text/html; charset=ISO-8859-1 • Japanese (Shift_JIS): text/html; charset=Shift_JIS • UTF-8: text/html; charset=UTF-8 If this is blank, there is no default value.
cs.emailcontenttype	Specifies the default character set that is used for the text in the body of an e-mail message. Examples: • Latin1: text/html; charset=ISO-8859-1 • Japanese (Shift_JIS): text/html; charset=Shift_JIS • UTF-8: text/html; charset=UTF-8 If this is blank, it reverts to the default text/plain
cs.emailhost	Defines the SMTP (e-mail server) host that is used by the ContentServer servlet to create and deliver e-mail messages. A valid value is required to send or receive mail.
cs.emailpassword	Specifies the password for the e-mail account used by Content Server (specified by cs.emailaccount). A valid value is required to receive mail.
cs.emailreturnto	Specifies the e-mail address from which mail is sent. That is, the e-mail address that appears in the From field of an e-mail message. Use one of the following formats: user@domain.com Full Name <user@domain.com> A valid value is required to send mail.

Export/Mirror

The **Export/Mirror** tab holds the properties that configure the Content Server Export and Mirror APIs that are used by the CS-Direct publishing system. These properties work in conjunction with the properties located on the **Publishing** tab in the `futuretense_xcel.ini` file, which is described in the "CS-Direct Property File" section.

When configuring the publishing operations for your CSEE systems, think of your individual systems (development, management, delivery) in the following terms:

- **Source**, which means the Content Server database that is the source for a publishing session. Because you can mirror assets and site configuration information from any CSEE system to any other CSEE, it is misleading to refer to the source as the CSEE management system in all cases.
- **Target**, which means either the Content Server database that you are mirroring to or the file server that you are exporting to.

For more information about publishing, see Chapter 8, "Managing the CSEE Publishing System."

These properties are documented here in alphabetical order:

Property	Description
`cs.mirrorhttpversion`	Specifies the HTTP protocol version to use to communicate with the Content Server target databases. Default value: 1
`cs.mirrorpassword`	Specifies the password for the mirror user on the target systems that this system publishes to. You set this value when you set up your CSEE system for publishing. For information, see "Configuring Your System for Mirror to Server Publishing" on page 222.
`cs.mirrorproxyserver`	Specifies the firewall server's IP address or name for the target system that this (source) system publishes to when the target and the source are separated by a firewall. You set this value when you set up your CSEE system for pubishing. Possible values: `your_server_name` or `your_server_ip_address` For information, see "Configuring Your System for Mirror to Server Publishing" on page 222.

Property	Description
`cs.mirrorproxyserverport`	Specifies the port number of the firewall server for the CSEE system that this system publishes to when the target system is separated from the source with a firewall. You set this value when you set up your CSEE system for pubishing. Possible value: `port_number` For information, see "Configuring Your System for Mirror to Server Publishing" on page 222.
`cs.mirrorrowsperpost`	Specifies the number of table rows that can be mirrored during each HTTP POST during a mirror operation If you are mirroring data that contains URL fields, you should set this to a low number because web servers impose a limit on the size of post packets. If you are mirroring data that contains only text, you can set this to a higher number. Default value: 6 For best performance, do not increase this value above 12. **Note**: If your database is configured for UTF-8 and holds non-ASCI content, you must set this value to 4 or lower.
`cs.mirrorthreads`	Specifies the number of threads to allocate to a mirror operation. Default value: 2 For best performance, do not increase this value above 8.
`cs.mirroruser`	Specifies the name of the mirror user on the target system that this (source) system publishes to. You set this value when you set up your system for publishing. For information, see "Configuring Your System for Mirror to Server Publishing" on page 222.
`cs.pgexportfolder`	Specifies the base export directory for the HTML files that are created when assets are published with the Export to Disk delivery type. • Windows NT example: `c:/FutureTense/export` • Solaris example: `/export/home/FutureTense/pgexport`

JSP

The **JSP** tab holds the properties that supply information that Content Server references when serving Java Server Pages files. If your CSEE system uses WebLogic, note that there are additional, WebLobic-only, JSP properties in the `jsprefresh.ini` file. See "jsprefresh.ini (WebLogic Only)" on page 334.

These properties are documented here in alphabetical order:

Property	Description
`cs.jspclear`	Configures the Content Server engine to delete any previously deployed JSP files and clear the application server's working folder (temp and class files) when the Content Server engine executes the first JSP deployed by Content Server. The working folder is defined by the application server. Possible values: `true` | `false`
`cs.jsppath`	Specifies the virtual root (zone) for executing deployed JSP pages. This property is used in conjunction with `cs.jsproot`, so the two properties must be in sync. The default value is synchronized with the WebLogic setting for `cs.jsproot` and is set at installation. Do not change the value of this property after installation.
`cs.jsprefresh`	When the Content Server engine deploys a new or changed JSP element, the application server may require special processing to complete the deployment of the JSP. In those cases, this property specifies the name of the class which completes the deployment. In other cases, it is left blank. For WebLogic 6.1, the default value is `com.divine.wl6special`. For others, the default is blank.
`cs.jspresponsewrapper`	Specifies whether the application server requires the PrintWriter when it runs a JSP element. The Content Server installation sets this to an appropriate value based on the type of application server you are using. Do **not** change the value of this property.

Property	Description
cs.jsproot	Specifies the folder where the application server expects to find JSP files. Some application servers allow this value to be modified by using property settings. WebLogic defines weblogic.httpd.initArgs for its JSPServlet object. The cs.jsproot property is used in conjunction with cs.jsppath, so the two must be in sync and both are set at installation. Do **not** change the value of this property after installation.
cs.jspwork	Specifies the directory where class files are created by the application server when executing JSP pages. This is not a required property and can be left blank, but it is normally set at installation.

Misc

The **Misc** tab holds miscellaneous properties such as the amount of idle time a connection can have before Content Server logs out of a connection and whether Content Server loads cache synchronization processing.

These properties are documented here in alphabetical order:

Property	Description
cs.charset	Specifies a variable that can be included in HTML forms as a hidden variable. The variable, which gets set by the browser, specifies the text encoding of the form data that Content Server must process. Default value: _charset_ Do **not** change the value of this property.
cs.contenttype	Specifies the default character set to use for HTTP headers (streaming text). Default value: text/html; charset=UTF-8 Specify a value that is appropriate for the online site that your CSEE system is delivering. Examples: • Latin1: text/html; charset=ISO-8859-1 • Japanese (Shift_JIS): text/html; charset=Shift_JIS

Property	Description
cs.disksize	Specifies the size limit in bytes for keeping uploaded files in memory while they are being posted. If an uploaded file is larger than the value specified, Content Server streams it to a temporary file until it is finished evaluating a page. This prevents excessive memory use and helps to prevent denial-of-service attacks. Default value: 102400
cs.documentation	Specifies the URL of the CSEE documentation. By default, this property is set to a FatWire documentation web site. If you prefer, you can download the most recent documentation kit from that web site, install it somewhere on your network, and then set this property to point to that location rather than to the FatWire documentation web site.
cs.HTTP_HOST	Specifies the HTTP host for CSEE systems on which the Web server does not reside on the application server machine (that is, it is a web connector installation), or in cases where an alternative web server is used to serve pages. Use one of the following formats: hostname:port IPaddress:port Default value: blank
cs.HTTP_PROTOCOL	Specifies the HTTP protocol for CSEE systems where the web server does not reside on the application server machine (that is, it is a web connector installation), or in cases where an alternative web server is used to serve pages. Default value: blank, which means the protocol is assumed to be http Possible values: http or https (or blank)

Property	Description
cs.urlfilerollup	Determines how changes to the files for URL columns are tracked. When this value is set to true, the previous versions of the related file are tracked in a name sequence. For example, if a file called filename.txt has been edited three times, filename.txt is the oldest, filename,1.txt is the next oldest version, and filename,2.txt is the current version. When this value is set to false, the file name toggles between filename.txt and filename,0.txt on alternate updates.. Default value: false for WebLogic 7.1 and true for WebLogic 6.1.
cs.xmlfolder	Specifies the working directory for HTML filtering. That is, if elements on your system use the XMLFILTER tag, temporary files are written to this directory. Default value: $HOME/FutureTense/xmltemp

Page Cache

The **Page Cache** tab holds the properties that configure Content Server's page cache settings. Content Server's page cache is monitored and maintained by the CacheManager. CSEE caching enables you to cache both complete web pages and their individual components (or pagelets).

To set up page caching on a CSEE system, you configure properties for the CacheManager and the CS-Satellite Satellite servlets. In addition, there are properties for configuring BlobServer, the servlet that serves blobs and caches blobs both through Content Server and the Satellite servlet.

For more information:

- Page caching – see the "Page Caching" chapter in the *CSEE Developer's Guide*. This chapter describes how Content Server's Cache Manager, the Satellite Servlets, and the BlobServer servlet interact and work together.
- CS-Satellite properties – see "Satellite Server (futuretense.ini)" on page 326 and "satellite.ini" on page 338.
- Resultset caching properties – see "ResultSet Cache" on page 323.
- BlobServer properties – see "Blobs/Eval" on page 302.

The page caching properties are documented here in alphabetical order:

Property	Description
`cs.alwaysusedisk`	Specifies the default cache setting for page entries in the `SiteCatalog` table that have no information in their `cacheinfo` column. If set to `yes`, then each page served from Content Server is cached to disk (the database), unless the value in that page entry's `cacheinfo` column specifies that it not be cached. Default value: `no`
`cs.freezeCache`	Specifies whether a cache maintenance event should regularly remove expired pages from the cache, or whether the expiration date of a page should be checked only when that page is requested. Set the value to `yes` if you do not want an event to regularly remove expired pages from the cache. Default value: `no`
`cs.IItemList`	Defines the file that is used for the ItemList interface that is used by the Cache Manager. The default value is provided for reference only: `COM.FutureTense.Export.ItemList` Do **not** change the value of this property.
`cs.nocache`	Provides you with the ability to disable all disk-based page caching (pages will still be cached in memory). Use this property to temporarily shut down page caching when you are debugging your site, but do not leave this value set to `true` on a live system. Default value: `false`
`cs.pgCacheTimeout`	Specifies the number of minutes that the CacheManager holds a page in the page cache. A value of 0 (zero) disables timeout, which means pages never expire. However, no matter what the setting for this property, CacheManager refreshes a cached page if the publishing system reports that it published any of the assets on that page. Default value: 1440

Property	Description
`cs.requiresessioncookies`	Specifies whether session ID information can be held in cookies or whether Content Server must encode session data into the links. Default value: `true` Set to `true` (the default) if Content Server expects session cookies to be enabled. This allows all pages to be cached and does not encode the session id into any links. A value of `false` enables URL rewriting, with a negative effect on page caching performance.
`cc.SystemPageCacheTimeout`	Specifies the number of minutes a cached page is held in memory (cached pages are cached both to disk and to memory). Default value: 1440 (that is, 24 hours)
`cc.SystemPageCacheCSz`	Specifies the maximum number of pages that can be cached in memory. Pages are cached both in memory and to disk (database). This property specifies the number of pages cached to memory, not to disk. Default value: 500

ResultSet Cache

The lowest level of cache support is database query, or resultset, caching. The **ResultSet Cache** tab holds properties that configure Content Server's resultset caching. For information about resultset caching and queries on your CSEE system, see the "Data Design" section in the *CSEE Developer's Guide*.

The three main resultset caching properties are `cc.cacheResults`, `cc.cacheResultsTimeout`, and `cc.CachResultsAbs`. They specify the default number of resultsets to cache in memory, the default amount of time to keep resultsets cached in memory, and how to calculate the expiration time.

The default values specified for these properties are used to determine how to cache the resultsets of any table in the Content Server database that does not have table-specific resultset caching properties for it.

You can create three resultset caching properties for each table in the Content Server database. Use the following syntax:

```
cc.<tablename>Csz=<number of resultsets>
cc.<tablename>Timeout=<number of minutes>
cc.<tablename>Abs=<true or false>
```

You can create as many table-specific resultset caching properties as are needed to implement your resultset caching strategy for each of your CSEE systems (which means that the values for these properties are different on each system).

Most of the Content Server system tables have table-specific resultset caching properties set for them by default. These properties are displayed on the **ResultSet Cache** tab.

However, when you create new table-specific resultset caching properties, they are displayed on the **User Defined** tab.

For information about page caching properties, see "Page Cache" on page 321. For information about BlobServer caching properties, see "Blobs/Eval" on page 302.

