Microsoft® Content Management Server 2001

Site Programmer's Guide

Microsoft®

Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording or otherwise), or for any purpose, wihout the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement with Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2001 Microsoft Corporation. All rights reserved.

Microsoft, Windows, Windows NT, MSDN, Site Server, Active Directory, NetShow, JScript, FrontPage, ActiveX, Visual Basic, and Visual InterDev are registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Table of Contents

Using This Guide

Who should use this guide v
How to use this guide vi
Conventions used in this guide viii
For more information ix

chapter 1 Elements of an MSCMS 2001 Web Interface

Channels and postings 3
Introduction to the MSCMS 2001 Publishing API 4
Navigation templates 5
Framed MSCMS 2001interfaces 5
Frameless MSCMS 2001 interfaces 6
Cover pages 7
Web server files 7
Authentication 7
Customizing your MSCMS 2001 interface 9

chapter 2 Creating a Framed MSCMS 2001 Interface

Introduction 13
Viewing the framed navigation tutorial interface 13
Framed sample interface overview 14
Development process summary 15
Resolution.inc 16
Creating the frameset 17
Creating the navigation template 19
Setting the Frameset URL 28
Previewing the template 29
Approving the template 30
Assigning the template to a channel 31
Viewing the channel in a Web browser 33
Step-by-step through the run-time execution 33

chapter 3 Creating a Frameless MSCMS 2001 Interface

Introduction 37
Frameless Navigation Tutorial interface overview 38
Important information about frameless interfaces 39
Development process summary 39
Resolution.inc 39

Creating the navigation template. 40
Creating a new template 41
Previewing the template 46
Approving the template 46
Assigning the template to a channel. 46
Viewing the channel in a Web browser 49
Creating the page template outline 49
Step-by-step through the run-time execution 53
Extending your interface 54

chapter 4 **Using the MSCMS 2001 Publishing API**

Introduction 57
About MSCMS 2001 URLs 57
Ensuring proper display of templates 57
About simulated channels 58
Dynamically generating a "What's New" page 59
Dynamically generating a "What's Important" page 60
Preventing unauthorized client-side script from executing 61
Switching templates 63
Sorting a list of channels or postings 65

chapter 5 **Working within the MSCMS 2001 Site Builder**

Opening the Site Builder 71
Configuring an HTML editor 71
Locating navigation templates in MSCMS 2001 74
About the template Design Palette 75
Quick start to template creation 75
Copying a template. 75
Editing the HTML of an existing template. 75
Creating a new template 76
Assigning custom properties to your template 77
Setting custom properties. 77
Guidelines for working with custom properties 78
Importing templates 82
Using external resources 83
Setting the Frameset URL 83
Previewing the template 84
About template workflow 84
Approving templates 86
Reverting a Saved template to its Approved version 88
Generating a Dependant Report 89

Deleting a template vs. purging a template 89
Deleting a template 89
Setting a channel's Cover Page URL property 90
Using the Find feature 92

chapter 6 **Site Programmer's FAQ**

How to distinguish between page and navigation templates 97
How do I redirect to a different template? 97
Can I use Java or ActiveX with a template? 98
What does a template's status represent? 98
What is a frameless page template outline? 98
What is MSCMS guest user access? 98
Where do the objects in Preview mode come from? 98
How do I control browser caching? 99

chapter 7 **Introducing the MSCMS 2001 Web Author**

What is the MSCMS 2001 Web Author? 103
Understanding the MSCMS 2001 publishing system 103
About setting up publishing 105
Location of the Web Author files 105
Getting started 105

chapter 8 **Working from the Sample Web Author**

About customization 113
Understanding Web Author directory structure 113
Understanding the authoring process 126
States of pages and postings 133

chapter 9 **Customizing Your MSCMS 2001 Web Author**

Customization example #1: re-designing the interface 137
Customization example #2: multilingual support 139
Best Practices 144

chapter 10 **Developing for MSCMS 2001 Site Stager**

MSCMS 2001 Site Stager compatibility 153
Site Stager redirection 155
Sample script for staging 156

chapter 11 **Implementing Full-Text Search**

Setting up full-text search. . . . 169
Supported search engines 169
Redirection 169
Setting the title of a page template 170
Considerations for framed sites. . . . 170
Using consistent URLs. . . . 172
Linking to the search page 172

chapter 12 **Content Connector Directory Structure and Framework**

Overview 175
Understanding Content Connector directory structure 176
Understanding the Content Connector Framework API 178

chapter 13 **Personalizing a Content Connector Site**

Personalization 183
Using personalized content objects 184
Page fragment targeting 188
Personalization using MSCMS 2001 security 189

chapter 14 **Accessing the Content Connector Product Catalog**

Accessing the product catalog 193
Managing the product catalog. . . . 193
Product catalog interfaces. . . . 193
Creating product templates 195
Product Search. . . . 197

chapter 15 **Accessing Other Commerce Server 2000 Objects**

Accessing other Commerce Server 2000 objects. . . . 201

appendix A **Query String Parameters**

General query string parameters. . . . 205

Glossary 211

Index 227

Using This Guide

This guide covers all the information required to work with navigation templates in the Microsoft® Content Management Server 2001 Site Builder and Microsoft® Content Management Server 2001 Web Author.

Who should use this guide

The site programmer role is a subset of the template designer role. In the Site Builder interface, the site programmer is distinguished only by their possession of rights to the template galleries containing navigation templates. In reality, site programmers are a distinct class of Content Management Server 2001 user. They are responsible for creating navigation templates and other Active Server Pages (ASP) files that provide the Web front-end for a Content Management Server 2001 site, and the Web Author. The site programmer is also responsible for creating Microsoft Content Management Server 2001 Content Connector for Commerce Server 2000 templates, personalizing a Content Connector site, and accessing the product catalog and other Commerce Server 2000 objects.

It is assumed that site programmers have experience with:

- HTML
- ASP scripting using VBScript
- basic HyperText Transfer Protocol (HTTP) server administration
- client-side JavaScript.

To fully understand examples in the Content Management Server 2001 sample Web sites, experience with Dynamic HTML (DHTML) is also required.

How to use this guide

To do this...	Go to this chapter
Understand the basics of Microsoft Content Management Server 2001 Web interfaces and the Application Programming Interface (API).	Chapter 1, "Elements of an MSCMS 2001 interface"
Walk through the coding and components used to create a basic framed or frameless interface.	Chapter 2, "Creating a Framed MSCMS 2001 Interface" and Chapter 3, "Creating a Frameless MSCMS 2001 Interface"
Extend and customize the functionality of your basic site. Topics include dynamically switching navigation templates based on the subscriber's browser, and dynamically generating a list of new postings.	Chapter 4, "Using the MSCMS 2001 Publishing API"
Work with the Site Builder, where navigation templates are stored and applied.	Chapter 5, "Working within the MSCMS 2001 Site Builder"
Address common questions about navigation programming.	Chapter 6, "Site Programmer's FAQ"
Understand the Web authoring tool.	Chapter 7, "Introducing the MSCMS 2001 Web Author"
Review the files that make up the sample Web Author and the methods called when an author creates a posting.	Chapter 8, "Working from the Sample Web Author"
Customize your Web Author by redesigning the look, creating a multilingual interface, and optimizing the ASP scripts.	Chapter 9, "Customizing your MSCMS 2001 Web Author"
Use the Site Stager on your site.	Chapter 10, "Developing for MSCMS 2001 Site Stager"
Make your site indexable, and link to local search pages.	Chapter 11, "Implementing Full-Text Search"

To do this... (Continued)	Go to this chapter (Continued)
Understand the Content Connector directory structure and important framework API objects	Chapter 12, "Content Connector Directory Structure and Framework"
Personalize a Content Connector site using personalized content objects and page fragment targeting.	Chapter 13, "Personalizing a Content Connector Site"
Access and manage the Commerce Server 2000 product catalog and create product templates.	Chapter 14, "Accessing the Content Connector Product Catalog"
Access Commerce Server 2000 support pages and processing scripts, analyze Content Connector sites, and integrate Content Management Server 2001 authentication with user profiles.	Chapter 15, "Accessing Other Commerce Server Objects"
Understand what the query string parameters used by the Web Author represent, and when they are used.	Appendix A, "Query String Parameters"

Conventions used in this guide

We have used several document conventions in this guide to help you identify and find information.

Two components of the Microsoft® Content Management Server 2001, the MSCMS 2001 Site Builder and MSCMS 2001 Web Author are always capitalized. However, references to the server and database are not capitalized: MSCMS 2001 server and MSCMS 2001 database.

Information that requires special attention comes in three types as described below: Caution, Tip, and Note.

! Caution
Caution means reader be careful. In this situation, you might do something that could result in loss of data or degradation of product performance.

Tip
Tips provide helpful suggestions for using Content Management Server effectively. Tips are not necessary to the basic understanding of the text.

Note
Note means reader take note. Notes emphasize or expand on important points in the main text around them. Notes can also provide information that applies only in special cases.

We also use typographical conventions to indicate specific information types in the documentation.

Convention	Usage in documentation
Bold	This style indicates menu items and buttons that must be clicked or selected. For example, "Once you have completed the form, click **Finish** to move to the next screen."
Italic	This style indicates the title of a book or manual. Most commonly it is used to refer to other documentation in the MSCMS 2001 information product suite.
`Monospace`	This style denotes source code and names of programming functions. For example, "When implementing image tags in HTML be sure to always use width and height attributes, as shown below." `<IMG SRC="PowerPanels.GIF" WIDTH="150" HEIGHT="150">`

For more information

The information products for Content Management Server 2001 are included on the product CD and include the following:

- *MSCMS 2001 Publishing API Help*: detailed reference information about the Publishing Application Programming Interface (API) exposed by the MSCMS 2001 server. This API is used by Web programmers in building an MSCMS 2001 site.
- *MSCMS 2001 Content Connector for Commerce Server 2000 Framework API Help:* detailed reference information about the framework for the MSCMS 2001 Content Connector for Commerce Server 2000 API. This API is used by Web programmers in building a Content Connector site.

The above information is in the form of compiled help (.chm) files.

The following information comes in Portable Document Format (PDF). You need Adobe Acrobat reader to access the PDFs—download a free copy from the Adobe site at `www.adobe.com`.

MSCMS 2001 Setup Guide: provides prerequisite information for installing Content Management Server 2001, with the InstallShield® Wizard, installing a database, configuring your system with the SCA, installing the MSCMS 2001 Content Connector for Commerce 2000 and unpacking a site, and installing the MSCMS 2001 Site Stager.

MSCMS 2001 Administrator's Guide: assists site administrators in the technical setup of the customer Web site using Content Management Server 2001 and MSCMS 2001 Content Connector for Commerce Server 2000.

MSCMS 2001 Page Template Designer's Guide: provides details about all aspects of template-based publishing, including creating templates for use in a multi-authoring environment.

MSCMS 2001 Content Connector for Commerce Server 2000 Business Desk User's Guide: provides information about the MSCMS 2001 Content Connector for Commerce Server 2001 add-on and explains how to personalize your site's content. This includes creating and managing page profiles, personalized content objects, and rich content.

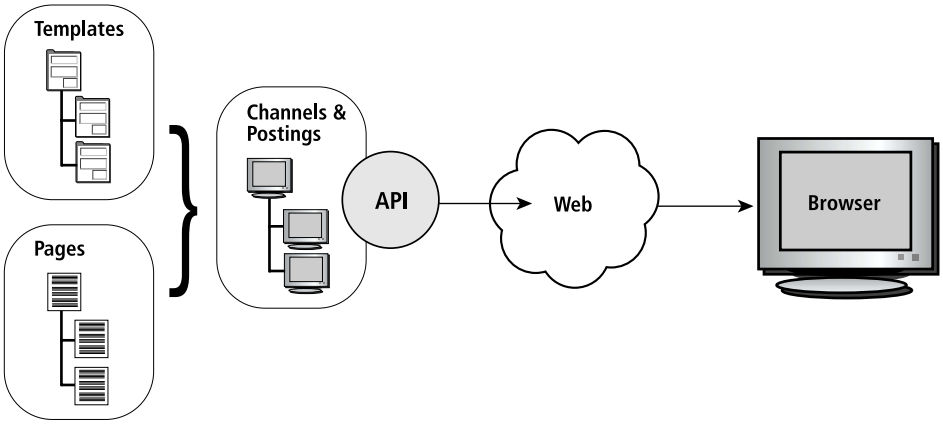
Check our corporate Web site at `www.microsoft.com/cmserver`, for a variety of useful information.