The following table describes the properties that appear on the **ResultSet Cache** tab. These properties are documented here in alphabetical order:

Property	Description
`cc.cacheResults`	Specifies the default number of resultsets to cache in memory. Note that this does not mean the number of records in a resultset, but the number of resultsets. Setting this value to -1 disables resultset caching for all tables that do not have their own caching properties configured. **Caution**: Do **not** set this value to -1. If you do, the Content Server interface will fail to save assets properly. Possible values: *n* (number of resultsets) Default value: 500 To set a different value for a specific table, create a property for that table using the following format: `cc.<tablename>Csz=<number of resultsets>`
`cc.cacheResultsAbs`	Determines how to calculate the expiration time for the resultsets in the resultset cache. If the value is set to `true`, the expiration time for a resultset is absolute. For example, if `cc.cacheResultsTimeout` is set to 5 minutes, then 5 minutes after the resultset was cached, it is flushed form the cache. If the value is set to `false`, the expiration time for a resultset is based on its idle time. For example, if `cc.cacheResultsTimeout` is set to 5 minutes, the resultset is flushed from the cache 5 minutes after the last time it was requested rather than 5 minutes since it was originally cached. To set this value for a specific table, create a property for that table using the following format: `cc.<tablename>Abs=<true or false>`

Property	Description
`cc.cacheResultsTimeout`	Specifies the number of minutes to keep a resultset cached in memory. Setting this value to -1 means that there is no timeout value for tables that do not have their own caching properties configured. Possible values: *n* (in minutes), or -1 to disable for tables that use this default setting. Default value: 5 To set this value for a specific table, create a property for that table using the following format: `cc.<tablename>Timeout=<number of minutes>`
`cc.ElementCatalogCSz`	Specifies the number of resultsets to cache against the `ElementCatalog` table. For best performance this value should be set to the number of rows in the table. Default value: 1000
`cc.ElementCatalogTimeout`	Specifies the number of minutes to keep idle resultsets for the `ElementCatalog` table in the resultset cache. Use -1 to disable timeout. Default value: 60
`cc.SiteCatalogCSz`	Specifies the number of resultsets to cache against the `SiteCatalog` table. For best performance, this value should be set to the number of rows in the table. Default value: 1000
`cc.SiteCatalogTimeout`	Specifies the number of minutes to keep idle resultsets for the `SiteCatalog` table in the resultset cache. Use -1 to disable timeout. Default value: 60
`cc.SystemACLCSz`	Specifies the number of resultsets to cache against the `SystemACL` table. For best performance, this value should be proportional to the number of rows in the table. Default value: 25
`cc.SystemACLTimeout`	Specifies the number of minutes to keep idle resultsets for the `SystemACL` table in the resultset cache. Default value: -1, which disables timeout for this table

Property	Description
cc.SystemInfoCSz	Specifies the number of resultsets to cache against the SystemInfo table. For best performance, this value should be set to the number of rows in the table. Default value: 500
cc.SystemInfoTimeout	Specifies the number of minutes to keep idle resultsets for the SystemInfo table in the resultset cache. Default value: -1, which disables timeout for this table.
cc.SystemUsersCSz	Specifies the number of resultsets to cache against the SystemUsers table. For best performance, this value should be proportional to the number of rows in the table. Default value: 100
cc.SystemUsersTimeout	Specifies the number of minutes to keep idle resultsets for the SystemUsers table in the resultset cache. Default value: -1, which disables timout for this table.
ft.filecheck	Specifies whether Content Server verifies the timestamp on data held in an upload field each time an item (like an element) with uploaded data is requested. On a management or delivery system, the same items are requested repeatedly and setting this value set to yes can slow the performance of the system. Set this property to no on management or delivery systems. You can set it to yes on a development system. Default value: no

Satellite Server (futuretense.ini)

The **Satellite Server** tab holds properties that describe how to communicate with any of the Satellite servlets (CS-Satellite), whether they are running locally or on remote servers.

Additionally, because CS-Satellite is installed by default on the server that hosts your Content Server application, each CSEE system also has a satellite.ini file that configures the local Satellite servlet. (Note that when CS-Satellite is also running on a remote server, that server also has a satellite.ini file.) For information about the properties in that file, see "satellite.ini" on page 338.

The following table describes the properties on the server that hosts Content Server that support communications with all Satellite servlets.

The value for each property is a comma-separated list. The ordinal position of an item in the list is what associates the host, user name, and password for each Satellite servlet. For example, the **third host** named in `cs.satellitehosts` is accessed using the **third user account** named in `cs.satelliteusers`, giving the **third password** listed in `cs.satellitepassword`.

For more information about setting these properties, see the book *Installing CS-Satellite*.

These properties are documented here in alphabetical order:

Property	Description
`cs.satellitehosts`	Specifies the host names of the servers that are hosting Satellite servlets that the CacheManager on this server (the one that hosts Content Server) needs to communicate with. Enter a comma-separated list of host names. The value for each host must include the path to the Content Server servlets. Use the following format: `http://hostname:port/servlet/` You can use https or special ports, if necessary. If required by your configuration, be sure to specify a fully-qualified domain name. The Satellite servlet that resides on this server is listed by default.
`cs.satellitepassword`	Specifies the passwords for the user accounts specified by the `cs.satelliteusrs` property. Note that the password for the Satellite servlet on this server is listed by default. The value of this property is encrypted as a single string. Therefore, when you edit the value of this property, you must enter all the passwords for all the Satellite servlet hosts, including the comma delimiter. Enter a comma-separated list of passwords in the order that matches the order in which you enter the corresponding users for the `cs.satelliteusers` property. Be sure that the order of this list also matches the order of the list of host names provided for the `cs.satellitehosts` property.
`cs.satelliteusers`	Specifies the user names for the CS-Satellite hosts. Note that the user name for the Satellite servlet on this server is listed by default. Enter a comma-separated list of user names in the order that matches the list of passwords that you specified for the `cs.satellitepassword` property.

Property	Description
satellite.blob.cachecontrol.default	Specifies a default value for the cachecontrol parameter for the satellite.blob, and RENDER.SATELLITEBLOB tags and their JSP equivalents. Default value: blank Set this property to a value that is appropriate for the majority of your blobs, and then use the cachecontrol parameter with the satellite.blob and RENDER.SATELLITEBLOB tags to override this value for individual blobs. Use the following format to set a value: hours:minutes:seconds daysOfWeek/daysOfMonth/months For more information about this format, see the description of the expiration property in the section "satellite.ini" on page 338.
satellite.page.cachecontrol.default	Specifies a default value for the cachecontrol parameter for the satellite.page, and RENDER.SATELLITEPAGE tags and their JSP equivalents. Default value: blank Set this property to a value that is appropriate for the majority of your pages and pagelets, and then use the cachecontrol parameter with the satellite.page and RENDER.SATELLITEPAGE tags to override this value for individual pages and pagelets. Use the following format to set a value: hours:minutes:seconds daysOfWeek/daysOfMonth/months For more information about this format, see the description of the expiration property in the section "satellite.ini" on page 338.

Search

The **Search** tab holds properties that Content Server uses to obtain configuration information about your third-party search engine, if your CSEE system uses one. If your organization purchased one of the search engine modules, it was installed when your CSEE system was installed.

Although the **Search** tab displays all the properties for both third-party search engine modules, this section organizes them by search engine.

AltaVista

The AltaVista properties are documented here in alphabetical order:

Property	Description
`av.cjkquery`	Specifies whether AltaVista must handle queries that use Chinese, Japanese, or Korean characters. Default value: `no` Set to `yes` if your CSEE system is localized for Chinese, Japanese, or Korean.
`av.defaultindex`	Specifies the AltaVista search index to open if none is specified as an XML parameter.
`av.dictionarypath`	Specifies the full path to the `dictionary` file located in the `doc_converters` subdirectory.
`av.license`	Specifies an explicit license value. This property exists only if you have obtained your own license for the AltaVista search engine. It replaces `av.oemkeytype`.
`av.oemkeytype`	Specifies one of the two embedded keys: For a management system, set this value to `management` to allow up to 5 million entries. You are not authorized to use this value on a delivery site where there is public access. For a delivery system, set this value to `delivery` to allow up to 250,000 entries. If you need more entries, you must obtain a specific license from AltaVista and use the `av.license` property instead of this one.
`cs.searchengine`	Specifies the search engine. Set this property to `AV`.

Verity

The Verity properties are documented here in alphabetical order:

Property	Description
`cs.searchengine`	Specifies the search engine. Set this property to `Verity`.
`verity.charset`	Specifies the name of the locale to use for all internal Verity engine operations. This property corresponds to a subdirectory of the common directory where the locale is defined. The property is optional.
`verity.debug`	Controls whether the Verity Search Engine should put debug messages into the log file. Possible values: `yes` | `no`

Property	Description
`verity.defaultindex`	The Verity search index to open if none is specified as a parameter for an XML or JSP tag or Java method.
`verity.defaultparser`	Specifies the default parser. Possible values: • `Simple` • `FreeText` • `BoolPlus`
`verity.knowledgebasepath`	The path to the Verity topic that is used for queries. This property is optional.
`verity.organization`	The organization specified in your Verity Information Server License Key. This property is case-sensitive and space-sensitive.
`verity.path`	The path to the Verity Information Server directory.
`verity.signature`	The signature text as specified in your Verity License Key. This property is case- and space-sensitive.

`verity.templatepath`	The path of the Verity vdktemplate directory. This is a subdirectory of the FutureTense directory. For example, in Windows NT:
	`C:\FutureTense\vdktemplate`
	and in Solaris:
	`/export/home/FutureTense/vdktemplate`

User Defined (futuretense.ini)

The **User Defined** tab displays custom properties that are not created by the core Content Server product but that it and the CSEE content applications use. This tab displays the following kinds of properties:

- There are two custom properties for the sample site if the Content Server portal sample site is installed.
- For each of the CSEE content applications that are installed, a custom property that specifies the version number of that application.
- Properties that specify the name of the primary key column for a content table (catalog). For information about content tables, see the "Data Design" section in the *CSEE Developer's Guide*. The installation program for CS-Direct creates custom properties of this type.
- Custom properties that specify resultset caching settings for individual tables in the Content Server database. The installation program for CS-Direct creates custom properties of this type.

The following table lists all of the properties that any of the CSEE products create in the `futuretense.ini` file, which means that they appear on the **User Defined** tab. Note that your system may have additional properties that are not included in this list.

These properties are documented here in alphabetical order:

Property	Description
`cc.AssetTypeCSz`	The number of resultsets to cache against the `AssetType` table, an object table that is created by the CS-Direct installation. Default value: 50
`cc.CategoryCSz`	The number of resultsets to cache against the Category table, an object table that the CS-Direct installation creates for asset types. Default value: 50
`cc.CheckOutInfoKey`	The primary key of the `CheckOutInfo` table, a content table that the CS-Direct installation creates for its implementation of revision tracking. Value: `checkout` Do not change the value of this property.
`cc.ComparatorsKey`	The primary key of the `Comparator` table, a content table that the the CS-Direct installation to hold field comparator classes. Value: `name` Do **not** change the value of this property.
`cc.MimeTypeKey`	The primary key of the `MimeType` table, a content table that the CS-Direct installation creates to store the mimetypes of the documents that it handles. Value: `mimetype` Do **not** change the value of this property.
`cc.SourceKey`	The primary key of the `Source` table, a content table that the CS-Direct installation creates for asset types. Value: `source` Do not change the value of this property.
`cc.StatusCodeCSz`	The number of resultsets to cache against the `StatusCode` table, a content table that the CS-Direct installation creates for asset types. Default value: 10
`cc.StatusCodeKey`	The primary key of the `StatusCode` table. Value: `statuscode` Do not change the value of this property.

Property	Description
`analysisconnector.version`	The version number of Analysis Connector, if it is installed. Do **not** change the value of this property.
`catalogcentre.version`	The version number of CS-Direct Advantage, if it is installed. CS-Direct Advantage was named Catalog Centre in previous versions of the product. Do **not** change the value of this property.
`commerceconnector.version`	The version number of the Commerce Connector utility, if it is installed. Do **not** change the value of this property.
`contentcentre.version`	The version number of CS-Direct if it is installed. CS-Direct was named Content Centre in previous versions of the product. Do **not** change the value of this property.
`image.time`	A property that holds a caching value that is passed to `satellite.blob` and `satellite.link` tags in the Content Server portal sample. The default is `5:0:0 */*/*`, which means that the images and stylesheet in the sample portal expire every day at 5:00 a.m.
`marketingstudio.version`	The version number of CS-Engage, if it is installed. CS-Engage was named Marketing Studio in previous versions of the product. Do **not** change the value of this property.
`page.time`	A property that holds a caching value that is passed to `satellite.page` tags in the Content Server portal sample. The default is: `*:0,5,10,15,20,25,30,35,40,45,50,55:0 */*/*`, which means that the portal sample pages expire every five minutes.
`soap.binaryRowsType`	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
`soap.iList`	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.

Property	Description
soap.likeconstraint	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.listrowstype	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.nestedconstraint	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.rangeconstraint	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.richtextconstraint	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.searchstate	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.standardconstraint	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.stringrowstype	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.urlrowstype	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.
soap.URLType	A CS-Direct system property used to instantiate server-side objects in response to SOAP requests made by web services. Do **not** change the value of this property.

jsprefresh.ini (WebLogic Only)

The `jsprefresh.ini` file holds properties that provide additional information that Content Server needs in order to serve JSP files correctly when your application server is WebLogic. See also the properties in the section "JSP" on page 318.

These properties are documented here in alphabetical order:

Property	Description
`app`	Specifies the name of the application that was created in WebLogic to refer to Content Server. This value is set during installation. It should match the value set for the `component` property.
`component`	Specifies the name of the application that was created in WebLogic to refer to Content Server. This value is set during installation. It should match the value set for the `app` property.
`domain`	Specifies the domain in which WebLogic is configured. This value is set during installation.
`password`	Specifies the password for the user account for the WebLogic Admin Server. This value is encrypted. This value is set during installation.
`url`	Specifies the URL at which you access WebLogic. This value is set during the installation of a CSEE system and it follows this format: `http://servername:port`
`user`	Specifies the user name of the user account for the WebLogic Admin Server. This value is set during installation.
`version`	Specifies the version number of the WebLogic server. If you are using WebLogic 6.1, set it to `6`. If you are using WebLogic 7.x, set it to `7`.

ldap.ini

If your CSEE system is using the LDAP user manager module, the `ldap.ini` file is present on the system. This file is created by the person who installs your CSEE system and it holds properites that identify the LDAP directory server.

There are additional user manager/directory services properties in two other property files. See also the following sections in this chapter:

- "dir.ini" on page 287, which describes the properties in the `dir.ini` file
- "Authentication" on page 297, which describes the properties on the `authentication` tab of `futuretense.ini`

Authentication (ldap.ini)

The **Authentication** tab holds the main configuration properties for identifying the LDAP directory server.

Property	Description
`LDAP.GroupsBase`	Specifies the base distinguished name under which the user groups that map to Content Server ACLs must be located if they are to be found. For example: `ou=groups,dc=FatWire,dc=com`
`LDAP.GUID`	Specifies the value of the LDAP user manager plug-in's AppLogic GUID. Do **not** change the value of this property. This property is provided as a convenience to the installers. They can copy this value and then paste it into the `futuretense.ini` file for the `cs.manageUser` property rather than having to type it manually.
`LDAP.Host`	Specifies the host name of the LDAP server used for user authentication.
`LDAP.Port`	Specifies the port number of the LDAP server used for user authentication.
`LDAP.PrivPassword`	Specifies the password of the user specified by the `LDAP.PrivUser` property. If there is no value set for the `LDAP.PrivUser` property, do not specify a value for this property.

Property	Description
LDAP.PrivUser	If the LDAP directory server does not allow anonymous access for attribute queries, specify the distinguished name of the user account that Content Server should use. This user must have access to search the base groups distinguished name for members. The value for this property must be the fully qualified distinguished name of a user with the required access rights. If the LDAP directory server allows anoymous access, leave this value blank.
LDAP.UserBase	Specifies the base distinguished name under which the Content Server user names must be located if they are to be found. For example: ou=people,dc=FatWire,dc=com **Note**: Set this to the lowest level in the directory hierarchy that you can.

Schema (ldap.ini)

The **Schema** tab holds attribute-mapping properties. You use these properties to specify how a user attribute that CSEE uses is identified in the directory server.

Property	Description
cn	Specifies the name of the group attribute that designates the group name. Possible values: iPlanet: cn Active Directory: cn
uniquemember	Specifies the attribute that designates a group assignment. Possible values: iPlanet: uniquemember Active Directory: member
username	Specifies the name of the user attribute that holds the user name (that is, the login ID). Possible values: iPlanet: uid Active Directory: sAMAAccountName

logging.ini

The `logging.ini` file holds properties that configure the Logging module. This module can handle logging messages of other modules, but, at present, only the Directory Services API uses the Logging module.

The Logging module writes messages to the `futuretense.txt` file.

In addition to a **User Defined** tab (which holds no properties by default), this file has two other tabs:

- Global Data (logging.ini)
- Message Resources

Global Data (logging.ini)

The **Global Data** tab holds one property:

Property	Description
`log.filterLevel`	The severity threshold that determines the amount of messages that the Logging module writes to the log. Possible values: • `info`: writes all informational, warning, error, severe, and fatal messages. • `warning`: excludes informational messages; writes warning, error, severe, and fatal messages. • `error`: excludes warning and informational messages; writes error, severe, and fatal messages. • `severe`: excludes error, warning, and informational messages; writessevere and fatal messages. • `fatal`: writes fatal messages only.

Message Resources

The **Message Resources** tab holds properties that provide logical mappings for the message bundles that various components of the applications use the Logging module to locate, deliver, and report.

Do **not** change any of the values on this tab.

Property	Description
`log.Directory.messages`	The Java resource bundle to use for the Directory Services API. The default is `com.openmarket.directory.DirectoryResources`. Do **not** change the value of this property.

Property	Description
`log.Logger.messages`	The Java resource bundle to use for the Logging module. The default is `com.openmarket.directory.LoggerResources`. Do **not** change the value of this property
`log.`*`module`*`.messages`	The Java resource bundle to use for other modules. For example, the WebMethods middleware uses a property named `log.wentconnector.messages`.

satellite.ini

The `futuretense.ini` file on each CSEE system has properties that inform that CSEE system how to connect to the servers hosting CS-Satellite. Each server that hosts a CS-Satellite application has a property file that configures the Satellite servlet that it controls. This file is named `satellite.ini`.

The `satellite.ini` file is listed in this section with the Content Server property files because each CSEE system has a Satellite servlet running on it, which means that each CSEE system has a `satellite.ini` file. When the Satellite servlet is running in the same virtual machine as the ContentServer servlet, it is said to be "co-resident."

In addition to the **User Defined** tab (which has no properties by default), this property file has the following tabs:

- Caching
- Configuration
- Remote Host
- Session

Caching

The **Caching** tab holds the CS-Satellite cache settings.

The `file_size` property can significantly influence performance. To optimize performance, try to maximize the amount of memory caching. However, be sure that you do not exceed the host's memory capacity.

If you have lots of memory or a relatively small web site, FatWire recommends caching everything to memory by setting a large value. However, in calculating whether your entire web site can fit in memory, remember that expired web pages stay in memory until explicitly removed or until the cache cleaning thread removes them. Be sure to consider this fact when you set the value of the `cache_check_interval` property.

These properties are documented here in alphabetical order:

Property	Description
`cache_check_interval`	Specifies how frequently, in minutes, the cache-cleaning program (thread) runs to delete expired pages and blobs. Default value: 3600
`cache_debug`	Specifies whether status messages about the cache are written to the Java console (a UNIX shell). Default value: `false`
`cache_folder`	Specifies the directory for that pages are cached to. Default value: `$SatelliteServerRoot/omkt/satellite/cache` For best performance, specify a directory that is on the same drive as the `\omkt` directory.
`cache_folder_persist`	For Satellite servlets that are hosted on separate servers, indicates whether the cache is cleared when the server is restarted or whether the cache that was present when the server was shut down should be reloaded when the server is reloaded again. Default value: `false` This feature is not yet fully developed. Therefore, do **not** change the default value.
`cache_max`	Specifies the maximum number of objects (pagelets and blobs) that can be cached at a time to both memory and disk cache. Default value: 500

Property	Description
`expiration`	The default expiration time from for pages, pagelets, and blobs when a cache expiration value is not specifically set for that item by the tag that generated the item or through the `satellite.page.cachecontrol.default` or `satellite.blob.cachecontrol.default` properties in the `futuretense.ini` file. Note the the value of this property is always used to determine the expiration time for a page that is generated by a URL rather than a tag. Default value: `5:0:0 */*/*` This means that everything in the CS-Satellite cache expires every day at 5:00 a.m. The format is as follows: `hours:minutes:seconds daysOfWeek/ daysOfMonth/months` Possible values: • hours: 0 through 23, where 0 is midnight • minutes: 0 through 59 • seconds: 0 through 59 • days of week: 0 through 6, where 0 is Sunday • days of month: 1 through 31 • months: 1 through 12 Other possible values: • `never`, which means the page can expire only if the cache is full and it is the least recently used page • `immediate`, which means to never cache the page Because at least the first web page that visitors use to enter your online site is retrieved from a URL, FatWire recommends the following cache timeout strategy: • Set the value of this property to a value appropriate for that first, entry page. • Be sure to set a default timeout value for the rest of your pages and blobs with the `satellite.page.cachecontrol.default` and `satellite.blob.cachecontrol.default` properties in the `futuretense.ini` file. • Use the `cachecontrol` parameter for the `satellite.page`, `satellite.blob`, `RENDER.SATELLITEPAGE`, or `RENDER.SATELLITEBLOB` tags to override the default timeout values for individual pages and blobs as appropriate.

Property	Description
file_size	Specifies the caching size in kbytes that determines the type of caching. CS-Satellite caches to disk any pagelet or blob larger than this size and caches to memory any pagelet or blob smaller than this size. This is a tuning parameter that will require some experimentation on your system. A value of 0 causes all files to be cached to disk.

Configuration

The **Configuration** tab holds the properties that configure the Satellite servlet.

These properties are documented here in alphabetical order:

Property	Description
blocktimeout	Specifies the number of seconds a request will wait when another thread is in the process of requesting the same data from the host. The default is 45. A value of -1 means wait until the previous thread returns. A value of 0 means never wait. This value must be tuned based on the host performance, average request size, and network latency. It is safe to use a large number or -1.
control_url	System property. Do **not** change the value of this property.
control_refresh_interval	System property. Do **not** change the value of this property.
password	The password for the user name specified by the username property. This value is encrypted. Default value: ftuser Be sure to change the username and password from the defaults.
readtimeout	Specifies the number of milliseconds that the Satellite servlet waits for Content Server to fulfill a request. A value of 0 leaves the timeout to the Java runtime environment. A value of 3000 sets a 3 second wait time.

Property	Description
servlet	The name of the Satellite servlet. It is used when links are constructed that access the Satellite servlet directly. The default value is Satellite. This is the locally registered name of the com.openmarket.Satellite.SatServer class in the resin.conf file.
servlet-path	The default servlet path of the application server that is running the Satellite servlet. For most application servers, including Sun ONE, it is /servlet/ For iPlanet Application Server (iAS) it is /NASApp/cs/
username	The user name for the CS-Satellite user. Default value: ftuser Be sure to change the username and password from the defaults.

Remote Host

The **Remote Host** tab holds properties that define the communications rules between CS-Satellite and Content Server. These properties are documented here in alphabetical order:

Property	Description
appserverlink	The initial string in Content Server URLs that is replaced to create a CS-Satellite URL. For most application servers, including Sun ONE, it is /servlet/ContentServer? For iPlanet Application Server (iAS) it is /NASApp/cs/ContentServer?
bservice	The pathname portion of the Blob Server URL for retrieving blobs. For most application servers, including Sun ONE, it is /servlet/BlobServer. For iPlanet Application Server (iAS) it is /NASApp/cs/BlobServer
host	The name of the host system running Content Server. This is required and there is no default.

Property	Description
`formaction`	The middle string to be replaced when converting form action URLs from the ContentServer servlet to create a corresponding URL for the Satellite servlet. For example: `formaction="ContentServer` This value is case sensitive.
`newformaction`	The string that replaces the form action string removed as specified by `formaction`. For example: `newformaction="Satellite` This value is case sensitive.
`pattern`	Set to `pagename`. Do **not** change the value of this property.
`port`	The port number for communicating with the Content Server host. The default is 80.
`protocol`	The communication protocol between the CS-Satellite host and the Content Server host. (Generally http:// or https://). Note that setting the protocol to https:// will not, in itself, ensure secure communications. You will still need to get a certificate.
`service`	The pathname portion of the Content Server URL. For most application servers, including Sun ONE, it is `/servlet/ContentServer`. For iPlanet Application Server (iAS) it is `/NASApp/cs/ContentServer`.

Session

The **Session** tab holds properties that provide information about how the Satellite servlet should interpret a user's browser session.

Property	Description
`cookieprefix`	Specifies the name of the Content Server application server's cookie. This cookie maintains browser sessions for users. Be sure that your Content Server application server's cookie name is included in the value for this property.

Property	Description
sharesession	Specifies whether the ContentServer servlet and the Satellite servlet share the user session. When CS-Satellite is co-resident, set this property to true so that user-specific information is consistent between pages. When CS-Satellite is installed in a separate virtual machine or a separate machine, set this value to false.

xmles.ini

The xmles.ini file configures the CS-Bridge XML module that is installed as part of the core Content Server product.

In addition to the **User Defined** tab (which has no properties by default), this property file has the following tabs:

- General Properties
- Parsing
- Inbound Handlers

General Properties

Property	Description
wc.icUploadDir	Specifies the path to the base directory where all CS-Bridge XML documents entered in the InBound, OutBound and DTD catalogs are stored. Default value: c:/FutureTense.