Chapter 1

Elements of an MSCMS 2001 Web Interface

Channels and postings

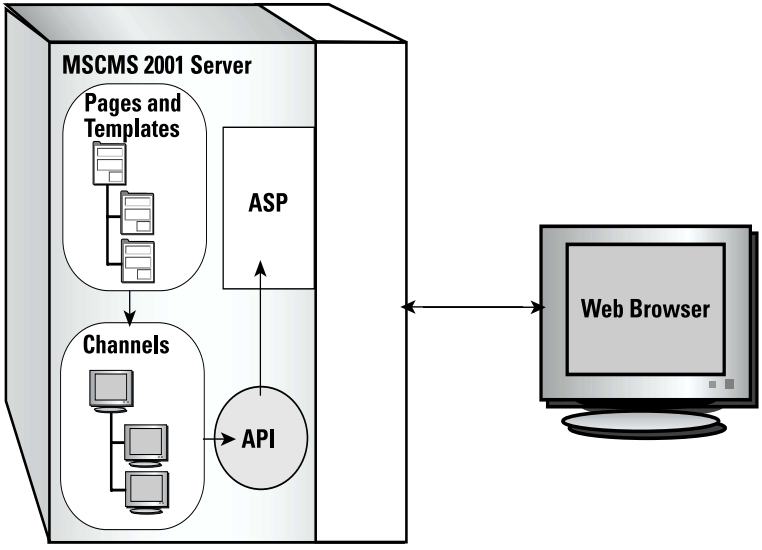
Authors publish content by creating postings in channels. A posting is a dynamically generated HTML page, made up of a Microsoft® Content Management Server 2001 (MSCMS 2001) page template and the content added to the template. The Content Management Server 2001 channel hierarchy acts like the virtual directory hierarchy of a file-based Web site; channels are containers for organizing published postings for the end-user.



MSCMS 2001 templates and channels are created with the MSCMS 2001 Site Builder, and are stored in the MSCMS 2001 database. Postings are created with the MSCMS 2001 Web Author, and are also stored in the database. MSCMS 2001 sites are dynamically generated by Active Server Pages (ASP) that interface with the server's Publishing Application Programming Interface (API).

Introduction to the MSCMS 2001 Publishing API

The MSCMS 2001 Publishing API is the basic interface to the underlying server object, and provides a layer of abstraction to system elements that shields site programmers from schema and architecture changes.



The Publishing API provides interfaces to postings and channels within a database, allowing a Web server to retrieve (for example):

- the name and URL of the selected channel or posting
- the current subchannels of the selected channel.

The Publishing API can be called from any scripting language supported by ASP, including server-side JScript and VBScript. Documentation for all Publishing API objects, properties, and methods is in the *MSCMS 2001 Publishing API Help*.

Navigation templates

Navigation templates are MSCMS 2001 ASP files that play a key role in defining Web interfaces.

Every MSCMS 2001 channel is assigned a navigation template. When a channel or posting in that channel is requested by IIS, the URL that gets returned and executed is determined by the navigation template assigned to the channel, or to the posting's channel.

To allow for the creation of framed and frameless Web interfaces for channels, there are two types of navigation templates: those with a frameset URL, and those set to frameless.

Framed MSCMS 2001interfaces

To create a framed interface for an MSCMS 2001 channel, you assign a frameset to a navigation template, then assign the navigation template to the channel. When the URL of the channel is requested by IIS, MSCMS 2001 returns the URL of the frameset assigned to the navigation template.

In the Framed Sample interface, the frameset file defines two frames. The navigation template itself is displayed in the left frame (it contains ASP scripting to create links to the parent channel and sub-channels). When the link to the channel is selected, the channel's cover page is displayed in the right frame (see "Cover pages" in this chapter for more information).

When a posting is selected, the frameset reloads, this time displaying the posting's content instead of the channel's cover page.



See Chapter 2, "Creating a Framed MSCMS 2001 Interface," for a step-by-step discussion of the development process you would use in creating a sample framed site.

Frameless MSCMS 2001 interfaces

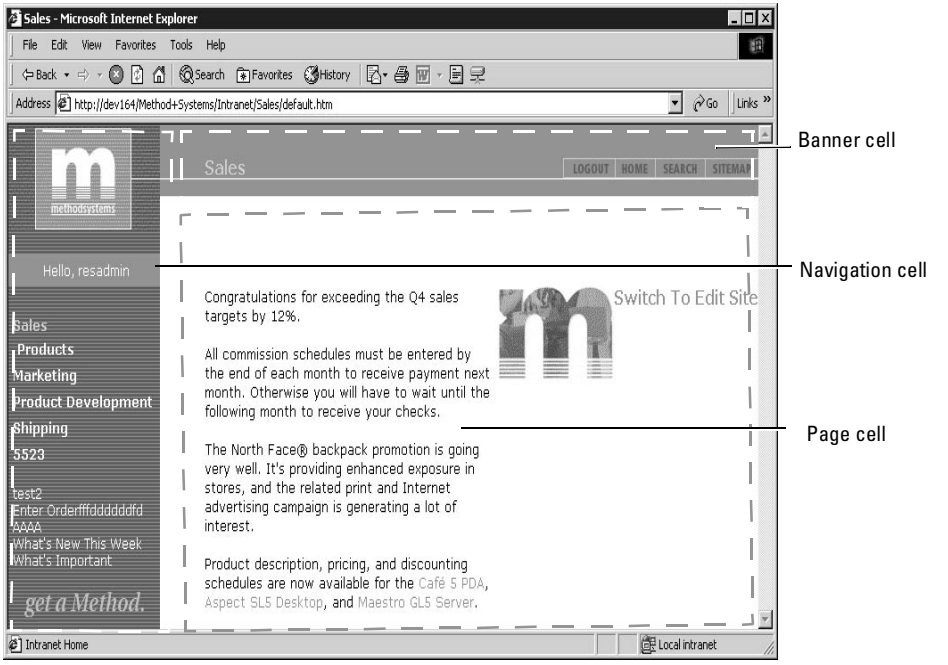
Frameless interfaces display only one ASP file at a time. This ASP file is either a page template (when a posting is selected) or a navigation template (when a channel is selected).

Because frameless navigation templates are displayed only when a channel is selected, they are usually designed to either:

- display channel navigation, or
- redirect to a posting in the channel.

Displaying channel navigation

The sample frameless interface below displays navigation context for the channel. The navigation template is split into three cells. The top cell has an ASP has file that defines common links. The left cell displays a graphic and title. The cell on the right includes another ASP include file with scripting to display the channel hierarchy. If a channel is selected, its navigation template replaces this page. The navigation templates will redirect users to the channel's default posting. If a posting is selected, its page template replaces this page. See "Development process summary" in Chapter 3, "Creating a Frameless MSCMS 2001 Interface" for a step-by-step discussion of the development process you would use in creating a sample frameless site.



Sample frameless interface

Cover pages

A posting or an external file can be assigned to a channel's Cover Page URL property. The URL assigned to this property is returned by the Publishing API `Channel.URLPostingFrame` property. If the Cover Page URL is not set for a channel, `URLPostingFrame` returns the URL of the default cover page, /NR/Shared/Cover.asp. The default cover page displays the selected channel's name and description.

Framed interfaces often use the Cover Page URL as the source for the content frame when a channel is selected.

In a frameless interface, it doesn't make sense to assign a Cover Page URL to be an external file. If used at all, the Cover Page URL property could be set to a posting within the channel. The channel's navigation template is then scripted to redirect to this posting. See "Setting a channel's Cover Page URL property" in Chapter 5, "Working within the MSCMS 2001 Site Builder" for more information about cover pages.

Web server files

Some files needed to create an MSCMS 2001 Web site are stored on the Web server file system. There is only one virtual directory mapping in the file structure which maps "NR" virtual directory to "<target-dir>\IIS_NR" physical directory.

Most important for navigation programmers are the Access and Shared folders. These are in the server's NR directory; for example:

```
C:\Program Files\Microsoft Content Management Server\Server\IIS_NR\System\Access\
```

Access

The access folder contains the ASP files required to do subscriber authentication. We do not recommend modifying these files because doing so may compromise your system's security.

Shared

The shared folder contains files used throughout an MSCMS 2001 site and not specific to any one interface.

Authentication

MSCMS 2001 releases data to authenticated users only. Authentication is handled through a combination of files referenced by one file in particular: Authenticate.inc. which is referenced by Resolution.inc.

Modifying Authenticate.inc may compromise the access restrictions you have set on your MSCMS 2001 site. For this reason, we do not recommend changing Authenticate.inc.

Frameless sites

Resolution.inc is automatically added at the top of all MSCMS 2001 templates. Frameless sites, which are based on MSCMS 2001 templates, are automatically authenticated.

Framed sites

Resolution.inc must be included manually in ASP files, such as the frameset file, that are stored on the Web server. To include Resolution.inc, use a server-side include:

```
<!--#include virtual="/NR/System/Access/Resolution.inc"-->
```

Authentication consists of three phases as follows.

Phase one - Verify user as a valid Windows 2000, Active Directory, or Site Server user

Except in the case of the Site Server login and manual login pages, failure in one of the steps in this phase simply moves the user to the next step. The user proceeds to Phase two after the user's ID has been successfully verified by any of these steps:

1 Attempt to authenticate user based on a cookie stored from a previous successful Windows 2000, Active Directory (AD), or Site Server login.

2 Attempt to log the user on as the guest account, if it has been configured by an MSCMS 2001 administrator.

3 If there is no cookie and Site Server is enabled, the Site Server login screen is displayed.



Note
Failure to log on to Site Server does not result in the process proceeding to the next step. A valid Site Server login must be entered before the process proceeds to Phase two.



Note
For more information about possible authenticatication schemes, including IIS authentication, consult the corporate Web site at `www.microsoft.com/cmserver` for a forthcoming White Paper.

4 If all other methods fail, the manual login page is displayed. From the manual login page a user can select a domain or distinguished name (for LDAP) and enter a user name and password with which to log on to MSCMS 2001. If a warning message is displayed, your connection to the server is not secure. Contact your system administrator if a secure connection is desired.

Phase two - Determine whether the user has rights to the requested item

MSCMS 2001 verifies the user is a member of any MSCMS 2001 rights group that has rights to the requested channel or posting.

Phase three - Verify that item is published

Before allowing subscribers to access a posting, MSCMS 2001 first checks that the posting is in a "Published" state.

See Chapter 8, "Working from the Sample Web Author," for information on enabling templates for use with the Web Author.

Customizing your MSCMS 2001 interface

MSCMS 2001 navigation templates and supporting files can include any client-side Web technology supported by your end-users' browsers. A practical example is the use of Cascading Style Sheets (CSS) to format the list of channels and postings generated by scripting. For information on template-switching, the MSCMS 2001 feature that provides the means to switch navigation templates based on properties of the browser, see "Switching templates" in Chapter 4. "Using the MSCMS 2001 Publishing API."

Chapter 2

Creating a Framed MSCMS 2001 Interface

Introduction

To show how the components of a framed interface for an MSCMS 2001 channel are created and integrated, in this chapter we step through the development process for a hypothetical interface called Framed Sample.

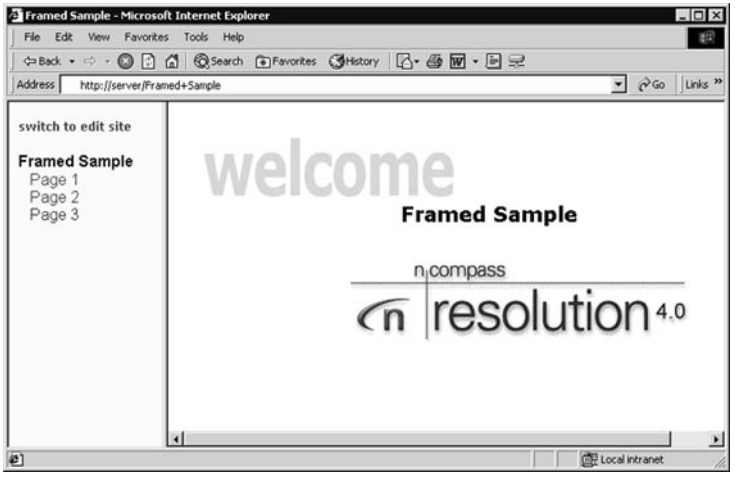
Note
Any references to the Method Systems sample site or the Framed Sample interface is intended for illustration only.

Viewing the framed navigation tutorial interface

To view the interface, open the URL for the Framed Sample channel:

```
http://<your-server>/framed+sample/
```

After confirming that you have the necessary rights to view the Framed Sample channel, MSCMS 2001 redirects you to the frameset associated with its navigation template.



The Framed Sample interface

Framed sample interface overview

Our first task was to define the interface's navigation requirements. We decided it should:

- link to the selected channel's parent channel, subchannels, and postings
- display the selected channel's cover page, or display the selected posting.

After considering the requirements, we decided to split the interface into two frames:

- the left frame displays the channel and posting hierarchy
- the right frame displays the selected posting, or, if a channel is selected, the channel's cover page.