Parsing

Property	Description
wc.enableCacheRet	Specifies whether or not the DTD should be returned from the cache if the remote system is down. Default value: true Set to false if you want the default parser behavior instead.
wc.validate	Specifies whether an incoming document must specify a grammar. When set to false, the incoming document must be well-formed XML only (no grammar is required). Default value: false

Inbound Handlers

Property	Description
`numHandlers`	Specifies the number of inbound handlers to be configured. Possible values: an integer greater than 0 This value also specifies how many `InboundHandler` properties appear on this tab (the **Inbound Handlers** tab). If you set this value to 2, there are two additional properties on the tab, one for each handler, named `InboundHander0` and `InboundHandler1`.
`InboundHandler0`, `InboundHanlder1`, and so on	The class that implements the inbound handler. Handlers are invoked in order, starting with InboundHandler0. Default value: `com.openmarket.xmles.handlers.HTTPPostHandler.`

CS-Direct Property File

CS-Direct installs one property file: `futuretense_xcel.ini`. It also inserts custom properties onto the User Defined tab in the futuretense.ini file, as described in "User Defined (futuretense.ini)" on page 330.

futuretense_xcel.ini

The futuretense_xcel.ini file is the only property file for CS-Direct. This section presents the properties in the `futuretense_xcel.ini` file grouped by the tab that represents their functional group.

The tabs are documented here in alphabetical order:

- Asset Default
- Debug (futuretense_xcel.ini)
- Element Override
- KeyView
- Preference
- Publishing
- xcelerate

Asset Default

The **Asset Default** tab holds properties that are used to define certain default details about asset types—such as cache information, default ACLs, and whether eWebEditPro is present—and properties that define alternate strings for field names when an asset is displayed by remote clients like CS-Desktop and CS-DocLink.

These properties are documented here in alphabetical order:

Property	Description
`xcelerate.Article.description`	Specifies the string to use as the label for the field that represents the `description` column for assets of type Article (a sample site asset type) when it is displayed by a remote client. Default value: `Headline`
`xcelerate.asset.createdby`	Specifies the string to use as the label for the field that represents the `createdby` column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: `Created By`
`xcelerate.asset.createddate`	Specifies the string to use as the label for the field that represents the `createddate` column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: `Created On`

Property	Description
xcelerate.asset.description	Specifies the string to use as the label for the field that represents the description column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: Description
xcelerate.asset.enddate	Specifies the string to use as the label for the field that represents the enddate column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: End Date
xcelerate.asset.name	Specifies the string to use as the label for the field that represents the name column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: Name
xcelerate.asset.Publist	Specifies the string to use as the label for the field that represents the Publist column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: Sites
xcelerate.asset.sizeofnamefield	Specifies the length of the **Name** field for basic and flex asset types. For 4.0 or later releases this value is set to 64. For installations that have been upgraded from releases earlier than 4.0, this value is usually set to 32.
xcelerate.asset.startdate	Specifies the string to use as the label for the field that represents the startdate column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: Sites
xcelerate.asset.status	Specifies the string to use as the label for the field that represents the status column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: Status
xcelerate.asset.template	Specifies the string to use as the label for the field that represents the template column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: Template

Property	Description
`xcelerate.asset.updatedby`	Specifies the string to use as the label for the field that represents the `updatedby` column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: `Modified By`
`xcelerate.asset.updateddate`	Specifies the string to use as the label for the field that represents the `updateddate` column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: `Modified On`
`xcelerate.asset.workflowid`	Specifies the string to use as the label for the field that represents the `workflowid` column for asset types created with AssetMaker or FlexFamilyMaker when the field is displayed by a remote client. Default value: `Workflow Process`
`xcelerate.body.length`	Specifies the number of characters that are stored for the **Body** field in the `Article` table in the `urlbody` column. Data entered in the **Body** field for an article asset (a sample site asset type) is written to the `urlbody` column. Because this is a URL column, that data 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 this property, is also stored in the `body` column so that you can search for text in the body of an article asset with the search feature in the Content Server interface. Default value: 1000 The maximum values: 256 for Windows NT or Windows 2000; 2000 for UNIX. If this property is missing or is not set, CS-Direct uses the value from the the `cc.maxvarcharsize` property in the `futuretense.ini` file instead.
`xcelerate.defaultacl`	Specifies an ACL that is automatically assigned to page entries in the `SiteCatalog` table when they are created. Default value: blank
`xcelerate.defaultcacheinfo`	Specifies the default value that is set for the `cacheinfo` column for page entries in the `SiteCatalog` table when they are created. Default value: `true,*`

Property	Description
`xcelerate.defaultpagecriteria`	Specifies the page critera variables that can be set by default for template and SiteEntry assets. While you can add variables to this list, do **not** delete any of the default values. Default value: `c,cid,p,rendermode` For definitions of these variables and for more information about page criteria variables in general, see the *CSEE Developer's Guide*.
`xcelerate.ewebeditpro`	If you have purchased the eWebEditPro HTML editor and your developers have designed asset types that use it, this property specifies the location of the `ewebeditpro.js` file. For more information, see "Configuring eWebEditPro" on page 187.
`xcelerate.Image.description`	Specifies the string to use as the label for the field that represents the `description` column for image assets when they are displayed in the user interface. Default value: `Caption`
`xcelerate.MaxLinks`	Specified the number of links that could be included in a linkset, an old asset type that is no longer used.

Debug (futuretense_xcel.ini)

The **Debug** tab holds properties that enable the various CS-Direct debugging utilities. These properties are listed in alphabetical order, rather than the order in which they appear on the tab:

Property	Description
`xcelerate.approvaldebug`	Specifies whether Content Server logs information about the approval subsystem. When set to `on`, information about the approval system is written to `futuretense.txt`. Default value: `off`
`am.debug`	Specifies whether AssetMaker logs debugging information about the asset types it makes. When set to `on`, information about the creation of asset types is written to the `futuretense.txt` file. Default value: `off`

Property	Description
asset.debug	Specifies whether CS-Desktop and the Export Assets to XML publishing method log debugging information when they manipulate assets. When set to on, information about the manipulation of assets is written to the futuretense.txt file. Default value: off

Element Override

The **Element Override** tab holds properties that you can use to help customize the user interface. These properties are documented here in alphabetical order:

Property	Description
xcelelem.manageuserpub	This property defines the element that CS-Direct uses to manage the roles that users fulfill for your CSEE sites. Default value: Openmarket/Xcelerate/Actions/Security/AccessUserPublication
xcelelem.publishfactors	The name of the element used to provide additional publishing control factors. May be empty. Default value: Openmarket/Xcelerate/Actions/Publish/OverrideFactor
xcelelem.setpubid	Specifies the name of the element that sets the pubid session variable when visitors first come to your site via a dynamic URL. It is run once per visitor session. Default value: OpenMarket/Xcelerate/Actions/Publish/SetPubid.
xcelem.publishoptions	This property allows customization of a portion of the common options area of the publishing form. It is used by the Action/Publish/PublishOptions element. If this is defined, it names an element to call which will lay out the publishing options section of all the publishing forms. Default value: empty

KeyView

The **KeyView** tab holds the properties that provide information to the Verity KeyView utility, a component used by the CS-Desktop and CS-DocLink applications.

These properties are documented here in alphabetical order:

Property	Description
`xcelerate.apidir`	Set at installation, this value sets the path to the shared libraries for Verity KeyView. Do **not** change the value of this property.
`xcelerate.imgdir`	Set at installation, this value sets the path to the image files that Verity KeyView creates for CS-Desktop. Do **not** change the value of this property.
`xcelerate.imgurl`	Set at installation, this value sets the prefix that is added to the SRC attributes of IMG tags that are generated for image files by Verity KeyView. Do **not** change the value of this property.
`xcelerate.transformpath`	Set at installation, this value sets the path to the directory when CS-Desktop temporarily stores files that are transformed by VerityKeyView.. Do **not** change the value of this property.

Preference

The **Preference** tab holds properties that you use to configure the search feature, the tree, and the character set used on your system.

These properties are listed in alphabetical order, rather than the order in which they appear on the tab:

Property	Description
`xcelerate.charset`	Specifies the character set for the HTTP headers on all the CS-Direct pages and incoming CS-Desktop files. Default value: `UTF-8` Do **not** change the value of this property.
`xcelerate.emailnotification`	Specifies whether the CS-Direct workflow e-mail notification feature is enabled. When set to `true`, the workflow system sends e-mail messages to users when assets are assigned to them through a workflow process. Default value: `false` See also, “Email” on page 314.

Property	Description
`xcelerate.restrictSiteTree`	Specifies whether users other than admin users can toggle the tree on in the Content Server interface when it is configured to be toggled off by default (that is, the `xcelerate.showSiteTree` property is set to `false`). Set to `true` to enable only users with the xceladmin ACL to be able to toggle the tree back on. Default value: `false`. For more information about this feature, see "Configuring Whether the Tree Is Displayed by Default" on page 81.
`xcelerate.seLimit`	Specifies the maximum number of query results that should be returned from any internally conducted search engine query. Default value: 10000
`xcelerate.showSiteTree`	Specifies whether the tree is displayed by default when any user logs in to the Content Server interface. Set to `false` if you want the tree to be toggled off by default. Default value: `true` For more information about this feature, see "Configuring Whether the Tree Is Displayed by Default" on page 81.
`xcelerate.treeMaxNodes`	Specifies the number of items that are displayed under a node in the tree in the Content Server interface. When a node has more than this number of items, CSEE prompts the user to enter search criteria to reduce the number. Default value: 100 For more information about this feature, see "Configuring the Number of Items Displayed Under Nodes on Tabs" on page 80.
`xcelerate.treeType`	Specifies the kind of tree that is used in the Content Server interface. Valid values are `OMTree` or a value that specifies a customized replacement tree. Default value: `OMTree` Do **not** change this property without first consulting FatWire Professional Services or FatWire Customer Support.

Property	Description
xcelerate.usese	Specifies whether CS-Direct should use an installed search engine. Valid values are true or false. Default value: false

Publishing

The **Publishing** tab holds the properties that provides information to the CS-Direct publishing system. The publishing system also uses the properties on the Export/Mirror tab in the futuretense.ini file. For descriptions of those properties, see "Export/Mirror" on page 316.

These properties are documented here in alphabetical order:

Property	Description
xcelerate.batchhost	Specifies the host name and port number of the web server hosting this CSEE system.
xcelerate.batchmode	Defines the batch publishing mode. Legal values: • single—batch host is a dedicated IP address. • multiple—batch host is a cluster IP address. Default value: single
xcelerate.batchpass	Specifies the password for the batch user. Default value: xceladmin Be sure to change this value after you create the batch user for this CSEE system. For information, see "Create the Batch User Account" on page 216.
xcelerate.batchuser	The CS-Direct publishing system runs as a background process and you must configure a batch user account for the publishing system to use. This property specifies the user name of the batch user. Default value: admin Be sure to change this value after you create the batch user for this CSEE system. For information, see "Create the Batch User Account" on page 216.

Property	Description
`xcelerate.blobref`	The name of the class that manages the publish references for blobs. The default is provided here for reference only: `com.openmarket.xcelerate.publish.BlobRef`. Do **not** change the value of this property. For information about published references, see "About Session History" on page 248.
`xcelerate.bulkapprovechunk`	Specifies the number of assets to approve at the same time, in the same batch or "chunk," when someone uses the **Approve Multiple Assets** feature in the Content Server interface. The feature approves all the assets that are selected for approval in batches and the number of assets in each batch is set by this property. Default value: 500 For information about the Approve Multiple Assets feature, see "Approving Multiple Assets" on page 237.
`xcelerate.donotregenerate`	Specifies whether cached pages are regenerated after a publishing session. Possible values: • blank, that is, no value—means that all the pages in the cache that were affected by the publish session are refreshed. • `unknowndeps`—means that cached pages that were generated from an element that used a `RENDER.UNKNOWNDEPS` tag are not refreshed. • `*` (asterisk)—means none of the pages in the cache are refreshed. In other words, the affected pages are refreshed only when a visitor requests the page. Default value: blank (no value) Do not change the value of this property without first consulting FatWire support personnel.
`xcelerate.mirrorini`	If you have had the element identified by the `xcelerate.remotecall` property modified in such a way that it needs information from additional property files other than `futuretense.ini`, this property specifies the names of all the property files that are needed. Default value: `futuretense.ini` Do **not** modify this value to include additional property files without consulting FatWire Professional Services or FatWire Customer Support.

Property	Description
`xcelerate.mirroruser`	No longer used. May exist on some systems for backwards compatibility.
`xcelerate.pageref`	Specifies the name of the class that manages publish references for pages. The default is provided here for reference only: `com.openmarket.xcelerate.publish.PageRef`. Do **not** change the value of this property.
`xcelerate.protectionleveldur ingpublish`	Specifies whether an asset that is approved for a specific destination can be edited while a publishing session is in process for that destination. Possible values are 1 and 2. `1` means that you can edit an approved asset as long as it has been previously published to that destination. You cannot edit the asset if any of the following conditions exist: • The asset is being published as you try to edit it. • The asset has been approved for the destination but has never been published to that destination. • The asset is approved for the destination, the asset has been published previously, but it has been edited and reapproved since it was published. `2` means that an asset that is approved for the destination cannot be edited while the publishing session is in progress for that destination. Default value: `1` The default value of `1` is appropriate when your system uses the Mirror to Server delivery type. If your system uses the Export to Disk delivery type and your pages have embedded assets, you should set this value to `2` if the users edit those assets separately from the pages they are embedded in.
`xcelerate.pubcleanupelt`	Specifies the name of the element that the publishing system uses during the cleanup phase of a mirror publish operation. Default value: `PubCleanupElement` Do **not** modify this value without consulting FatWire Professional Services or FatWire Customer Support.

Property	Description
xcelerate.pubkeydir	Specifies the directory where the publishing system writes information about the items that have been published to the various target systems. This value is set by the CS-Direct installation. Default value: c:/FutureTense/pubkeys/
xcelerate.publishinvalidate	Specifies whether an asset is marked as changed on the destination system when an asset is published. If it is marked as changed, it must be approved on that system before it can be published from that system to a new destination. Possible values: true | false Default value: true Because having the publishing system take the time to mark the assets as changed on the destination adds time to the publishing session, typically you leave this property set to true on development and management systems but change it to false on delivery systems.
xcelerate.pubsetupelt	Specifies the name of the element that the publishing system uses during the setup phase of a mirror publish. Default value: PubSetupElement Do **not** modify this value without consulting fatwire Professional Services or fatwire Customer Support.
xcelerate.remotecall	Specifies the pagename that is invoked on the target system during a mirror publishing session. Default value: Openmarket/Xcelerate/PrologActions/Publish/Mirror1/RemoteCall Do **not** modify this element or change the value of this property without assistance from FatWire Professional Services or FatWire Customer Support.
xcelerate.templatedefault	The name of the template to use if a template cannot be found to render an asset type. The template can be specific to the site and asset type, specific just to the site for all asset types, or shared among all sites and asset types. Default value: Openmarket/TemplateDefault

xcelerate

The **xclerate** tab holds the properties that specify such things as default administrative settings, whether the InSite Editor is enabled, whether workflow configuration and search engine are being used, whether LDAP is being used, and so on.

These properties are documented here in alphabetical order:

Property	Description
`xcelerate.adminacl`	Specifies the ACL that users must be assigned so they can access administrator functions (that is, any of the functions that appear on the **Admin** tab in the Content Server interface). Default value: `xceladmin` If you change the value of this property to a different ACL, be sure to assign that ACL to all the tables that currently have the xceladmin ACL assigned to them.
`xcelerate.adminrole`	Specifies an ACL that is set for all administrative tables during installation. Default value: `xceladmin`. Do **not** change the value of this property after your system is installed.

`xcelerate.base`	Specifies the top-level (base) directory of the CS-Direct elements. During installation, the installer might need to edit this value to indicate where the CS-Direct elements are in your installation.
	Default values:
	• Windows NT or Windows 2000:
	`c:/FutureTense/elements/OpenMarket/Xcelerate`
	• Solaris or AIX:
	`/export/home/FutureTense/elements/OpenMarket/Xcelerate`
	Do **not** change the value of this property after your system is installed.

`xcelerate.crosssiteassign`	Specifies whether users from more than one site can participate in the same workflow process. Possible values: `true` | `false` Default value: `false`
`xcelerate.domain`	Specifies the domain name of the system, not including the server (machine) name. This property is used by applications that have been integrated with Content Server and that have a browser interface.

Property	Description
`xcelerate.editrole`	Specifies an ACL that is set for editorial tables during installation. Default value: `xceleditor`. Do **not** change the value of this property after your system is installed.
`xcelerate.emailattr`	Specifies the name of the user attribute that is used to identify a user's e-mail address to your CSEE system. These attributes are kept in the `SystemUserAttr` table. Default value: `mail`.
`xcelerate.enableinsite`	Enables or disables the InSite Editor for this CSEE system. A value of `true` enables the InSite Editor. Default value: `false.` **Note**: do not enable the InSite Editor on your CSEE delivery system.
`xcelerate.localeattr`	Specifies the name of the user attribute that identifies the locale that a user specifies if you have more than one language pack installed on your CSEE system. Default values: • When there is one language present, the value is blank. • When there is more than one language present, the the default is `locale`.
`xcelerate.lockdir`	Specifies the directory path, including the final slash, to the directory where CS-Direct stores information about the locks that lock data during database operations. If this CSEE system is installed on a cluster, this directory must have write permissions for and be accessible to all cluster members. Default value: `c:/FutureTense/locks/`.

Property	Description

`xcelerate.previewhost`	One of two properties that enable the preview host feature, this property sets the cgi path to use for the preview host.
	For information about this feature, see "Maintaining Separate Browser Sessions for Preview" on page 187.
	If you provide a value for this property, use the following syntax:
	• For most application servers, including Sun ONE:
	`http://servername:port/servlet/`
	• For iPlanet Application Server (iAS):
	`http://servername:port/NASApps/cs/`

`xcelerate.previewservlet`	One of two properties that enable the preview host feature, this property specifies which servlet the the preview host should use. For information about this feature, see "Maintaining Separate Browser Sessions for Preview" on page 187. Default value: `ContentServer` Possible values: `ContentServer` or `Satellite`
`xcelerate.pubrolesattr`	This is an old property that remains for backward compatibility. It implemented a rudimentary form of LDAP attribute-mapping for sites and roles that we no longer recommend. By default, this value is blank, which means that information about a user's roles is stored in the `UserPublication` table. If you want to implement LDAP attribute-mapping, see the descriptions for `xcelerate.sitenameattr`, `xcelerate.siteroot`, and `xcelerate.usermanagerclass`.
`xcelerate.rolemanagerclass`	Specifies the name of the role manager class. By default, the value of this property is set to the CS-Direct role management system. Do not change the value of this property.
`xcelerate.saveSearchDir`	Specifies the defdir (default storage directory) for the `SaveSearch` table. This table has a URL column that holds the saved searches on a development or management system. Default value: `c:/FutureTense/Storage/SaveSearch.`

Property	Description
xcelerate.sePath	Specifies the directory where search indexes are stored when you are using a search engine on your CSEE system. If you change this value, be sure to specify a directory that exists. (This property does not create the directory for you.) Default values: • Windows NT or Windows 2000: c:/FutureTense/sedb • Solaris or AIX: /export/home/FutureTense/sedb
xcelerate.sitesattr	This is an old property that remains for backward compatibility. It implemented a rudimentary form of LDAP attribute-mapping for sites and roles that we no longer recommend. By default, this value is blank, which means that information about a user's roles is stored in the UserPublication table. If you want to implement LDAP attribute-mapping, see the descriptions for xcelerate.sitenameattr, xcelerate.sitesroot, and xcelerate.usermanagerclass.
xcelerate.sitenameattr	Specifies the LDAP naming attribute for the LDAP nodes that specify CS-Direct sites when you are using LDAP attribute mapping to track the sites that a user has roles for. By default, this value is blank, which means that information about a user's roles is stored in the UserPublication table. If there is a value specified for this property, the xcelerate.usermanagerclass, and xcelerate.sitesroot properties must also be configured correctly.
xcelerate.sitesroot	Specifies the root site node in LDAP when you are using LDAP attribute mapping to track the sites that a user has roles for. By default, this value is blank, which means that information about a user's roles is stored in the UserPublication table. If you specify a value, use the full distinguished name of the attribute that represents the root site node in your directory server. If there is a value specified for this property, the xcelerate.usermanagerclass and xcelerate.sitenameattr properties must also be configured correctly.

Property	Description
xcelerate.tempobjectsdir	Specifies the defdir (default storage directory) for the TempObjects table, a CS-Direct table that stores information about objects that are uploaded or in the process of being created until they are either saved or canceled. Default value: c:/FutureTense/tempobjectsdir/
xcelerate.treetabmanagerclass	The class that implements ITreeTabManager to provide tree tab descriptions for CS-Direct. The default is provided here for reference only: com.openmarket.xcelerate.treetab.TreeTabManager Do **not** change the value of this property.
xcelerate.usermanagerclass	The class that implements IUserManager to provide user services for CS-Direct. Default value: com.openmarket.xcelerate.user.UserManager If you want to implement LDAP attribute-mapping for site and role names, set this property to the following value, exactly: com.openmarket.xcelerate.user.LDAPSchemaUserManager
xcelerate.workflowdir	Specifies the name of the directory that holds files related to workflow processes. Default value: $HOME/FutureTense/workflow If you change the value from the default, be sure that directory exists.
xcelerate.workflowengineclass	The class that implements IWorkflowEngine to provide workflow services for CS-Direct. The default is provided here for reference only: com.openmarket.xcelerate.workflow.WorkflowEngine Do **not** change the value of this property.

CS-Direct Advantage Property Files

CS-Direct Advantage installs the following property files:

- assetframework.ini
- catalog.ini
- gator.ini
- transact.ini
- visitor.ini

assetframework.ini

The `assetframework.ini` file holds properties that determine where files that hold information about flex asset history and publishing are stored.

These properties are documented here in alphabetical order:

Property	Description
`afk.historydata`	Specifies the directory that holds history data. Default value: `c:/futuretense/history/`
`afk.publishdata`	Specifeis the directory that holds publish data. Default value: `c:/futuretense/publish/`

catalog.ini

The `catalog.ini` file holds properties that that CS-Direct Advantage uses to configure the shopping cart. These properties are documented here in alphabetical order:

Property	Description
`disc.couponFields`	The list of field names to set in the coupon template that communicates Cart discounts to the CommerceEngine. For each name in this list, there must be a property called `disc.`*`name`* with a value defined. Default value: `Type OfferURL DiscountTypeA NotAppliedMessage ItemDescriptionModifier` The default defines the five fields normally used for this purpose and requires the corresponding five property values. Do **not** change the value of this property unless you are instructed to do so by qualified FatWire support personnel.

Property	Description
`disc.DiscountTypeA`	Cart-level discount Coupon Template setting. Default: `total currency` See also the `disc.couponFields` property. Do **not** change the value of this property unless you are instructed to do so by qualified FatWire support personnel.
`disc.ItemDescriptionModifier`	Cart-level discount Coupon Template setting. Default: `-` See also the `disc.couponFields` property. Do **not** change the value of this property unless you are instructed to do so by qualified FatWire support personnel.
`disc.NotAppliedMessage`	Cart-level discount Coupon Template setting. Default: `-` See also the `disc.couponFields` property. Do **not** change the value of this property unless you are instructed to do so by qualified FatWire support personnel.
`disc.OfferURL`	Cart-level discount Coupon Template setting. Default: “`pagename=index`” See also the `disc.couponFields` property. Do **not** change the value of this property unless you are instructed to do so by qualified FatWire support personnel.
`disc.Type`	Cart-level discount Coupon Template setting. Default: `coupon` See also the `disc.couponFields` property. Do **not** change the value of this property unless you are instructed to do so by qualified FatWire support personnel.
`discitem.addDiscountCF`	Specifies whether the commerce engine should add fields for Full Price and Discount Description. Values can be given as `yes/no`, `true/false`, `on/off`, or `1/0`. Default: `yes`

Property	Description
discitem.CF__DiscDesc	Specifies the field name in which to insert the Discount Description of a line item when sending a Cart to the CommerceEngine. This property is relevant only if discitem.addDiscountCF is set to yes, true, on, or 1. Default: CF__DiscDesc
discitem.CF__FullPrice	Specifies the field name in which to insert the Full Price of a line item when sending a Cart to the CommerceEngine. This propverty is relevant only if discitem.addDiscountCF is set to yes, true, on, or 1. Default: CF__FullPrice
mwb.cartsetdir	Specifies the directory path (including the terminating slash character) where cartset data files are stored. This value is set by the installation. It points to the /gator/cartset directory in the installation directory Do **not** change the value of this property.

gator.ini

The gator.ini property file is the main property file for CS-Direct Advantage. This section presents the properties in gator.ini grouped by the tab that they are located on.

In addtion to the **User Defined** tab (which has no properties by default), the gator.ini file has one tab named **Gator**. These properties are documented here in alphabetical order:

Property	Description
cc.attrDisplayStyle	Specifies whether CS-Direct Advantage uses a flex attribute's name or its description when it displays that flex attribute as a field on a **New**, **Edit**, and **Inspect** form for flex assets and flex parent assets. Default value: name Possible values: name or description
cc.fullconstraint	Specifies whether the queries that are generated by the ASSETSET family of tags repeat outer constraints inside inner constraints. Possible values: true | false Default value: true

Property	Description
cc.money	Specifies the SQL string for defining a field that holds a monetary value. This value is database specific. Do **not** change this value without consulting your database administrator.
cc.querystyle	Specifies how queries are generated by the ASSETSET family of tags. Possible values: • subquery • join • intersect Default value: join Note that setting the value of this property to intersect functions only if your database can support intersection queries.
cc.string	Specifies the SQL string for defining a field that holds string values. This value is database specific. Do **not** change this value without consulting your database administrator.
cc.url	Specifies the SQL string for defining a field of URL values. This value is database specific. Do **not** change this value without consulting your database administrator.
cc.urlattrpath	Specifies the defdir (default storage directory) for flex attributes of type URL. Default value: c:/futuretense/futuretense_cs/ccurl/
mwb.assetsetclass	Specifies the name of the class that supplies the services for assetset management. The default is provided here for reference only: com.openmarket.gator.assetset.AssetSet Do **not** change the value of this property.
mwb.cartclass	Specifies the name of the class that supplies the services for cart management. The default is provided here for reference only: com.openmarket.catalog.txcart.Cart Do **not** change the value of this property.
mwb.cartsetclass	Specifies the name of the class that supplies the services for cart set management. The default is provided here for reference only: com.openmarket.com.cartset.CartSet Do **not** change the value of this property.