Development process summary

The steps we followed in creating this interface were:

1 Create the Framed Sample directory.

The Framed Sample directory organizes our files on the Web server.

2 Create the frameset file.

The frameset defines the layout and source of the interface's frames. The frameset is an ASP file stored on the Web server.

3 Set the template's Frameset URL.

In the Site Builder, we set the template's frameset URL to the frameset file we created.

4 Create the navigation template.

Navigation templates are ASP files stored in the MSCMS 2001 database. Our navigation template contains code for displaying the channel and posting hierarchy.

5 Preview the template.

6 Submit the template.

We submit the template to make it available for use by channels.

7 Assign the template to the Framed Sample channel.

8 Enable the Web Author.

In the Site Builder, we set the channel's properties to enable it for the Web Author.

9 View the channel in a Web browser.

Resolution.inc

Resolution.inc references Authenticate.inc, which confirms that subscribers have rights to requested channels and postings. Resolution.inc is included automatically at the top of MSCMS 2001 templates. Resolution.inc must be included manually within ASP files on the Web server. To include Resolution.inc, use a VBScript server-side include:

```
<!--#include virtual="/NR/System/Access/Resolution.inc"-->
```

See "Authentication" in Chapter 1, "Elements of a MSCMS 2001 Web Interface" for a description of the authentication process.



Note
Modifying Authenticate.inc may compromise the access restrictions you have set on your MSCMS 2001 site. For this reason, we recommend you not change Authenticate.inc.

Initializing the Autosession object

Authenticate.inc (called from Resolution.inc) also creates a new `Autosession` object.

The `Autosession` object stores information about the page being requested, the user making the request, and other details of the session. All subsequent Publishing API scripting relies on this `Autosession` object. Refer to the *MSCMS 2001 Publishing API Help* for more information on the Autosession object, and its methods and properties.



Note
The Authenticate.inc file uses the `Option Explicit` VBScript declaration. Script errors occur if you attempt to declare `Option Explicit` a second time or if you don't explicitly declare all your VBScript variables.

Creating the frameset

The frameset file specifies the placement and content of frames in a browser.

We created the frameset file with an HTML editor, and saved it to the Framed Sample folder on the Web server. Its virtual directory path is `/NR/Framed Sample/Frameset.asp`.

The frameset defines two frames: channel and page.



The frameset displayed in a Web browser

The navigation frame

The navigation frame displays the current channel's navigation template. The navigation template contains code for displaying the channel's postings, parent channel, and sub-channels.

The posting frame

The frameset's posting frame either displays:

- a cover page

or

- a dynamically-generated page created from components stored in the MSCMS 2001 database.

A cover page is displayed when a channel is selected to provide information related to the channel, such as the channel's name and description.

The scripting in our frameset sets the source of our posting frame as the value of `Autosession.This.URLPostingFrame`. If the currently selected item is a channel, `URLPostingFrame` is set to the value of the channel's Cover Page URL. If the currently selected item is a posting, the source of the page frame is the value of the posting's `URLPostingFrame` property. `URLPostingFrame` returns the URL of the page the posting was created from.



Tip

A channel's Cover Page URL property can be set by an administrator, in the Site Builder. If a channel does not have a Cover Page URL assigned, the `URLPostingFrame` property returns the URL of the default cover page, Cover.asp. Cover.asp is in the /NR/Shared virtual directory. See "Setting a channel's Cover Page URL property" in Chapter 5, "Working within the MSCMS Site Builder" for more information about cover pages.

The frameset file's source code

/NR/Framed Sample/Frameset.asp

```
<!-- #include virtual="/NR/System/Access/Resolution.inc" -->
<!--
   Since this file is in the file system, it is not automatically
   parsed by the server before being served up. Therefore
   we need to manually add a reference to Resolution.inc
-->
<html>

<head>
<title><%= Autosession.This.Name %></title>
</head>

<frameset cols="250,*">

   <!--
      navigation frame: this will always point to the
      navigation template.
   -->
   <frame src="<%= Autosession.This.URLNavigationFrame %>" name="NavigationFrame">
   </frame>

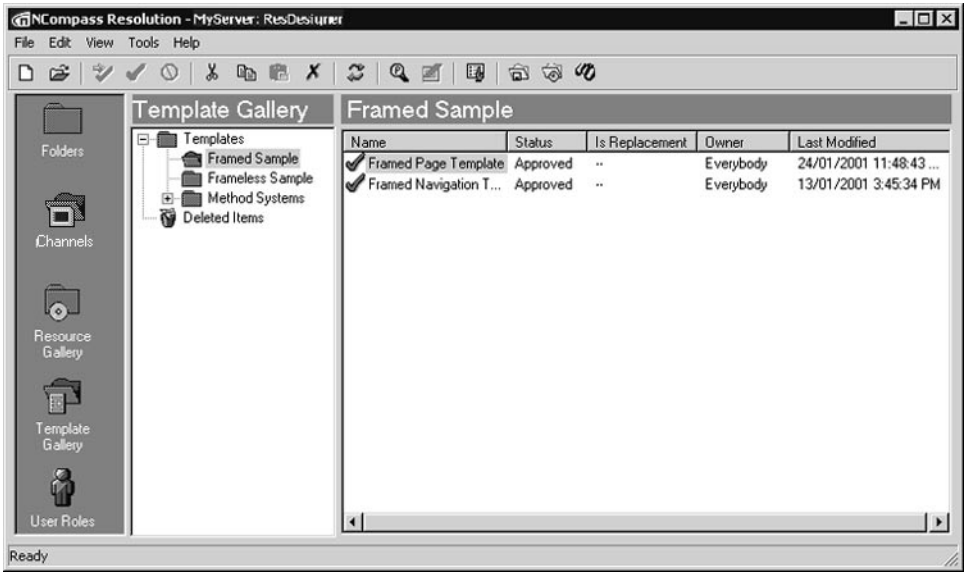
   <!--
      posting frame: if Autosession.This is a channel, then
      URLPostingFrame will point to the default cover page
   -->
   <frame src="<%= Autosession.This.URLPostingFrame%>" name="PostingFrame">
   </frame>
</frameset>
</html>
```

Creating the navigation template

Navigation templates are ASP files stored in the MSCMS 2001 database and assigned to channels. Navigation templates have an optional Frameset URL property that allows the navigation template to be associated with a frameset stored on the Web server. Framed navigation templates are assigned to channels, and they define the frameset used to provide the navigation context for the channel, and postings within the channel. In many framed interfaces, this one included, the navigation template contains code for displaying the channel and posting hierarchy, and is loaded into one of the frameset frames.

Opening the Site Builder

Navigation templates are created and maintained within the Site Builder. To open the Site Builder, select the Site Builder icon on your desktop, or select Site Builder from the Content Management Server Start Menu program group.



Sample Site Builder window

Tip

How can I control which server the Site Builder connects to?

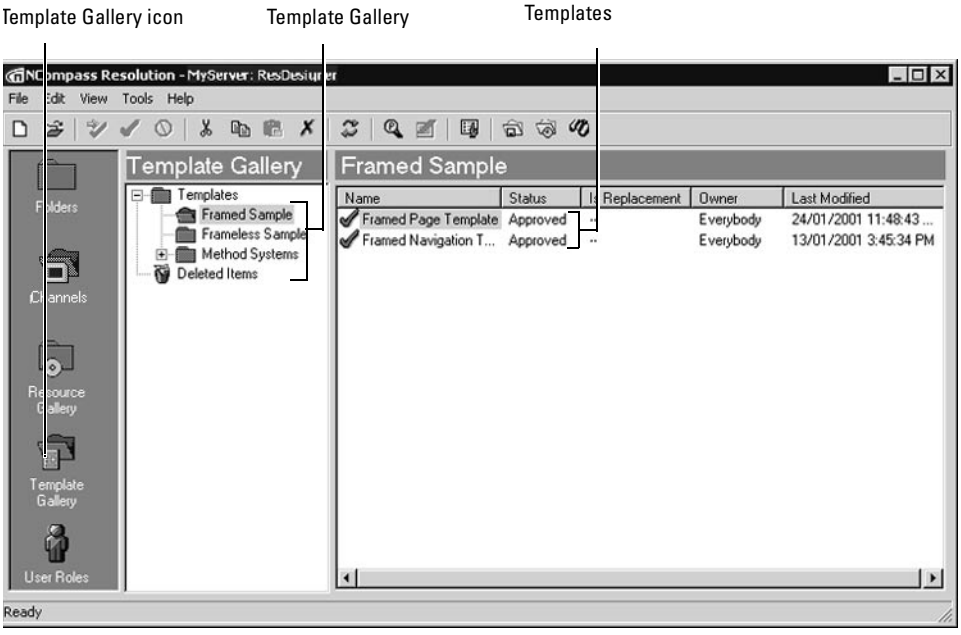
Right-click the shortcut the installation program created on your desktop and select Properties. Select the Shortcut tab. Edit the server name that appears in the URL in the Target box to the name of the server you wish to connect to. Click **OK** to save the changes and close the Properties dialog box. If you frequently connect to different servers, you can make copies of the shortcut and edit them to point to the servers you use.

Creating the Framed Sample template gallery

Note
You must have MSCMS 2001 administrator rights to create a template gallery. If you do not see the Template Gallery icon, you do not have any editing rights for templates. If a gallery is dimmed, you do not have rights to edit templates in that gallery. Ask your MSCMS 2001 administrator to modify your rights accordingly.

Creating Template Galleries, step 1: naming and describing the template gallery

To keep things organized, we'll create a new gallery for our Framed Sample template.

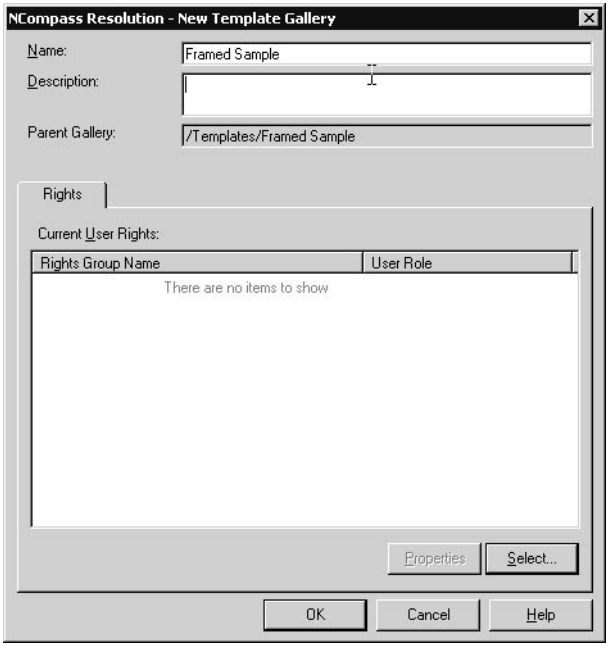


Template Gallery hierarchy expanded in the Site Builder, showing the templates it contains

1 In the Site Builder window, display the template gallery hierarchy by selecting the Template Gallery icon.

2 Right-click the Navigation Templates gallery, and select **New**, then **Gallery**.

3 Right-click a template gallery, and select **New** then **Gallery**. The New Template Gallery dialog box opens.



New Template Gallery dialog box

4 Name the gallery "Framed Sample."

5 Enter a description in the description box to make the template gallery's use easily identifiable to users. This description should briefly say what type of content the template gallery is for, and indicate which rights groups have rights to it. The description can be up to 256 characters.

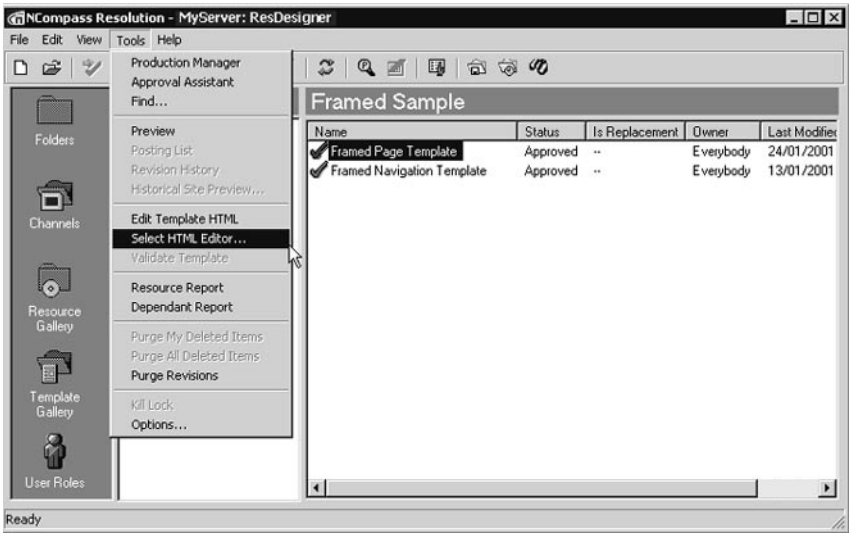
6 If you need to assign rights to the new template gallery to template designers who do not have MSCMS 2001 administrator rights, see Chapter 2, "Setting up Publishing: Assigning Publishing Rights," of the *MSCMS 2001 Site Administrator's Guide*.

Store navigation templates in galleries separate from page template galleries. Name the galleries so that users know which ones contain page templates and which ones contain navigation templates.

Configuring your HTML editor

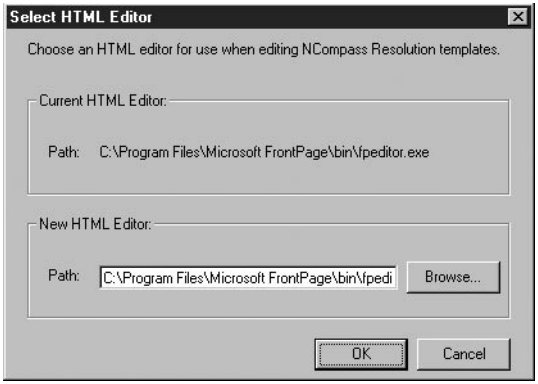
We recommend using an editor designed for working with code, such as Microsoft FrontPage 2000. The only requirement for your HTML editor is that it not modify code.

1 In the Microsoft Content Management Server window, choose the **Select HTML Editor** command from the **Tools** menu. The Select HTML Editor dialog box opens.



2 In the Select HTML Editor dialog box, type in the full path name to your editing application, or use the Browse button to locate it.

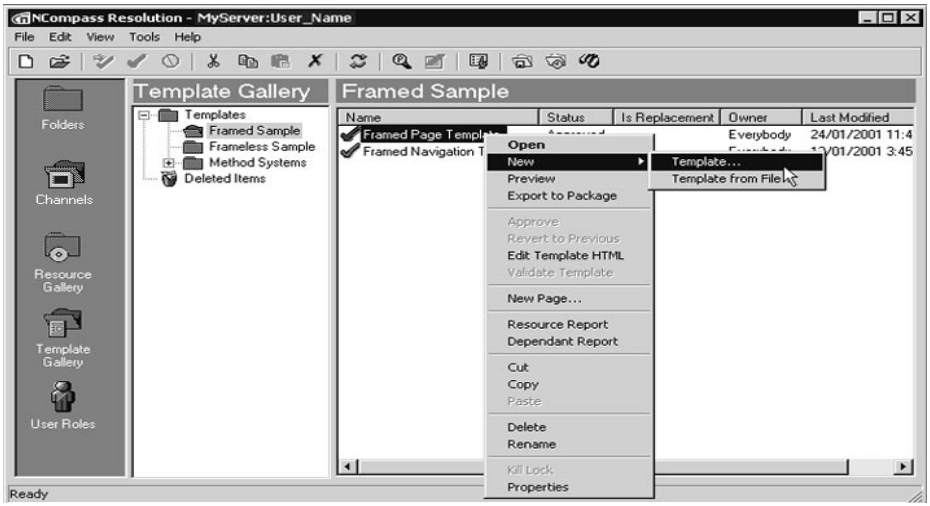
3 When finished, click **OK**.



The HTML editor can be re-configured at any time by repeating this setup procedure.

Creating a new template

1 Right-click the Framed Sample gallery, and select **New**, then **Template**.



Template Gallery hierarchy expanded in the Site Builder, showing the templates it contains

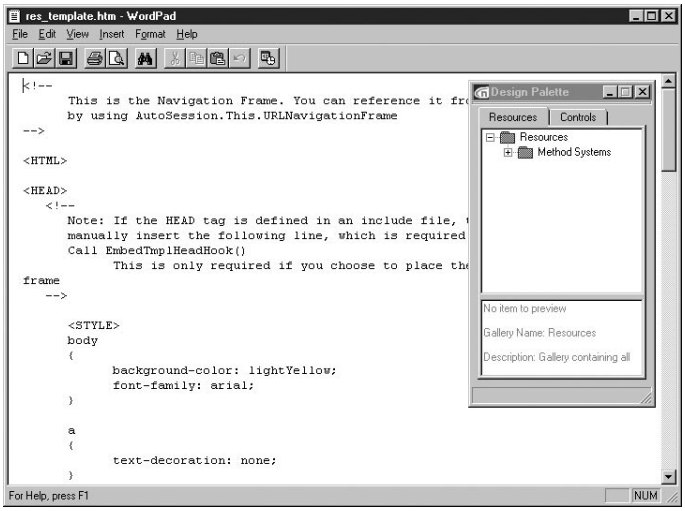
2 In the Select Template Type dialog box, select **Navigation Template**. Click **OK**.

3 The template Design Palette opens on top of a new HTML file in your HTML editor. A new HTML file opens in the editor you configured.

You are prompted to select an editor if you have not configured one yet. See "Configuring your HTML editor" previously in this chapter for information on HTML editors recommended for use with MSCMS 2001.

4 When finished coding your template's functionality, save the HTML file in your HTML editor; the template Design Palette imports and saves the HTML file as an "Untitled" template in the gallery you created it in.

The template Design Palette



Template Design Palette, showing the Resources tab

The template Design Palette opens automatically when you edit a template's HTML or create a new template from scratch. The Design Palette's two principal functions are:

- To manage the joint session between the HTML editor and MSCMS 2001, importing and converting the HTML file when it's saved in the editor, thereby saving it as an MSCMS 2001 template.

Note
The Design Palette must remain open while you are editing a template in your HTML editor. Changes made to a template after the Design Palette is closed will not be saved to the Template Gallery.

- To provide access to the MSCMS 2001 authoring control and resources that can be added to the template in the HTML editor. The Resources tab shows items in the resource gallery that can be added (again, by dragging or copying) to your HTML file.

Note
An authoring control is converted to content placeholders when viewed in the Site Builder's Template Design window, Authoring window, and when viewed in the Web Author's Authoring mode. Page template designers position an authoring control to create page templates. The authoring control is not usually used in navigation templates.

Code the navigation template

The Framed Sample template creates a link to the parent channel, and creates a link to the subchannels and postings of the currently selected channel. It also displays the links for the Web Author. The Web Author links will appear for users who have the appropriate MSCMS 2001 rights.

Templates/Framed Sample/Framed Navigation Template

```
<!--
   This is the Navigation Frame. You can reference it from the frameset
   by using Autosession.This.URLNavigationFrame
-->

<HTML>

<HEAD>
   <!--
   Note: If the HEAD tag is defined in an include file, then you may have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedTmplHeadHook()
      This is only required if you choose to place the Web Author console into this
frame
   -->

   <STYLE>
   body
   {
      background-color: lightYellow;
      font-family: arial;
   }

   a
   {
      text-decoration: none;
   }

   .channel
   {
      font-weight: bold;
   }

   .posting
   {
   }
   </STYLE>
```

```
   <BASE TARGET="_top">

   <TITLE></TITLE>
</HEAD>

<BODY>
<!--
   Note: If the <BODY> tag is defined in an include file, then you will have to
   manually insert the following line, which is required for the Web Author
integration:
   Call EmbedBodyBeginHook()
-->

<%
' Show the button that allows the user to switch to and from
' edit mode.
Call ShowSiteModeSwitch()

' This will embed the Web Author edit console when the user goes to
' edit mode. It will not show up in live mode.
Call EmbedDropDownEditSiteConsole()

Call PrintNavigation() %>

<!--
   Note: If the </BODY> tag is defined in an include file, then you will have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedBodyEndHook()
-->
</BODY>
</HTML>

<%
'=========================================================
' Below are some helper functions used in the template.

'=========================================================
' Prints all of the channels visible from the root.
' Assumes that the user has read rights in the Frameless Sample channel.


Sub PrintNavigation()
   ' In framed navigation templates, Autosession.This always returns the
   ' Channel object. So wee need to figure out by hand if the object we're
   ' linking to is actually a posting.
   Dim pThisObject
```

```
   Set pThisObject = Autosession.ThisPosting
   If pThisObject Is Nothing Then
      Set pThisObject = Autosession.ThisChannel
   End If

   Call PrintChannelNavigation(pThisObject, Autosession.Searches.GetByURL
("/Framed+Sample/"), "")
End Sub

Sub PrintChannelNavigation(pThisObject, pChannel, strIndent)

   If pThisObject.GUID <> pChannel.GUID Then %>
      <div class="channel"><%= strIndent%><a href="<%= pChannel.URL %>">
<%= pChannel.Name %></a></div> <%
   Else %>
      <div class="channel"><%= strIndent%><%= pChannel.Name %></div><%
   End If

   Call PrintChannelPostings(pThisObject, pChannel, strIndent +
"   ")

   Dim pSubchannel
   For Each pSubchannel In pChannel.Channels
      Call PrintChannelNavigation(pThisObject, pSubchannel, strIndent +
"   ")
   Next
End Sub

Sub PrintChannelPostings(pThisObject, pChannel, strIndent)
   Dim pPosting
   For Each pPosting In pChannel.Postings
      If pThisObject.GUID <> pPosting.GUID Then %>
         <div class="posting"><%= strIndent%><a href="<%= pPosting.URL %>"><%=
pPosting.Name %></a></div> <%
      Else %>
         <div class="posting"><%= strIndent%><%= pPosting.Name %></div><%
      End If
   Next
End Sub
%>
```

Saving and naming the template

When you've finished coding your template's functionality, save the HTML file in your HTML editor; the template Design Palette imports and saves the HTML file as an untitled template in the gallery you created it in.

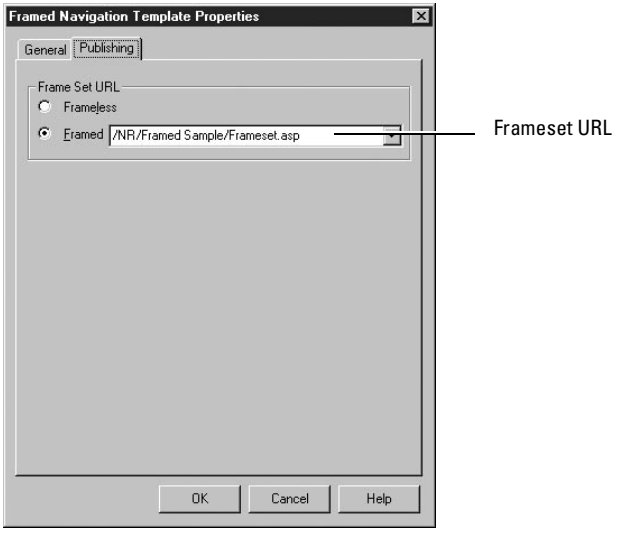
In the Microsoft Content Management Server window, click the new template and enter a descriptive name, such as "Framed Sample Template."

Setting the Frameset URL

To associate the navigation template with its frameset file, we need to set the template's Frameset URL property in the Site Builder. The navigation template can be assigned to only one Frameset URL at a time. We assign the Framed Sample Navigation Template to the frameset file we created earlier, Frameset.asp.

To assign the Frameset URL property:

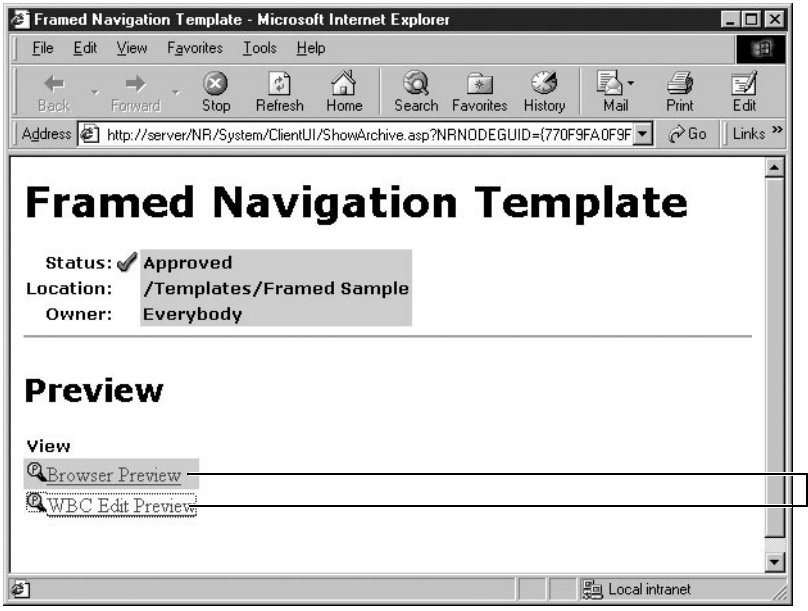
1. Right-click the Framed Navigation Template in the Microsoft Content Management Server window, and select **Properties**.
2. Click the Publishing tab.
3. Select the **Framed** option.
4. In the Framed text box, enter /NR/Framed Sample/Frameset.asp. This is the virtual directory path to the frameset file we created. See Chapter 1, "Elements of an MSCMS 2001 Web Interface" for more information on how MSCMS 2001 virtual and physical directories are mapped.