Property	Description
mwb.commercecontextclass	Specifies the name of the class that supplies the services for commerce context.The default is provided here for reference only: com.openmarket.gator.commercecontext.CommerceContext. Do **not** change the value of this property.
mwb.defaultattributes	Specifies the default flex attribute to use when creating assetsets. That is, if a SEARCHSTATE tag does not specify an attribute to use to sort by, CS-Direct Advantage uses the value of this property. Default value: blank
mwb.path	Specifies the directory path (including the terminating slash character) where CS-Direct Advantage is installed. Do **not** change the value of this property.
mwb.promotioncutoff	A system setting for CS-Engage. Default value: 50 Do **not** change the value of this property.
mwb.searchdir	Specifies the directory path (including the terminating slash character) where rich text indexes are stored. The default is provided here for reference only: c:/futuretense/gator/searchdir/ Do **not** change the value of this property.
mwb.searchstateclass	Specifies the name of the class that supplies the services for search state management. The default is provided here for reference only: com.openmarket.gator.searchstate.SearchState Do **not** change the value of this property.
mwb.segmentcutoff	A system setting for CS-Engage. Default value: 50 Do **not** change the value of this property.

transact.ini

The `transact.ini` property file holds properties that support an interface with either Transact or another transaction processing system. In addition to the User Defined tab (which has no properties by default), it has the following tabs:

- Commerce Engine
- Tx Data Files
- Tx Transaction Server
- Tx Store Defaults
- Overrides for TxStore0
- Overrides for TxStore1

Commerce Engine

Property	Description
`cm.class`	The name of the class supplying CommerceEngine services. For an installation without Transact, the value is `com.openmarket.commerceengine.stubengine.StubEngine`. For an installation with Transact (the default), the value is: `com.openmarket.transactengine.TransactEngine`.
`tx.debugMode`	Flag (on/off) to set TransactEngine debug level logging with messages written to the app server console. The default is `off`, which should be the value for normal operation.
`tx.httpDebugMode`	Flag (on/off) to set TransactEngine debug level logging with messages written to the app server console that include the actual XML documents written and received by the CommerceEngine in communication with Transact. The default is `off`, which should be the value for normal operation.
`tx.transportTimeout`	Seconds to wait before deciding Transact is down. The default is 300 (5 minutes).

Tx Data Files

Property	Description
tx.authServiceDTD	The file containing XML DTD for an Authentication message. The default is conf/auth-service.dtd.
tx.CEngineDirectory	File system directory prefix for other file name properties. The default is /CommerceEngine.
tx.certificates	The location of certificates for multithreaded Java SSL Stack. The default is certs.
tx.configFile	The file containing mappings between TransactEngine service IDs and the class names that implement them. The default is conf/TransactEngine.cfg.
tx.orderEntryDTD	File containing XML DTD for a Transact Order. The default is conf/order-entry.dtd.
tx.oslOfrFile	The file containing PDO<->DO data mapping. The default is conf/osl.ofr.
tx.payAccountDTD	File containing XML DTD for an Authentication message. The default is conf/pay-account.dtd.
tx.prefixProperties	The list of properties to prefix with the value given in tx.CEngineDirectory. The default is .configFile .oslOfrFile .orderEntryDTD .authServiceDTD .payAccountDTD .purchaseSummaryDTD .certificates .offerKeyFile .receiptKeyFile. Do **not** change the value of this property
tx.purchaseSummaryDTD	The file containing XML DTD for an Authentication message. The default is conf/purchase-summary.dtd.
tx.transportclass	Specifies the class name of the class that the SSL client service uses to communicate with Transact. The default is given here for refernce only: com.openmarket.com.commerceengine.ContentServerTransport Do **not** change the value of this property.

Tx Transaction Server

Property	Description
`tx.authServer`	The URL of the application providing Authentication/Authorization services. The default is `https://`*`HOSTNAME`*`/tms-subs-s/bin/authorize.cgi`.
`tx.transactHttpPort`	Port number for HTTP connections to the transaction server. The default is 80.
`tx.transactScheme`	The method (`http`/`https`) for connection to the transaction server. The default is `https`.
`tx.transactServer`	Web-accessible host name of the transaction server.
`tx.transactSSLPort`	Port number for SSL connections to the transaction server. The default is 443.
`tx.transactUrl`	Pathname portion of the transaction server URL which receives DOs and CSEE. The default is `/tms-ts/bin/order-form.cgi`.

Tx Store Defaults

Property	Description
`tx.defaultStore.accessname`	This store's Transact seller account accessname. The default is `accessname`.
`tx.defaultStore.catalogPort`	Web-accessible port of the store's catalog server. The default is 80.
`tx.defaultStore.catalogScheme`	Protocol (`http` or `https`) for contacting store's catalog server. The default is `http`.
`tx.defaultStore.catalogServer`	Web-accessible server name of the store's catalog server. The default is your current server.
`tx.defaultStore.catalogUrl`	Pathname portion of the Store's catalog server. The default is `/cgi-bin/gx.cgi/AppLogic+FTContentServer?`.
`tx.defaultStore.currencyCode`	Three-letter ISO 4017 designation for currency accepted at this store. The default is USD.

Property	Description
`tx.defaultStore.fulfillmentPort`	Web-accessible port of the store's fulfillment server. The default is 80.
`tx.defaultStore.fulfillmentScheme`	Protocol (`http` or `https`) for contacting store's fulfillment server. The default is `http`.
`tx.defaultStore.fulfillmentServer`	Web-accessible server name of the store's fulfillment server.
`tx.defaultStore.fulfillmentUrl`	Pathname portion of the Store's fulfillment server. The default is `/cgi-bin/gx.cgi/AppLogic+FTContentServer?`.
`tx.defaultStore.offerKeyFile`	The file containing Digital Offer Shared Secret Keys. The default is `secrets/flat_O.kf`.
`tx.defaultStore.password`	The password for the account given in `tx.defaultStore.accessname`. The default value is `password`. The default should be changed.
`tx.defaultStore.receiptKeyFile`	The file containing Digital Receipt Shared Secret Keys. The default is `secrets/flat_R.kf`.
`tx.defaultStore.storeID`	An integer used as Transact's identifier for the store. The default is 1.
`tx.numStores`	The number of stores to be configured. Each store configured has a number of properties named in the form `tx.store_`*`n.propname`* where *`n`* is an integer from 0 through one less than the value given here and *`propname`* is the name of a specific property. There must be at least 1 store defined. The default is the *`n`* value in the `-Dtx.numStores=`*`n`* command-line option given to the JRE to start the PropEditor. See the properties `tx.store_0.`*`propname`* and `tx.store_1.`*`propname`* for examples of properties set when this value is 2. For a value of 3, there would also be a set `tx.store_2.`*`propname`*. For 4, there would also be `tx.store_3.`*`propname`*, and so on.

Overrides for TxStore0

The **TxStore0** tab holds properties that override the default Tx Store Defaults settings for the store that is identified as Store0.

Property	Description
`tx.store_0.accessname`	This store's Transact seller account accessname. The default is `accessname`. See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_0.catalogPort`	Web-accessible port of the store's catalog server. The default is 80. See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_0.catalogScheme`	Protocol (`http` or `https`) for contacting store's catalog server. The default is `http`. See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_0.catalogServer`	Web-accessible server name of the store's catalog server. See `tx.numStore`ss for the value that sets the number of stores to be configured.
`tx.store_0.catalogUrl`	Pathname portion of the Store's catalog server. The default is `/cgi-bin/gx.cgi/AppLogic+FTContentServer?` See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_0.currencyCode`	Three-letter ISO 4017 designation for currency accepted at this store. The default is `USD`. See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_0.fulfillmentPort`	Web-accessible port of the store's fulfillment server. The default is 80. See `tx.numStore`s for the value that sets the number of stores to be configured.

Property	Description
tx.store_0.fulfillmentScheme	Protocol (http or https) for contacting store's fulfillment server. The default is http. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_0.fulfillmentServer	Web-accessible server name of the store's fulfillment server. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_0.fulfillmentUrl	Pathname portion of the Store's fulfillment server. The default is /cgi-bin/gx.cgi/AppLogic+FTContentServer? See tx.numStores for the value that sets the number of stores to be configured.
tx.store_0.offerKeyFile	The file containing Digital Offer Shared Secret Keys. The default is secrets/flat_O.kf See tx.numStores for the value that sets the number of stores to be configured.
tx.store_0.password	The password for the account given in tx.defaultStore.accessname. The default value is password. The default should be changed. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_0.receiptKeyFile	The file containing Digital Receipt Shared Secret Keys. The default is secrets/flat_R.kf. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_0.storeID	An integer used as Transact's identifier for the store. The default is 1 See tx.numStores for the value that sets the number of stores to be configured.

Overrides for TxStore1

The **TxStore1** tab holds properties that override the default Tx Store Defaults settings for the store that is identified as Store1.

Property	Description
`tx.store_1.accessname`	This store's Transact seller account accessname. The default is `accessname`. This property only exists if at least two stores are configured. See tx.`numStores` for the value that sets the number of stores to be configured.
`tx.store_1.catalogPort`	Web-accessible port of the store's catalog server. The default is 80. This property only exists if at least two stores are configured. See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_1.catalogScheme`	Protocol (`http` or `https`) for contacting store's catalog server. The default is `http`. This property only exists if at least two stores are configured. See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_1.catalogServer`	Web-accessible server name of the store's catalog server. This property only exists if at least two stores are configured. See `tx.numStore`s for the value that sets the number of stores to be configured.
`tx.store_1.catalogUrl`	Pathname portion of the Store's catalog server. The default is `/cgi-bin/gx.cgi/AppLogic+FTContentServer?`. This property only exists if at least two stores are configured. See `tx.numStore`s for the value that sets the number of stores to be configured.

Property	Description
tx.store_1.currencyCode	Three-letter ISO 4017 designation for currency accepted at this store. The default is USD. This property only exists if at least two stores are configured. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_1.fulfillmentPort	Web-accessible port of the store's fulfillment server. The default is 80. This property only exists if at least two stores are configured. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_1.fulfillmentScheme	Protocol (http or https) for contacting store's fulfillment server. The default is http. This property only exists if at least two stores are configured. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_1.fulfillmentServer	Web-accessible server name of the store's fulfillment server. This property only exists if at least two stores are configured. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_1.fulfillmentUrl	Pathname portion of the Store's fulfillment server. The default is /cgi-bin/gx.cgi/AppLogic+FTContentServer?. This property only exists if at least two stores are configured. See tx.numStores for the value that sets the number of stores to be configured.
tx.store_1.offerKeyFile	The file containing Digital Offer Shared Secret Keys. The default is secrets/flat_0.kf. This property only exists if at least two stores are configured. See tx.numStores for the value that sets the number of stores to be configured.

Property	Description
`tx.store_1.password`	The password for the account given in `tx.defaultStore.accessname`. The default value is `password`. The default should be changed. This property only exists if at least two stores are configured. See `tx.numStores` for the value that sets the number of stores to be configured.
`tx.store_1.receiptKeyFile`	The file containing Digital Receipt Shared Secret Keys. The default is `secrets/flat_R.kf`. This property only exists if at least two stores are configured. See `tx.numStores` for the value that sets the number of stores to be configured.
`tx.store_1.storeID`	An integer used as Transact's identifier for the store. The default is 1. This property only exists if at least two stores are configured. See `tx.numStores` for the value that sets the number of stores to be configured.

visitor.ini

Although it is CS-Direct Advantage that installs the `visitor.ini` property file, the properties in this file configure CS-Engage rather than CS-Direct Advantage.

These properties configure the visitor data collection and other features provided by CS-Engage. There are two tabs in this file, **User Defined**—which has no properties by default—and **Visitor Data**.

The **Visitor Data** tab holds the main propeties in the file. These properties are documented here in alphabetical order:

Property	Description
`vis.adminrole`	Specifies the ACL that CS-Engage users need in order to work with the visitor attribute, history attribute, history type, and recommendation asset types. Default value: `VisitorAdmin` Do **not** change the value of this property.