Previewing the template

Before submitting the template, making it available for use by channels, preview it in a Web browser.

1 Right-click the Framed Navigation Template in the Microsoft Content Management Server window, and select **Preview**. The Preview window opens.



Preview window, opened from the Site Builder

2 You can view the template as it would appear in a browser to a user with MSCMS 2001 subscription rights only, or as it would appear to a user with MSCMS 2001 publishing rights. The template appears in your Web browser.

Note
The template does not appear within a frameset.

Where do the channels and postings I see in Preview mode come from?

When using the Preview command with templates, a predefined hierarchy of simulated channels and postings is generated to provide return values for calls to Autosession and other Publishing API objects.

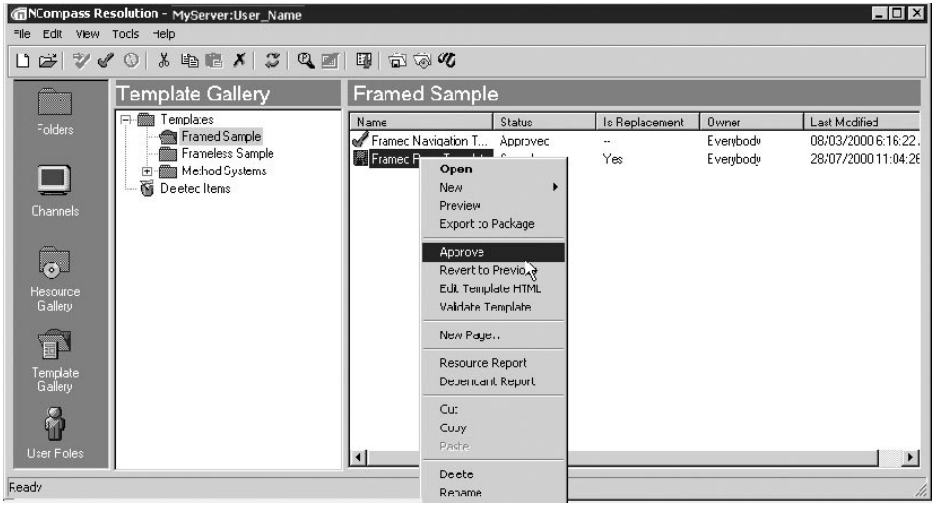
See “About simulated channels” in Chapter 4. “Using the MSCMS 2001 Publishing API” for more information.

Approving the template

Once a template is approved, its status changes from Saved to Approved, and it is available for use immediately.

To approve the template:

1 Right-click the template in the Microsoft Content Management Server window, and select **Approve**.



2 The template's status changes from Saved to Approved.

Tip
You cannot approve a template while the template Design Palette is open.

About template revisions

Every time a template is approved a revision is created. Template revisions appear as revisions of a posting's Revision History report.

Assigning the template to a channel

In a framed site, a navigation template is assigned to a channel to determine the frameset displayed when a channel's URL is requested. We want the Framed Sample channel to display the Framed Navigation Template, so we must set the channel's Navigation Template property.

To assign the template to the channel:

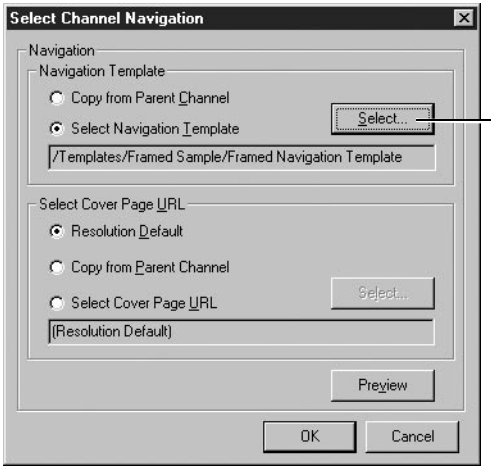
1 In the Microsoft Content Management Server window, display the channels hierarchy by selecting the Channels icon.

2 Right-click the Sample Framed channel, and select **Properties**.



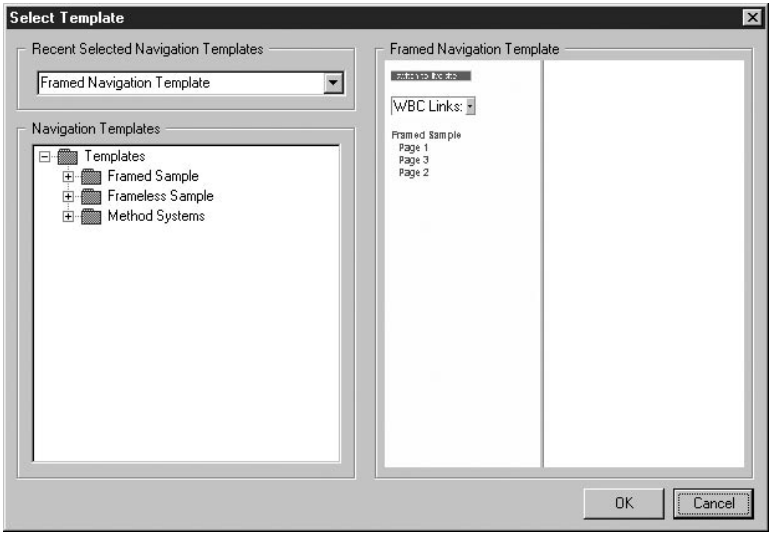
3 Select the Publishing tab, and click **Select**. The Select Channel Navigation dialog box opens.

4 Click **Select** to browse for a navigation template. The Select Template window opens.



Click to browse for a navigation template

5 Select a template and click **OK**.



If the channel has not been enabled for the Web Author, see Chapter 2, "Setting up Publishing: Creating Containers for Users" of the *MSCMS 2001 Site Administrator's Guide* for instructions.

Viewing the channel in a Web browser

The path in MSCMS 2001 to the channel we want to view is /Channels/Framed Sample, so the URL for the channel is `http://yourserver/Framed+Sample/`. This is derived from the scheme in which MSCMS 2001 channels are translated to URLs:

- The root channel, called by default "Channels", is not part of the URL, except when you need to address the root channel (`http://yourserver/Channels/`).
- You must enter a " + " in place of a space.

Step-by-step through the run-time execution

Following is a summary of how the Framed Sample interface is constructed at run time, from the point of view of first a channel, and then a posting object.

Channel

1 A browser links to the Tutorial channel's URL (`http://yourserver/Framed+Sample/`).

2 The server determines that the URL corresponds to an MSCMS 2001 channel.

3 MSCMS 2001 checks the channel's navigation template, and determines that it's framed.

4 MSCMS 2001 instructs IIS to redirect the browser to the navigation template's Frameset URL.

5 Code in Resolution.inc (included in the frameset file) calls Authenticate.inc, which initiates the authentication process. (Refer to the section on authentication in Chapter 1, "Elements of an MSCMS 2001 Web Interface" for a description of this process.)

6 Code in the frameset executes. It sets the name string is displayed in the browser title bar, and defines the source for the posting frame.

Since the currently selected item is a channel, the posting frame source is set to the value of the Publishing API property `URLPostingFrame`. We didn't set the channel's Cover Page URL property, so returned value is /NR/Shared/Cover.asp, the default cover page.

7 The channel frame is defined. The channel frame uses the selected channel object's `URLNavigationFrame` property as its source. For `Channel` objects, `URLNavigationFrame` returns the URL for the channel's navigation template.

8 The navigation template's ASP code executes, defining links to the selected channel's parent, sub-channels, postings, and Web Author links.

9 The posting frame is defined. The source for the posting frame was defined previously in the frameset code. In this case, being that a channel is selected, the source for the frame is defined as the Publishing API `Channel` object property `URLPostingFrame`. Since there is no Cover Page URL defined for the channel, the frame loads the default cover page, /NR/Shared/Cover.asp.

Posting

1 A Web browser links to a posting within the Framed Sample channel, either by selecting it from the channel and postings hierarchy displayed for the channel, or by entering its URL directly, or by some other link.

2 The server determines that the URL corresponds to an MSCMS 2001 posting.

3 MSCMS 2001 checks the posting's channel's navigation template, and determines that it's framed.

4 MSCMS 2001 instructs IIS to redirect the browser to the navigation template's Frameset URL.

5 Code in Resolution.inc (included in the frameset file) initiates the authentication process. (Refer to "Authentication" in Chapter 1, "Elements of an MSCMS 2001 Web Interface" for a description of this process.)

6 Code in the frameset executes. It sets the name string that should be displayed in the browser title bar, and defines the source for the posting frame.

Since the currently selected item is a posting, the posting frame source is set to the value of the Publishing API property `URLPostingFrame`. When the selected object is a posting, `URLPostingFrame` returns the URL for displaying the content of the posting.

7 The first frame to be defined is the Channel frame. The Channel frame uses the selected posting object's `URLNavigationFrame` property as its source. For `Posting` objects, `URLNavigationFrame` returns the URL for the parent channel's navigation template, and additionally specifying that `This.Posting` is the selected posting.



Note

Using the `Channel.URL.NavFrame` or deprecated `Channel.FramelessURL` does not populate the value of `This.Posting`. Instead, use the `URLNavigationFrame` property to enable the navigation template to be aware of the current posting and provide Web Author links for `This.Posting`.

8 The navigation template's ASP code executes, defining links to the selected channel's parent, sub-channels, and postings.

9 Finally, the posting frame is defined. The source for the posting frame was defined previously in the frameset code. In this case, being that a posting is selected, the source for the frame is defined as the Publishing API `Posting` object property `URLPostingFrame`. The `URLPostingFrame` of a posting returns the URL of the posting content without a frameset context.

10 The server assembles the page template and content for the posting, and displays it in the posting frame.

Chapter 3

Creating a Frameless MSCMS 2001 Interface

Introduction

To help you get started creating a frameless interface for an MSCMS 2001 channel, in this chapter we step through the process you would follow to develop a sample Frameless Navigation interface.

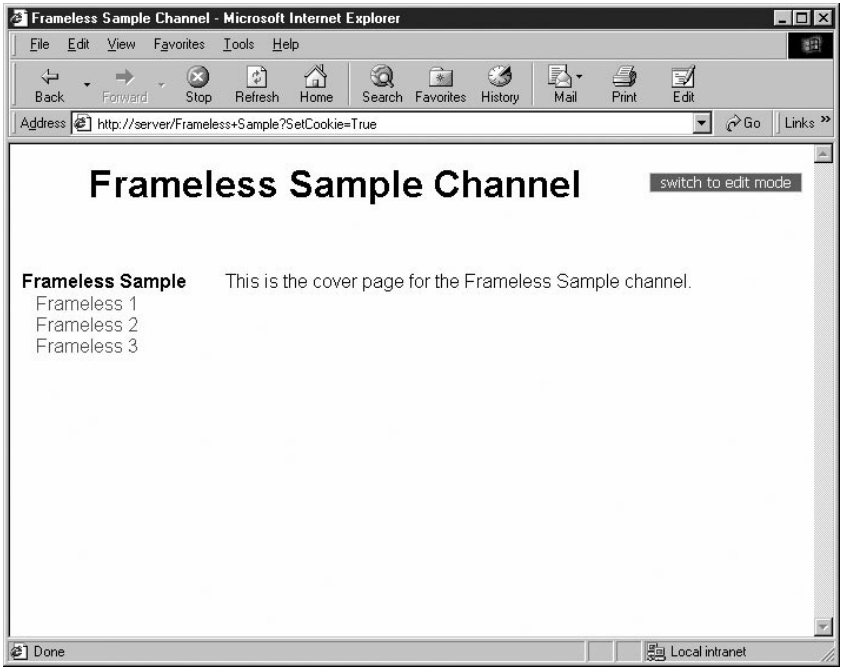
Note
Any references to the Method Systems sample site or the Frameless Sample interface are intended for illustration only.

Viewing the frameless navigation interface in your browser

Open the URL for the Frameless Sample channel to be viewed. For example:

```
http://<your-server>/Frameless+Sample/
```

After confirming that you have the necessary rights to view the Frameless Sample channel, MSCMS 2001 redirects you to its navigation template's URL.



The sample frameless navigation template

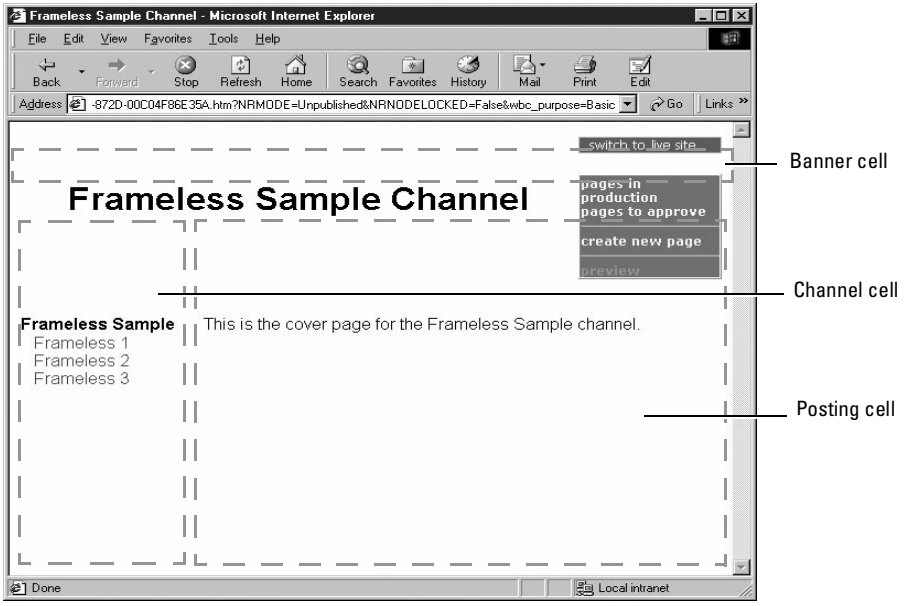
Frameless Navigation Tutorial interface overview

Frameless sites display only one ASP file at a time. This ASP file is either a page template or a navigation template. This means that although we could design the interface to be nearly identical to the Framed Sample Navigation interface, the components and their functions will be different.

The requirements for our interface are:

- to provide the means for navigating in the channel and posting hierarchy
- to display the selected channel's cover page, or display the selected posting's page.

Based on those requirements, we split the navigation template into two table cells: Channel and Page. The Channel cell loads an ASP include file that displays the channel and posting hierarchy. In the navigation template, we'll code the posting cell to display a title and a graphic. We'll also create a page template outline for template designers to use when creating frameless page templates.



Important information about frameless interfaces

Implementation of frameless interfaces in MSCMS 2001 is different from that of framed interfaces.

1. Navigation templates are only displayed when channels are selected. Navigation templates and page templates are both ASP files. A Web browser cannot display two ASP files without the use of a frameset. A navigation template is displayed when a channel is selected; when a posting is selected, the page's template is displayed.
2. Navigation templates can redirect to the channel's Cover Page URL. If you use the same navigation template for several channels, but still want to customize the content users see when they select the channel's link, a channel's navigation template can be programmed to redirect users to the channel's Cover Page URL. Each channel on your site can have a different Cover Page URL value. The Cover Page URL property can be set to an introductory posting.
3. Frameless page templates must contain navigation elements. Unlike framed interfaces, where pages are displayed within a framed navigation context, a frameless page template is a stand-alone ASP file. Therefore, it should contain the same navigation links as the Frameless Sample Navigation Template.

Development process summary

1. Create the navigation template.The navigation template is an ASP file stored in the MSCMS 2001 database. It provides the navigation context for the Frameless Sample channel.
2. Preview the navigation template.
3. Submit the navigation template.
4. Assign the navigation template to the Frameless Sample channel.
5. View the channel in a Web browser.
6. Create a page template outline.
7. Frameless page templates require the same navigation elements as a navigation template. We create an outline page template to make it easier to create page templates.

Resolution.inc

Resolution.inc references Authenticate.inc, which confirms that subscribers have rights to requested channels and postings.

For further information, including tips for initializing the Autosession object, see "Resolution.inc" in Chapter 2, "Creating a Framed MSCMS 2001 Interface"

Creating the navigation template

A navigation template is an ASP file stored in MSCMS 2001. Navigation templates are assigned to channels. When the channel's URL is requested, if the channel has a frameless navigation template assigned, the URL returned is that of the navigation template. The navigation template we'll create for our frameless channel displays the channel hierarchy, and a link to the Web Author.

Opening the Site Builder

Navigation templates are created and maintained within the Site Builder. To open the Site Builder, select the Site Builder icon on your desktop, or select Site Builder from the Content Management Server Start Menu program group.

Tip

How can I control which server the Site Builder connects to?

Right-click the shortcut the installation program created on your desktop and select Properties. Select the Shortcut tab. Edit the server name that appears in the URL in the Target box to the name of the server to connect to. Click **OK** to save the changes and close the Properties dialog box.

If you frequently connect to different servers, you can make copies of the shortcut and edit them to point to the servers you use.

Creating the Frameless Sample template gallery

To keep things organized, we'll need a new gallery for our frameless navigation template. Only MSCMS 2001 administrators can create containers. If you have MSCMS 2001 administrator rights, see the *MSCMS 2001 Site Administrator's Guide* for instructions on creating a template gallery.

Note

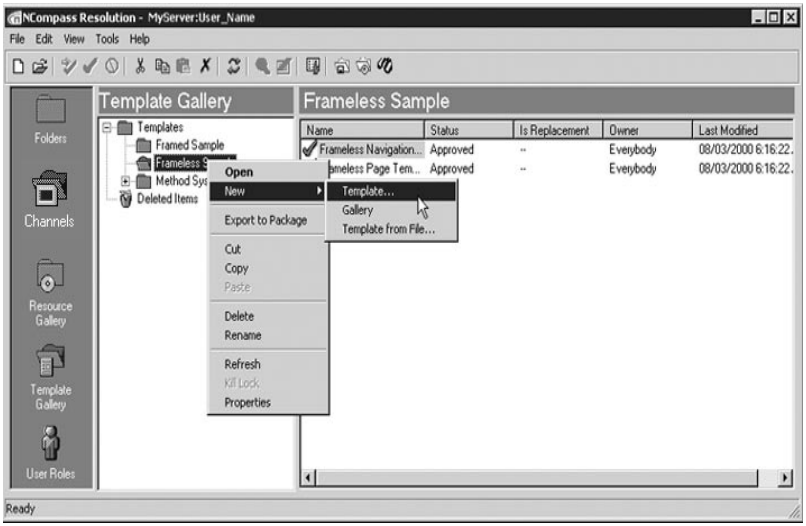
If you do not see the Template Gallery icon, you do not have any template editing rights. If the navigation templates gallery is dimmed, you do not have rights to create a new gallery. Ask your system administrator to modify your rights accordingly.

Configuring your HTML editor

See "Configuring your HTML editor" in Chapter, 2 "Creating a Framed MSCMS 2001 Interface" for more information.

Creating a new template

1 Right-click the Frameless Sample gallery, and select **New**, then **Template**.



Template Gallery hierarchy expanded in the Site Builder, showing the templates it contains

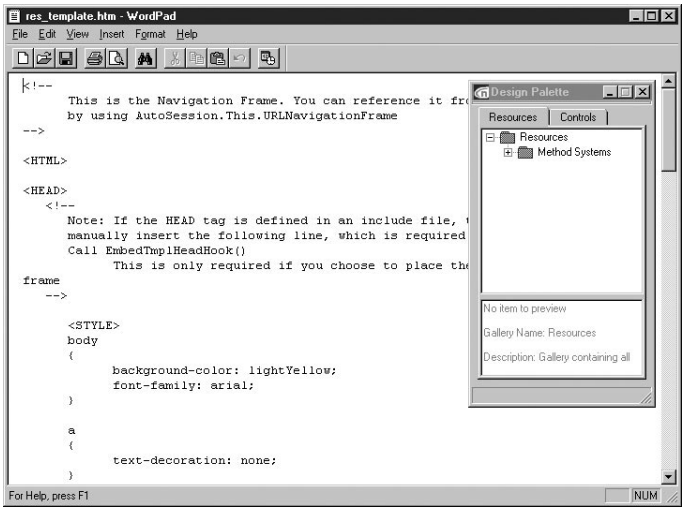
2 A new HTML file opens in the editor you configured.

Note

You are prompted to select an editor if you have not configured one yet. See "Configuring your HTML editor" in Chapter, 2 "Creating a Framed MSCMS 2001 Interface" for information on HTML editors recommended for use with MSCMS 2001.

3 When finished coding your template's functionality, save the HTML file in your HTML editor; the template Design Palette imports and saves the HTML file as an "Untitled" template in the gallery you created it in.

The template Design Palette



Template Design Palette, showing the Resources tab

The Template Design Palette opens automatically when you edit a template's HTML or create a new template from scratch. The Design Palette's two principal functions are:

- To manage the joint session between the HTML editor and MSCMS 2001, importing and converting the HTML file when it's saved in the editor, thereby saving it as a template.

Note
The Design Palette must remain open while you are editing a template in your HTML editor. Changes made to a template after the Design Palette is closed will not be saved to the template gallery.

- To provide access to MSCMS 2001 authoring control and resources that can be added to the template in the HTML editor. The Resources tab shows items in the resource gallery that can be added (again, by dragging or copying) to your HTML file.

Note
An authoring control is converted into content placeholders when viewed in the Site Builder's template Design window, Authoring window, and when viewed in the Web Author's Authoring mode. Page template designers position authoring controls to create page templates. An authoring control is not usually used in navigation templates.

Coding the frameless navigation template

```
<HTML>

<HEAD>
   <!--
   Note: If the HEAD tag is defined in an include file, then you will have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedTmplHeadHook()
   -->

   <STYLE>
   body
   {
      background-color: lightYellow;
      font-family: arial;
   }

   a
   {
      text-decoration: none;
   }

   .channel
   {
      font-weight: bold;
   }

   .posting
   {
   }
   </STYLE>

   <TITLE><%= Autosession.This.Name %> Channel</TITLE>
</HEAD>
<BODY>
<!--
   Note: If the <BODY> tag is defined in an include file, then you will have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedBodyBeginHook()
-->

<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
<td align="center" width="100%">
<h1><%= Autosession.This.Name %> Channel</h1>
```

```
</td>
<td> <%

   ' Show the button that allows the user to switch to and from
   ' edit mode.
   Call ShowSiteModeSwitch() %>

   <br> <%

   ' This will embed the Web Author edit console when the user goes to
   ' edit mode. It will not show up in live mode.
   Call EmbedSimpleEditSiteConsole() %>
</td>
</tr>
</table>

<br><br>

<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
<td vAlign="top"> <%
   ' Show the channel & posting hierarchy
   Call PrintNavigation() %>
</td>

<td vAlign="top">
   This is the cover page for the <%= Autosession.ThisChannel.Name %> channel.
</td>
</tr>
</table>

<!--
   Note: If the </BODY> tag is defined in an include file, then you will have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedBodyEndHook()
-->
</BODY>
</HTML>

<%
'============================================================
' Below are some helper functions used in the template.

'============================================================
' Prints all of the channels visible from the root.
```

```
' Assumes that the user has read rights in the Frameless Sample channel.

Sub PrintNavigation()
   Call PrintChannelNavigation(Autosession.Searches.GetByURL("/Frameless+Sample/"),
"")
End Sub

Sub PrintChannelNavigation(pChannel, strIndent)

   If Autosession.This.GUID <> pChannel.GUID Then %>
      <div class="channel"><%= strIndent%><a href="<%= pChannel.URL %>"><%=
pChannel.Name %></a></div> <%
   Else %>
      <div class="channel"><%= strIndent%><%= pChannel.Name %></div><%
   End If

   Call PrintChannelPostings(pChannel, strIndent + "   ")

   Dim pSubchannel
   For Each pSubchannel In pChannel.Channels
      Call PrintChannelNavigation(pSubchannel, strIndent + "   ")
   Next
End Sub
Sub PrintChannelPostings(pChannel, strIndent)
   Dim pPosting
   For Each pPosting In pChannel.Postings
      If Autosession.This.GUID <> pPosting.GUID Then %>
         <div class="posting"><%= strIndent%><a href="<%= pPosting.URL %>"><%=
pPosting.Name %></a></div> <%
      Else %>
         <div class="posting"><%= strIndent%><%= pPosting.Name %></div><%
      End If
   Next
End Sub
%>
```

Saving and naming the template

When finished coding your template's functionality, save the HTML file in your HTML editor; the template Design Palette imports and saves the HTML file as an untitled template in the gallery you created it in.

In the Microsoft Content Management Server window, click the new template and enter a descriptive name, such as "Frameless Navigation Template."

Previewing the template

You can view the template as it would appear in a browser to a user with subscription rights only, or as it would appear to a user with publishing rights.

See "Previewing the template" in Chapter 2, "Creating a Framed MSCMS 2001 Interface" for more information.

Approving the template

Once a template is approved, its status changes from Saved to Approved, and can be used immediately.

See "Approving the template" in Chapter 2, "Creating a Framed MSCMS 2001 Interface" for more information.

Assigning the template to a channel

In a frameless site, a navigation template is returned when a channel's URL is requested. The navigation template defines the channel of the Web interface.