Property	Description
`vis.compileclasspath`	Specifies the classpath against which to compile the rules. This value is set during the installation and should not be changed after that point. For information about how this value should be set, see *Installing the CSEE Content Applications*.
`vis.editrole`	Specifies the ACL that two kinds of CS-Engage users need: • Content providers who use the CSEE system to create segments and promotions. • The visitors to your online site when you are using CS-Engage to gather information about them for segments. Default value: `Visitor` Do **not** change the value of this property.
`vis.genclasspath`	Specifies the directory (including the final slash character) where rules-engine-generated class files for visitor data are stored. This value is set during the installation and should not be changed after that point. For information about how this value should be set, see *Installing the CSEE Content Applications*.
`vis.money`	Specifies the SQL string for defining fields that hold monetary values. Do **not** change the value of this property without consulting your database administrator.
`vis.path`	Specifies the directory that holds the ruleset.dtd file, which is usually the installation directory This value is set during installation. Do **not** change the value of this property..
`vis.rulesetxmlpath`	Specifies the defdir (default storage directory) for the XML versions of the rule sets. This value is set during installation. Do **not** change the value of this property.
`vis.sessiondata`	Specifies the defdir (default storage directory) for storing visitor session data. This value is set during installation. Do **not** change the value of this property.

Property	Description
`vis.url`	Specifies the SQL string for defining visitor and history attributes of type URL. Default value: `VARCHAR(128)` Do **not** change the value of this property without consulting your database administrator.
`vis.urlpath`	Specifies the defdir for binary visitor and history attributes. Default value: `/futuretense/visurl/`

CS-Engage Property File

The configuration properties for CS-Engage are located in the CS-Direct Advantage property file named `visitor.ini`.

CS-Engage installs one property file that holds only one property. The file is named `ms.ini` and the property is named `ms.enable`. The `ms.enable` property is set to `true` when CS-Engage is installed and enabled.

CS-Satellite Property Files

When you install CS-Satellite as a stand-alone application on a separate server, there are two property files that are present on that server:

- `satellite.ini`
- `resin.conf`

For information about the properties in `satellite.ini`, see “satellite.ini” on page 338.

For information about the properties in `resin.conf`, refer to your Resin documentation.

Chapter 12

System Information: Content Server Database

This chapter 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 your Content Server database appropriately.

This chapter contains the following sections:

- Cache Management Tables (Content Server)
- Approval System Tables (CS-Direct)
- Publishing System Tables (CS-Direct)
- Workflow Tables (CS-Direct)
- Flex Family Tables (CS-Direct Advantage)
- Visitor Tables (CS-Engage)

This chapter also contains information about how to purge inactive data from the visitor tables.

For information about all the tables in the Content Server database, see the *CSEE Database Schema*.

Cache Management Tables (Content Server)

Content Server delivers the CacheManager, a page caching utility that manages both the Content Server page cache and the CS-Satellite 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 CSEE 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 (CS-Direct)

The CS-Direct 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 (CS-Direct)

The CS-Direct 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 CSEE 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 (CS-Direct)

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 `WorkflowAssignment` 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
`WorkflowAssignment`	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.
`WorkflowObject`	One row for each asset in workflow. When an asset leaves workflow, the row is deleted.

Basic Asset Tables (CS-Direct)

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 Family Tables (CS-Direct Advantage)

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 (CS-Engage)

CS-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. CS-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. CS-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 CS-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 CS-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 *CSEE Developer's Tag Reference*, and the "Coding Basics" section of the *CSEE 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 *CSEE 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 CS-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 *CSEE Developer's Tag Reference*.

Deleting Unnecessary .class Files

CS-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 CS-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, CS-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 CS-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.

CS-Engage regenerates any `.class` files that it needs when it needs them.

Index

A

ACL (access control list)
  adding 32
  and roles 28, 45
  assigned to system and sample site users 37
  assigning to page 35
  assigning to tables 34
  bit mask numbers 28
  creating 32
  database tables 28
  default system ACLs 29
  deleting 34
  described 27
  editing 33
  for batch user 216
  for mirror user 222, 250
  list 33
  page entries 28
  permissions 28
  required for users 39
  restriction message 36
  role in user management 24
  SystemACL table 29
  user accounts 27
actions
  creating 119
  deadlock 87, 100
  delegate 88, 98
  deleting 121
  editing 121
  group deadlock 88, 101
  step 85, 98
  timed 84, 95, 97
Active List tab 66
admin
  user name 37
Admin tab 66
  enable when migrate a site 235
  xcleadmin ACL 32
administrators
  ACLs they need 40
  property that sets admin ACL 357
  roles they need 45
aliases
  table that stores aliases for visitors 383
all-voting step 86
AltaVista search engine
  properties 329
Analysis Connector
  properties 280, 285
analyst role 46
APIs
  Directory Services 25
  Export 189
  Mirror 189
application servers
  properties 297
approval system 190
  calculating dependencies 192, 194
  debug property 349

interactions with publishing system 192
types of dependencies 193
Approve Assets 238
Approve for Publish step action
configuring 120
Approve Multiple Assets 197, 237
vs. BulkApprover utility 197
ApprovedAssetDeps table 201, 211
ApprovedAssets table 201, 211
Approver role 46
approving assets
automatic upon delete 193
multiple assets 197
Arguments field (publishing destination) 219, 230
arguments, publishing
DIR 202, 204, 215
OLDTEMPLATE 203
REMOTEPASS 211
REMOTEUSER 211
SIMPLEDIR 203, 204
SIMPLENAME 202, 206
SUFFIX 202, 206
URLPREFIX 201
VERBOSE 203, 211
asset invalidation 228
asset types 258
Burlington Financial 143
Burlington Financial Extension 157
determining locale for 168
disabling for search engines 273
disabling for sites 70
disabling revision tracking for 264
enabling for Content Server Desktop 172
enabling for Content Server DocLink 179
enabling for search engines 272
enabling for sites 69
enabling revision tracking for 261
GE Lighting 150
Hello Asset World 139
indexing differences 271
when missing from the destination 251
AssetPublication table 209
AssetPublishList table 210
AssetRelationTree table 209
assets
calculating dependencies 194
clearing checkouts (administrator) 265
held from being published 193
locked during publish 198
marked as changed on destination after being published 210
mirror queue and basic assets 209
mirror queue and flex assets 210
assignment methods 102
assignments
clearing 88, 135
cross-site 94
deadlines 90
delegating 88
described 84
associations 82
attributes, user
e-mail 36
locale 36
authentication properties 297
Author role 46
auxiliary tables
examples 212
for asset types 234, 237

B

base export directory 217
Export Assets to XML 229
Export to Disk 204
basic assets
dependencies 194
mirror queue 209
batch host 216
error when not configured correctly 254
batch process properties 287
batch user
ACLS needed 216
creating 216
described 197
error when not configured correctly 253
binary visitor attributes
where stored 383
bit mask numbers for ACL permissions 29
blob (binary large object)

file for when exported 202
BlobServer
cache settings 302
properties 302
security 55
security setting 300
Bobo user name 37
Browser ACL 30
BulkApprover utility 197
Burlington Financial Extension 155
asset types 157
roles 155
start menu items 158
tree tabs 160
users 155
workflow 161
Burlington Financial sample site 142
asset types 143
Normal Article Process workflow 111
roles 45
sources 145
start menu items 145
tree tabs 147
users 36
users, ACLs, roles 142

C

CacheManager 211
communicates with CS-Satellite 210
refresh the page cache after Mirror to Server publish session 210
catalogs, *See* database tables
categories 82
CGI path 227
Checker role 46
checkins
enabling with revision tracking 258
explicit 259
implicit 259
checkouts
clearing 258
clearing with administrative feature 265
enabling with revision tracking 258
explicit 259
implicit 259
clearing
checkouts 258
deadlocks 87
workflow assignments 135
cluster
properties 303
Coco user name 37
conditions
creating 119
deleting 121
described 100
Example Step Condition 100
planning 100
configuring
Export Assets to XML 229
Export to Disk 217
function privileges 131
Mirror to Server 222
publishing 216
security measures 56
sites 69
system default locale 166
tree tabs 76
Content Server
database 379
e-mail feature 117
user 37
Content Server Desktop
.dot files 178
configuring 172
Content Server toolbar 173
creating start menu items 174
described 170
installing the client application 176
RemoteClient ACL 31
required ACLs 40
testing configuration 177
URL for logging in 176
users 175
Content Server Direct
disabling certain forms from the delivery system 59
properties 346
required ACLs 39
Content Server Direct Advantage
properties 362
Content Server DocLink
configuring 179

creating start menu items 183
described 178
document types 182
installing the client application 184
required ACLs 40
specify fields 180
testing configuration 185
URL for logging in 184
users 183
Content Server Engage
properties 377
required ACLs 40
Visitor ACL 31
VisitorAdmin ACL 31
Content Server Satellite
properties 326, 338
symptoms when not configured correctly 254
content tables
properties 304, 307
ContentEditor ACL 30
conventions
directory names with Export to Disk 203
Export Assets to XML file names 215
Export to Disk file names 206
creating
ACLs 32
batch user 216
e-mail objects 117
end step 130
export starting points 219
publishing destination for Export to Disk 218
publishing destination for Export to XML 230
publishing destination for Mirror to Server 224
roles 48
search start menu items 75
sites 67
start menu items of type New 72
tree tabs 76
user profiles 41
users 41
workflow actions 119
workflow conditions 119
workflow process 125
workflow start step 127
workflow states 123
workflow steps 128
cross-site assignments 94
CS-Desktop, *See* Content Server Desktop
CS-Direct Advantage, *See* Content Server Direct Advantage
CS-Direct, *See* Content Server Direct
CS-DocLink, *See* Content Server DocLink
CS-Engage, *See* Content Server Engage
CS-Satellite, *See* Content Server Satellite

D

database
properties 307
tables that grow quickly 379
DataBase Loader
properties 284
database tables
ACLs 28, 34
auxiliary tables for Initialize Mirror 212
disabling revision tracking 264
enabling revision tracking 261
revision tracking 260
that grow quickly 379
unlocking revision tracked rows 265
VMVISITOR 383
VMVISITORALIAS 383
VMVISITORSCALARVALUE 383
VMVISTORSCALARBLOB 383
VMz 383
deadlines
assignment 90
process 90
deadlock actions 100
planning 100
Send Deadlock Email 100
deadlocks
actions 87
defined 86
group deadlocks 88
preventing 87
resolving 87
debugging properties 349
default tree tabs 66
DefaultReader user 30, 37
and security 55

delegate actions 98
  planning 98
deleting
  ACLs 34
  e-mail objects 119
  function privileges 134
  properties 278
  roles 48
  sites 69
  steps 134
  tree tabs 79
  users 43
  users profiles 42
  workflow actions 121
  workflow conditions 121
  workflow processes 133
  workflow states 124
delivery system
  defined 14
  moving a site to 236
  security goals 56
Delivery Type field (publish destination) 219, 230
delivery types 191
dependencies, approval
  basic assets 194
  CSElement and SiteEntry assets 195
  exact 193
  exists 193
  flex families 195
  visitor data assets 196
dependencies, compositional
  unknown 200
Designer role 46
Destination Address (publish destination) 225
destination directory
  Export Assets to XML 229
  Export to Disk 204
destinations, publishing 191
  error message when incorrect 251
development system
  defined 14
  security goals 56
DIR publishing argument 202, 204, 215
directories
  base export directory, Export to Disk 204
  root storage for revision tracking 261
directory names
  conventions for Export to Disk 220
  examples for Export to Disk 205
Directory Services API 25
disk cache
  properties 322
due date *See* deadlines

E

editing
  ACLs 33
  e-mail objects 118
  functions privileges 132
  publish destinations 239
  publish events 243
  roles 48
  sites 68
  tree tabs 79
  user profiles 41
  users 42
  workflow actions 121
  workflow conditions 121
  workflow processes 132
  workflow states 124
  workflow steps 132
Editor role 47
editor user name 37
ElementEditor ACL 31
ElementReader ACL 31
elements
  workflow 95
e-mail
  enabling the Content Server e-mail feature 117
  properties 314
  specifying which user attribute holds address 358
e-mail objects 95
  Assignment Due Reminder 97
  Assignment Message 97
  creating 117
  deleting 119
  described 96
  editing 118

planning 97
emailname workflow variable 95
end step
creating 130
error messages
publishing 249
EvalServer
properties 302
events
APPEVENT 384
publish 197
timed action event 122
eWebEditPro
configuring 187
users 187
exact dependency 193
exists dependency 193
Expert role 47
explicit checkins and checkouts 259
Export API 189
Export Assets to XML publishing 191
configuring 229
creating a publishing destination 230
file names 215
specifying the base export directory 229
testing the results 231
export starting points 200
creating 219
defined 208
how many do you need? 208
when OLDTEMPLATE is set to true 220
Export to Disk publishing 191
calculating dependencies 194
configuring 217
creating a publishing destination 218
creating export starting points 219
described 199
directory-naming conventions 203
export directory 204
export queue 200
file names 203
publishing arguments 201
specifying the base export directory 217
testing the results 221