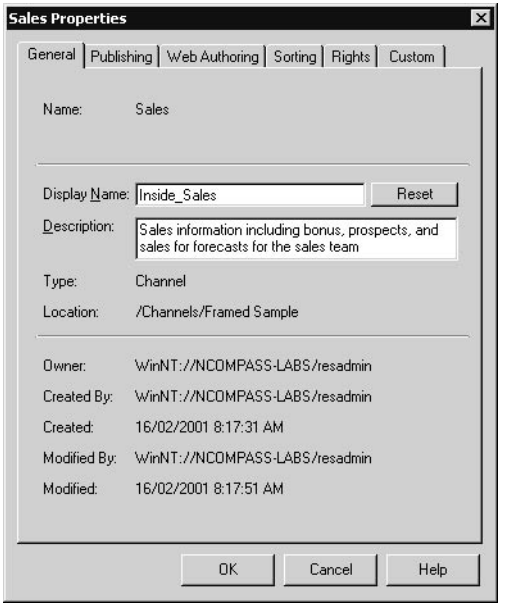
We want the Frameless channel to display the Frameless Sample Navigation template, so we must set the channel's Navigation Template property.



Note
Only MSCMS 2001 administrators can modify channel properties.

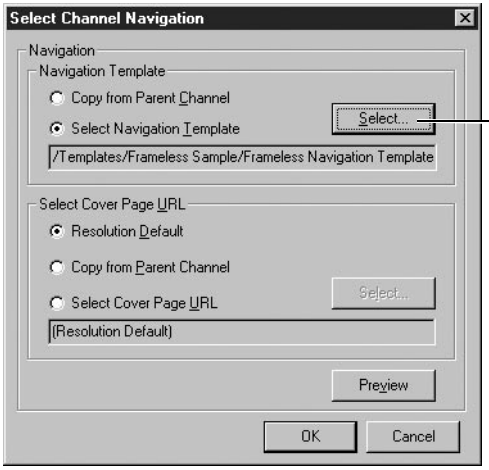
To assign the template to the channel:

1. In the Content Management Server window, display the channels hierarchy by selecting the Channels icon.
2. Right-click the Sample Frameless channel, and select **Properties**.



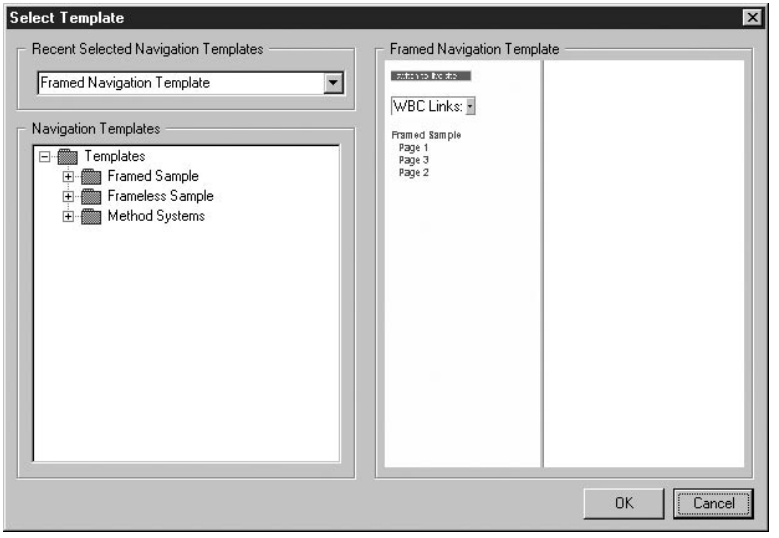
3 Select the Publishing tab, and click **Select**. The Select Channel Navigation dialog box opens.

4 Click **Select** to browse for a navigation template.



Click to browse for a navigation template

5 The Select Template window opens.



6 Select a template and click **OK**.

If the channel has not been enabled for the Web Author, see Chapter 2, "Setting up Publishing: Creating Containers for Users" of the *MSCMS 2001 Site Administrator's Guide* for instructions.

Viewing the channel in a Web browser

The path to the channel we need to view is /Channels/Frameless Sample/, so the URL for the channel is `http://yourserver/Channels/Frameless+Sample/`.

This is derived from the scheme in which MSCMS 2001 channels are translated to URLs:

- The root channel, called by default “Channels”, is not part of the URL, except to address the root channel (`http://yourserver/Channels/`).
- You must enter a “ + ” in place of a space.

Creating the page template outline

Because the navigation components of a frameless site are incorporated within page templates, navigation programmers and page template designers must work together.

We recommend that navigation programmers create an “outline” template for page template designers to modify further. The outline template is a frameless HTML page with one or more HTML tables with cells containing ASP scripting (or server-side includes pointing to the external ASP include files). The outline template is used by page template designers, who add placeholders and additional HTML to the table cell intended for the posting content.

The following example page template outline uses an HTML table for the navigation and content. Comments are used to indicate what areas the page template designers should edit.

To create a template outline, follow this example:

Page template outline example

```
<HTML>

<HEAD>
   <!--
   Note: If the HEAD tag is defined in an include file, then you will have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedTmplHeadHook()
   -->

<!--  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
#
# Frameless Page Outline
# This file contains references to navigation elements which
# can be included in Page Templates. Page Template designers
# can make a copy of this Template to use as a basis for their Templates.
#
```

```
# See the Template "Frameless Page Template" for an example of how a
# Page Template designer would modify a copy of this template to
# prepare it for use by authors.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->


   <STYLE>
   body
   {
      background-color: lightYellow;
      font-family: arial;
   }

   a
   {
      text-decoration: none;
   }

   .channel
   {
      font-weight: bold;
   }

   .posting
   {
   }
   </STYLE>

   <TITLE><%= Autosession.This.Name %> Channel</TITLE>
</HEAD>

<BODY>
<!--
   Note: If the <BODY> tag is defined in an include file, then you will have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedBodyBeginHook()
-->

<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
<td align="center" width="100%">
<h1><%= Autosession.This.Name %> Channel</h1>
</td>
<td> <%
```

```
   ' Show the button that allows the user to switch to and from
   ' edit mode.
   Call ShowSiteModeSwitch() %>

   <br> <%

   ' This will embed the Web Author edit console when the user goes to
   ' edit mode. It will not show up in live mode.
   Call EmbedSimpleEditSiteConsole() %>
</td>
</tr>
</table>

<br><br>

<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr>
<td vAlign="top"> <%
   ' Show the channel & posting hierarchy
   Call PrintNavigation() %>
</td>

<!--  ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# This table is equivalent to the framed site's content
# (page) frame and navigation frame. In this frameless template,
# navigation is handled by an include file containing scripting
# similar to that in a framed navigation template.
# Site programmers should set up a template similar to this
# one for Page Template designers to use. Page Template
# Designers should only need to add HTML and placeholders within
# the one table cell reserved for content.
# ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ -->

<td vAlign="top">
   This is the cover page for the <%= Autosession.ThisChannel.Name %> channel.
</td>
</tr>
</table>

<TABLE>
<TR>
   <TD valign="top" WIDTH=500>

     <!-- This is the start of the cell intended for use by page template designers
-->
```

```
      <H2>Frameless Page Template</H2>
      Page template designers add HTML and placeholders to this table cell.

      <!-- Page template designers should not edit anything below this comment -->
   </TD>
   <TD valign="top">

<!--
   Note: If the </BODY> tag is defined in an include file, then you will have to
   manually insert the following line, which is required for Web Author integration:
   Call EmbedBodyEndHook()
-->
</BODY>
</HTML>

<%
'============================================================
' Below are some helper functions used in the template.

'============================================================
' Prints all of the channels visible from the root.
' Assumes that the user has read rights in the Frameless Sample channel.

Sub PrintNavigation()
   Call PrintChannelNavigation(Autosession.Searches.GetByURL("/Frameless+Sample/"),
"")
End Sub

Sub PrintChannelNavigation(pChannel, strIndent)

   If Autosession.This.GUID <> pChannel.GUID Then %>
      <div class="channel"><%= strIndent%><a href="<%= pChannel.URL %>"><%=
pChannel.Name %></a></div> <%
   Else %>
      <div class="channel"><%= strIndent%><%= pChannel.Name %></div><%
   End If

   Call PrintChannelPostings(pChannel, strIndent + "   ")

   Dim pSubchannel
   For Each pSubchannel In pChannel.Channels
      Call PrintChannelNavigation(pSubchannel, strIndent + "   ")
   Next
End Sub

Sub PrintChannelPostings(pChannel, strIndent)
```

```
   Dim pPosting
   For Each pPosting In pChannel.Postings
      If Autosession.This.GUID <> pPosting.GUID Then %>
         <div class="posting"><%= strIndent%><a href="<%= pPosting.URL %>"><%=
pPosting.Name %></a></div> <%
      Else %>
         <div class="posting"><%= strIndent%><%= pPosting.Name %></div><%
      End If
   Next
End Sub
%>
```

Save this HTML page as an HTML file on your system so you can import it as a template later and modify it to suit your needs.

Step-by-step through the run-time execution

The following is a summary of how the Sample Frameless interface is constructed at run time, from the point of view of first a channel, and then a posting object.

Channel

1. A Web browser links to the Frameless channel's URL (`http://yourserver/Frameless+Sample/`).
2. The server determines that the URL corresponds to an MSCMS 2001 channel.
3. MSCMS 2001 checks the channel's navigation template, and determines that it's frameless.
4. MSCMS 2001 instructs IIS to redirect the browser to the navigation template's URL.
5. Code in Resolution.inc (included automatically in the navigation template) calls Authenticate.inc, which initiates the authentication process. (Refer to the section on authentication in Chapter 1, "Elements of an MSCMS 2001 Web Interface" for a description of this process.)
6. A table cell displays links to the channel's postings.
7. Another table cell displays information about the currently selected channel (such as a cover page or posting).

Posting

1. A Web browser links to a posting within the Frameless channel, either by selecting it from a channel and postings hierarchy, or by entering its URL directly.
2. The MSCMS 2001 server determines that the URL corresponds to an MSCMS 2001 posting.
3. MSCMS 2001 checks the posting's channel's navigation template, and determines that it is frameless.
4. MSCMS 2001 instructs IIS to redirect the browser to the posting's page template.
5. Code in Resolution.inc (automatically included in Frameless page templates at run time) calls Authenticate.inc, which initiates the authentication process. (Refer to the section on authentication in Chapter 1. "Elements of an MSCMS 2001 Web Interface: for a description of this process.)
6. The page template's ASP code begins to run. The first table cell to be defined is the banner cell, in this case used to call `ShowSiteModeSwitch()`, which presents the link to switch to the Web Author.
7. The next table cell that includes ChannelHierarchy.inc is defined. ChannelHierarchy.inc contains ASP code that uses the Publishing API to define links to the selected posting's channel's parent channel, sub-channels, and postings.
8. Finally, a table cell includes the content placeholders, which are filled with the content the author added to the page.

Extending your interface

For more advanced examples of scripting the navigation for frameless MSCMS 2001 sites, see Chapter 4, "Using the MSCMS 2001 Publishing API." Chapter 9, "Customizing Your MSCMS 2001 Web Author" also provides working examples of integrating advanced uses of the Publishing API and client-side JavaScript with your navigation templates and server files.

Chapter 4

Using the MSCMS 2001 Publishing API

Introduction

This section is intended to give you ideas for customizing and extending your MSCMS 2001 Web interface using the MSCMS 2001 Publishing API. For a complete guide to creating a basic framed interface, see Chapter 2, "Creating a Framed MSCMS 2001 Interface"; for frameless sites, see Chapter 3, "Creating a Frameless MSCMS 2001 Interface." For information on all the Publishing Application Programming Interface (API) objects, methods, and properties, see the *MSCMS 2001 Publishing API Help*.

Remember that all ASP files external to the MSCMS 2001 that reference the Publishing API must include Resolution.inc at the top:

```
<!--#include virtual="/NR/System/Access/Resolution.inc"-->
```

About MSCMS 2001 URLs

The URLs you see in the browser, either hierarchy-based or unique, ID-based, are the ones returned from calls to the Publishing API. These are the URLs you would bookmark or pass to users to access postings or channels. When a browser requests one of these URLs, the server translates it to an internal URL form before running the ASP script. The internal URL form often contains a query string with parameters describing which channel or posting you're viewing, in what mode, and other information. The Channel and Posting `QueryString` property can be appended to the URL of an external ASP file in order to run this ASP file as the selected object. For example:

```
strURL = "http://YourServer/NR/Shared/YourFile.ASP?"- & pPosting.QueryString
```

Calling `Autosession.This` in the external ASP file will return the posting or channel that generated the query string.

The parameters in the query string are subject to change with subsequent releases of MSCMS 2001, and should not be accessed directly.

The Web Author uses specific query string properties to determine contextual information. For more information, see Appendix A.