F

file names, Export Assets to XML 215
file names, Export to Disk
blobs 202
conventions 206, 220
examples 207
filename field 206
firewall server
IP address 316
port number 317
flex assets
creating the mirror queue for flex assets 210
dependencies 195
Flo user name 38
flushing
inactive data 384
session objects (scratch) table 384
Force Specified Filename field (export starting point) 220
Force Specified Path field (export starting point) 220
From State 85, 102
function privileges
configuring 131
defined 105
deleting 134
described 89
editing 132
planning 107
futuretense.txt file 250
debug properties that specify what gets written to it 312
where located 312

G

GE Lighting sample site 148
asset types 150
roles 45, 148
start menu items 151
tree tabs 153
users 36, 148
workflow process 113
general error message 251
GeneralAdmin role 45
group deadlock actions 101

planning 101
Send Deadlock Email 101
groups, workflow 88
synchronize steps 88
tab 78

H

held assets 193
Hello Asset World 138
asset types 139
HelloArticle workflow process 109
roles 138
start menu items 140
tree tabs 141
users 138
HelloArticle workflow process 109
HelloAuthor role 47
HelloDesigner role 47
HelloEditor role 47
history
publishing 249
History tab 66, 141, 147, 154, 161
history types
where stored 383
HTTP
properties 320

I

IDs
visitor 383
implicit checkins and checkouts 259
inactive visitor data 384
indexes
how created for asset types 271
rebuilding for asset types 273
Initialize Mirror Destination 212
moving a site from development to management 233
moving a site to the delivery system 236
setting up a mirror destination 226
InSite Editor
configuring 185
property 358
required ACLs 40
users 186
installing
Content Server Desktop client 176
Content Server DocLink client 184

J

Java Server Pages, *See* JSP
Joe user name 38
JSP
properties 318

K

KeyView properties 351

L

LDAP
properties 335, 336
locale
asset types 168
Content Server Desktop 176
Content Server interface 166
properties 167
single-language restrictions 168
specifying system default 166
locked assets, clearing 265
log files
futuretense.txt 250
publish history 249
logging, message
properties 337

M

management system
defined 14
moving a site to 233
security goals 56
Marketer role 47
Marketing tab 66
message logging
properties 337
migrating
a site from one system to another 232
mimetypes 82, 182
Mirror API 189
mirror queue
for basic assets 209
for flex assets 210
unpacking the items in the queue 210

Mirror to Server publishing 191
  calculating dependencies 194
  configuring 222
  creating a publishing destination 224
  described 208
  identifying the mirror user 223
  initializing a target destination 226
  mirror queue 209
  proxy servers 224
  publishing arguments 211
  setting up the target system 222
  testing the results 227
mirror user
  creating 222
  described 211
  error messages for 250
  identifying 223
mirroruser user name 38
missing table publishing error 251
Moe user name 38
moving
  a site from one system to another 232

N

Normal Article Process 111

O

OLDTEMPLATE argument 203

P

page cache
  refreshed after Mirror to Server 210
PageEditor ACL 31
PageReader ACL 31
participants
  cross-site 94
  described 84
  determining how selected 94
paths
  for links within exported files 206
  precedence of path information for export 204
performance
  ft.filecheck property 326
permissions
  ACL 28
planning
  deadlock actions 100
  delegate actions 98
  e-mail objects 97
  function privileges 107
  group deadlock actions 101
  roles 93
  step actions 99
  step conditions 100
  timed actions 98
  workflow processes 93
  workflow states 101
  workflow steps 102
preferences
  properties 351
Preserve formatting option 171
preview
  separate browser sessions for 187
Pricer role 47
process, workflow, *See* workflow
properties
  adding 277
  administrator 357
  AltaVista 329
  Analysis Connector 280, 285
  application server 297
  asset marked as not approved on destination after publish 228
  authentication 297
  batch 287
  cluster 303
  content table 304, 307
  CS-Direct Advantage 362
  CS-Engage 377
  CS-Satellite 326, 338
  database 307
  DataBase Loader 284
  deleting 278
  e-mail 314
  EvalServer 303
  HTTP 320
  InSite Editor 358
  JSP 318
  KeyView 351
  large text fields 309
  LDAP 335, 336
  message logging 337

preferences 351
publishing 316, 353
resultset caching 323, 331
search engine 328
security 56, 300
setting 276
Transact 367
URL columns 321
Verity 329
visitor data 375
Property Editor
adding properties 277
deleting properties 278
setting properties 276
starting 276
proxy servers
and Mirror to Server 224
PubKey table 201
Publish Console 246
examining publishing history 247
publish events 197
creating 240
editing 243
example 241
overriding 243
reading the schedule abbreviations 242
publish point, *See* export starting point
published references
Export to Disk 248
Mirror to Server 248
PublishedAssets table 200, 201
publishing
See also publishing arguments, publishing destinations, publishing schedule, publishing session, and publish events
all approved assets, regardless of publish status 249
auxiliary tables 226
configuring a system to publish 216
defined 189
dependencies 192
directory path for exported files 202
directory path for exported XML files 215
enable error logging with VERBOSE 203
error messages 249
Export Assets to XML destinations 230
export destinations 218
export directory for Export Assets to XML 229
export directory for Export to Disk 217
history 247
history log files 249
mapping URL prefixes 219
messages 249
Mirror to Server destinations 224
properties 316, 353
setting mirror usernames and passwords 223
testing sites 221, 231
troubleshooting 249
publishing arguments
Export to Disk 201
Mirror to Server 211
publishing destinations
approving multiple assets 237
creating for Export to Disk 218
creating for Export to XML 230
creating for Mirror to Server 224
described 191
editing 239
initializing 226
publishing errors
assets with associations and DB2 255
cannot contact the destination 252
cleanup failed 252
complex or flex asset could not be delivered 252
complex or flex asset could not be saved on destination 253
incorrect ACLs for mirror user 250
incorrect username or password for mirror user 250
mirror destination is incorrect 251
missing table on destination 251
pages not refreshed after publish 254
session does not end 253
session failed 252
system stops 254
publishing method, *See* delivery type
publishing schedule 197, 240
See also publish events
interpreting 242
publishing sessions

amount of disk space needed for 254
described 198
displayed in Publish Console 246
publishing system
described 190
export queue 200
firewalls 224
interactions with approval system 192
mirror queue 209
testing results for Mirror to Server 227
PubMessage table 249
PubSession table 250
PubSessionID
in publish history files 249
in PubSession table 250
purging
inactive data 384
session objects (scratch) table 384

Q

queries
for approved assets 198
queues
export publish queue 200
mirror publish queue 209

R

references
published references 248
RemoteClient ACL 31
REMOTEPASS publishing argument 211
REMOTEUSER publishing argument 211
resolving
deadlocks 87
links 201
resultset caching
properties 323, 331
revision tracking
and non-asset tables 260
changing the number of versions to store 262
defined 257
disabling for asset types 264
disabling for non-asset database tables 264
editing settings 262
enabling for asset types 261
enabling for non-asset database tables 261
implicit and explicit checkouts and checkins 260
root storage directory 262
unlocking assets 265
unlocking rows in non-asset tables 265
roles
and users for sites 50
assigning to users 49
Burlington Financial 142
Burlington Financial Extension 155
creating 48
default system 45
deleting 48
described 44
determining for workflow 93
editing 48
GE Lighting 148
Hello Asset World 138
needed by administrators 45
place in user management 24
relationship with ACLs 28, 45
required to configure workflow 92
sample site 45
start menu items 65
rollback
defined 258

S

sample sites
configuration descriptions 137
roles 45
users 36
workflow processes 109
scalar objects 383
scheduling
publish events 197
timed action event 122
search
start menu items 75
search engines 270
disabling asset types for 273
enabling asset types for 272
indexing differences between asset types 271
properties 328

rebuilding indexes for asset types 273
Searching option 270
security
access control for assets 107
ACLs 54
BlobServer 55
cc.security property 54
changing default passwords 57
configuring 56
DefaultReader 55
disabling CS-Direct forms on delivery system 59
function privileges and assets 89
goals for various systems 56
properties 56, 300
testing 59
URLs 58
separate browser sessions for preview 187
setting field values
start menu items 74
SIMPLEDIR publishing argument 203, 204
SIMPLENAME publishing argument 202, 206
simplified access control 107
single-language restrictions 168
Site Admin tab 66
xceladmin ACL 32
site designers
ACLs they need 40
Site Plan tab 66
SiteCatalog table
ACLs 28
file names for exported assets 206
SiteGod ACL 31
SitePlanTree table 64
sites
configuring 69
creating 67
defined 64
deleting 69
disabling asset types for 70
editing 68
enabling asset types for 69
role in user management 24
roles 44
SitePlanTree table 64
sources 82
Burlington Financial 145
defined 192, 316
Specify Path/Filename link 220
start menu items
Burlington Financial 145
Burlington Financial Extension 158
Content Server Desktop 174
Content Server DocLink 183
creating for the New list 72
defined 65
field values set by 65
for Search list 75
GE Lighting 151
Hello Asset World 140
setting field values with 74
types 71
start step
See also steps
creating 127
described 85
states
creating 123
deleting 124
described 84
editing 124
From State 85, 102
planning 101
To State 85, 102
step actions 98
Approve for Publish 99
configuring Approve for Publish 120
planning 99
Send Assignment Email 99
Send Rejection Email 99
step conditions
creating 119
described 100
Example Step Condition 100
planning 100
steps
actions 85, 95, 98
all-voting step 86
conditions 85
creating 128
deleting 134

described 85
editing 132
end step 130
planning 102
start step 85
start steps 127
synchronize step 88
SUFFIX publishing argument 202, 206
SystemACL table 29
SystemInfo table 28

T

TableEditor ACL 31
tables
VMVISITOR 383
VMVISITORALIAS 383
VMVISITORSCALARVALUE 383
VMVISTORSCALARBLOB 383
VMz 383
tabs, tree, *See* tree tabs
target, *See* publishing destinations
targets
defined 316
variable 95, 121
testing
security measures 59
The 78
timed action event
scheduling 122
timed actions 95
described 97
examples 84
planning 98
Send Email 98
To State 85, 102
Transact properties 367
tree tabs
Burlington Financial 147
Burlington Financial Extension 160
creating 76
default 66
defined 66
deleting 79
editing 79
example of custom tab 66
GE Lighting 153
Hello Asset World 141
kinds of 76
roles 44
sort order of 79
workflow groups 78
troubleshooting
locale problems 167
publishing 249

U

unknown dependencies
Export to Disk 200
URL columns
properties 321
URLPREFIX argument 201
URLs
determining so you can test a site 227
for CS-Desktop 176
for CS-DocLink 184
in HREFs of exported pages 206
mapping URLs for specific servlets 58
user attributes 36
adding 43
user management
and sites 64
authentication properties 297
overview of concepts 24
user profiles 36
creating 41
deleting 42
user_anaylst 38
user_approver 38
user_author 38
user_checker 38
user_designer 38
user_editor 38
user_expert 39
user_marketer 39
user_pricer 39
UserEditor ACL 31
UserReader ACL 31
users 36
and ACLs 27
and required ACLs 39
and user profiles 41

assigning roles to users 49
batch user for publishing 216
Burlington Financial 142
Burlington Financial Extension 155
changing default users for security purposes 57
creating 41
default system 36
deleting 43
editing 42
GE Lighting 148
granting users access to sites 49
Hello Asset World 138
list for a site 50
mirror user 222
role in user management 24
sample site 36

V

variables
emailname 95
for workflow e-mail objects 96
targets 95, 121
VERBOSE publishing argument 203, 211
Verity search engine
properties 329
version number 258
vis.genclasspath property 385
Visitor ACL 31
visitor attributes
binary 383
where stored 383
visitor data
inactive 384
properties 375
VisitorAdmin ACL 31
visitors
IDs 383
VMVISITOR table 383
VMVISITORALIAS table 383
VMVISITORSCALARVALUE table 383
VMVISTORSCALARBLOB table 383
VMz tables 383
voting
described 85

W

web server alias 201
Word
templates for Content Server Desktop 178
workflow
See also, actions, conditions, deadlocks, states, steps, timed action event, and workflow processes
actions 84, 85
assignment 84
conditions 85
configuring 116
deadlock 86
delegate actions 88
described 84
elements 95
e-mail objects 95
e-mail variables 96
ending 92
function privileges 89
group deadlock action 88
groups 88
moving your work 135
participants 84
placing an asset in workflow 89
planning 93
process 84
required roles to configure 92
roles 44
sample site processes 109
set with a start menu item 65
states 84
steps 85
testing your process 134
variables 95
voting 85
Workflow Groups tab
creating 78
workflow process
creating 125
deadlines 90
defined 84
deleting 133
editing 132
Workflow tab 66
xceladmin ACL 32

X

xceladmin ACL 32
xceleditor ACL 32