Ensuring proper display of templates

All templates must include the following tag between the <HEAD> and </HEAD> tags:

```
<BASE target="top">
```

This tag ensures that framed navigation templates are displayed within their assigned framesets, and that frameless navigation templates are not displayed within a frameset.

About simulated channels

When using the Preview command with navigation or page templates, a predefined hierarchy of simulated channels and postings is generated to provide return values for calls to Autosession, and other Publishing API objects. Once this hierarchy is generated, its simulated channels and postings respond to all Channel and Posting methods just as normal channels or postings do. In Preview mode, any call to `Autosession.ThisChannel`, `Autosession.RootChannel`, or `Autosession.This` can generate the hierarchy of simulated channels.

Only the Channels, News & Events, and Press Releases (Archived) simulated channels have any subchannels, and only Press Releases (Archived) has any postings. All other simulated channels return empty collections for these objects.

For navigation templates, the preview is centered on the navigation template and its parent. Calling `Autosession.ThisChannel` or `Autosession.This` returns that navigation template simulating the properties of a Channel object. `Autosession.RootChannel` calls always return the simulated root channel (Channels) for any template preview, or when the template is being edited.

The placeholder collections return those of the template being previewed for `This`, the currently selected object. Any other simulated channel or posting in the hierarchy returns an empty placeholder or resource collection. Any properties or methods that return URLs return a URL with a `JavaScript(void)null` so that the target of such a link is the current URL. All other interface methods called on the simulated objects return reasonable default values.

These predefined simulated channels and postings should be sufficient for a reasonable preview of the look, feel, and functionality of new templates. There are newer and older postings and channels; some are marked important, others are not. Also, the sorting functions can produce different orders of the postings or channels. This functionality can be used to fine-tune templates before they are available for general use.

Dynamically generating a "What's New" page

The Searches object's NewPostings and NewChannels methods return a collection of channels or postings created or changed within a specified number of days.

Displaying New Items: /NR/Shared/Whatsnew.asp

The following example uses the NewChannels and NewPostings methods to return those channels and postings that have been created or changed in the last seven days. The syntax for NewPostings is:

```
pPostings As Postings = NewPostings([lNumberOfDays As Long])
```

NewChannels uses a similar syntax. For details about these methods, see the *MSCMS 2001 Publishing API Help.*

To make managing the search interface easier, the g_SearchInterface variable is used to store the results of a call to Autosession.Searches before calling the NewChannels and NewPostings method. Alternatively, navigation programmers could use Autosession.Searches.Channels(7):

```
Dim g_Searches
Dim g_Channels
Dim g_Postings

Set g_Searches = Autosession.Searches

Set g_Channels = g_Searches.NewChannels(7)

Set g_Postings = g_Searches.NewPostings(7)
```

Dynamically generating a "What's Important" page

The Searches object's ImportantChannels and NewPostings method allows subscribers to view all postings and channels, appropriate to the subscriber's rights, that have been flagged (by their creators) as important.

Displaying Important Items: Important.asp

The syntax for ImportantChannels is:

```
pChannels As Channels = Searches.ImportantChannels
```

ImportantPostings uses a similar syntax. For more details about these methods, see the *MSCMS 2001 Publishing API Help*.

ImportantChannels and ImportantPostings return all items that are appropriate to the subscriber's rights and have been classified as important items. They will return both subscribed and unsubscribed items.

The following example uses ImportantChannels and ImportantPostings to return items marked as important.

```
Dim g_Searches
Dim g_Channels
Dim g_Postings

Set g_Searches = Autosession.Searches
Set g_Channels = _ g_Searches.ImportantChannels
Set g_Postings = _ g_Searches.ImportantPostings
```

Preventing unauthorized client-side script from executing

Generally template designers and administrators are the only users who are permitted to add script content into a posting or channel. However, the potential exists for a user with malicious intent to add client-side script. For example, an author could set the Description value of a posting to `<script language=Javascript> location.href = "http://www.othersite.com"; </script>`. If the posting has script to display the Description property, then when a user browsing the site navigates to this posting the client-side JavaScript would automatically be injected into the posting and the page would be rerouted to `www.othersite.com`.

To prevent this unexpected execution of script within channels and postings we recommend that the ASP Server.HTMLEncode method be used to encode MSCMS 2001 Publishing API text properties before displaying them. This will allow values to be displayed on a Web page, but will prevent malicious script from being executed. Any text property that can be altered by a user other than a template designer or administrator should be encoded. For example:

Channel: Description, DisplayName, Name, Path
Posting: Description, DisplayName, Name, Path
Resource: Description, DisplayName, Name, Path
CustomProperty: Value

Example

This example demonstrates how to use the HTMLEncode method to set the Description property before displaying it:

```
<%
   Dim pPosting, strDescription

   Set pPosting = Autosession.ThisPosting
   strDescription = Server.HTMLEncode( pPosting.Description )
   Response.Write("<P>" & strDescription & "</P>")
%>
```

Additionally, we recommend that the ASP Server.URLEncode method be used to encode all query string names and values before propagating them with the Autosession.PropagateParameter method. This will prevent users with malicious intent from appending script to the end of hyperlinks within postings and channels.

Example

This example demonstrates how the URLEncode method can be used to encode the name and value pair before propagating a query string parameter.

```
<%
   Dim strName, strValue

   ' ... set strName and strValue by accessing
   ' Request.QueryString, etc

   Call Autosession.PropagateParameter( Server.URLEncode(strName), _
          Server.URLEncode(strValue) )
%>
```

Switching templates

About template switching

The Template Switching feature allows MSCMS 2001 to automatically switch templates based on any property that can be determined using either server-side or client-side scripting. For example, after determining the user is browsing with Internet Explorer 5, you could switch to an alternate template designed specifically for IE5 browsers. Or, you could determine the user is browsing with a Personal Digital Assistant (PDA), and switch to a template designed specifically for that PDA platform.

Switching templates and alternate templates

Template switching involves a switching template and one or more alternate templates. The switching navigation template is assigned to channels, and contains the code for switching to alternate templates based on conditions. The switching template should also contain code and content for the "default" template. If alternate templates are not found, and a redirect is not possible, script errors can be ignored using "On Error Resume Next" and the switching template's content will be used.

Alternate templates are customized to suit a specific switching condition. Continuing with our example of switching based on browser types, one of our alternate templates may be designed for Netscape Navigator 4.x, and another for a PDA.

Switching logic at run time

When a channel that has a switching template is requested by a Web browser, code in the switching template evaluates the conditions set out by the template designer and either displays itself or switches to an alternate template.

To switch to an alternate template, it displays the posting using the alternate template chosen instead of using its own template.

Implementation Notes

- Alternate templates must have a unique name throughout the template galleries.
- Any property accessible using server-side or client-side scripting can be used as a switch condition.
- For page templates, the switching template must be the one that is used to create the page. For navigation templates, it must be the one assigned to the channel.
- Alternate templates must use the same placeholder names for placeholders intended to share content.

Switching with client-side JavaScript

Compatibility with MSCMS 2001 Site Stager

If your site is intended for use with MSCMS 2001 Site Stager, client-side switching using JavaScript is the method you should choose. Client-side switching works for both dynamic and staged MSCMS 2001 sites.

Example of client-side template switching

Add the following code to a switching template. Note that the code could be added to a server-side include file as a Function or Sub, taking the template name as an argument and generating the appropriate client-side JavaScript code for the switching.

```
<head>
<!-- Client side template switching -->
<script language="JavaScript"><!--

var strUserAgent = LCase(navigator.userAgent)
if (InStr(strUserAgent, "windows ce") != 0) ||
   (InStr(strUserAgent, "avantgo") != 0) ||
   (InStr(strUserAgent, "palm") != 0)
{
     window.location = "<%=Autosession.URLUsingAlternateTemplate("Product Listing
PDA")%>"
}
//-->
</script>
<!-- End of client side template switching -->
</head>
```

Description of functionality at run time

When the switching template's HTML is returned to the browser, JavaScript code executes an `If` statement to determine if a specified browser property is true (for example, the browser is running on the PDA platform). If one of the conditions is a match, the script does a client-side redirect to the alternate template.

Switching with server-side ASP

Compatibility with MSCMS 2001 Site Stager

Switching using server-side ASP is simpler than using client-side JavaScript, but it does not function after a site is staged with MSCMS 2001 Site Stager. Use client-side switching if you are developing for use with MSCMS 2001 Site Stager.

Example of server-side template switching

The comments in the following code sample, refer to a hypothetical MSCMS 2001 Web site and are intended for illustration only:

```
<%
' This template is the one referenced by the Product Listing page. However,
' if the browser is a PDA, then we'll use an alternate template which doesn't
' use tables or images. This is done using the URLUsingAlternateTemplate
' method on the Autosession.
' We recommend using client-side JavaScript for template switching if Staging is
' desirable.
If Autosession.IsModePublished Or Autosession.IsModeUnPublished Then
   Dim strUA
   strUA = LCase(CStr(Request.ServerVariables("HTTP_USER_AGENT")))
   If ( (InStr(strUA, "windows ce") <> 0) Or _
        (InStr(strUA, "avantgo") <> 0) Or _
        (InStr(strUA, "palm") <> 0) ) Then

      ' Replace the string “Product Listing PDA” with the name
      ' of the alternate template
      strURLToSwitchTo = Autosession.URLUsingAlternateTemplate("Product Listing PDA")
      Call Response.Redirect(strURLToSwitchTo)
   End If
End If
%>
```

Description of functionality at run time

When a switching template is requested by a subscriber's browser, the server-side ASP checks whether a specified set of browser properties is true (for example, the browser is running on a PDA platform). If one of the conditions is a match, a call is made to the `Autosession.URLUsingAlternateTemplate` method, passing it the name of the alternate template to switch to. `URLUsingAlternateTemplate` returns the URL for the current posting or channel, with the alternate template. A redirect occurs to the new URL. If the browser property is not true, the redirect is not done, and the current template (the one containing the browser check) is used.

Sorting a list of channels or postings

The ItemCollection object

The `Channels` and `Postings` objects encapsulate any collections of channels and postings. The collections exclude items that are deleted or that point to deleted items. The `Channels` and `Postings` objects are usually used for looping through or sorting collections consisting of channels, postings, or a combination of both.

Setting an explicit order for a channel's children

Properties and methods that return a collection of channels normally return collections sorted by their `SortOrdinal` value, which is set by an administrator or moderator. It refers to the order set in the channel's Properties dialog box, under the Postings tab.



Note
A moderator can sort only in a channel, not a subchannel.

If an administrator does not explicitly set the order of the subchannels and postings in a channel, sorting by `SortOrdinal` will have no effect. Channel properties which would return collections of channels sorted by `SortOrdinal` will default to sorting by name; collections of postings will default to sorting by `ChangeDate`.



Note
To make Publishing API sorting methods language-sensitive, the `SessionSettings.Locale` property should be set to an appropriate value. See the *MSCMS 2001 Publishing API Help* for more information.

Sorting collections by their properties

Use the sorting methods, such as `SortByName` and `SortByChangeDate`, to sort a collection by the value of a particular property. For each sort method, the default direction is given.

Method	Default Direction	Example of Default Sorting
SortByOwner	Ascending	"WinNT://Domain/Sally", "WinNT://Domain/Tom"
SortByName	Ascending	"Products", "Services"
SortByDisplayName	Ascending	"Products", "Services"
SortByStartDate	Ascending	8/21/98 10:51:00 AM, 8/21/98 8:51:00 AM
SortByExpiryDate	Ascending	8/22/98 10:51:00 AM, 8/22/98 8:51:00 AM
SortByLastModifiedDate	Ascending	8/20/98 10:51:00 AM, 8/20/98 8:51:00 AM
SortByChangeDate *	Ascending	8/19/98 10:51:00 AM, 8/19/98 8:51:00 AM
SortByOrdinal	Descending	(in the exact order set in the channel's properties dialog box)

SortByRevisionDate	Ascending	19/98 10:51:00 AM, 8/19/98 8:51:00 AM
SortByExtension	Alphabetical	doc jpg xls
SortByMimeType	Alphabetical	application/msword application/vnd.ms-excel image/jpeg
SortBySize	Ascending	Smallest to largest size
SortByPath	Ascending	"/Channels/Framed sample", "/Channels/Intranet"
SortByImportance	Important first	(items set to Important appear first)

* See the *MSCMS 2001 Publishing API Help* for details on how the `ChangeDate` property is determined, and how it differs from the `LastModifiedDate` property.

All sort methods have a Boolean parameter that determines whether to sort in an ascending direction or not. If not explicitly specified, the defaults listed in the previous table are used.

- True—sorts in the ascending direction for the property.
- False—sorts in the descending direction.

The following is an example of using the `Channels` and `Postings` objects to sort postings in alphabetical order. The sort direction of the `SortByDisplayName` method is not specified, so the default direction (ascending) is used:

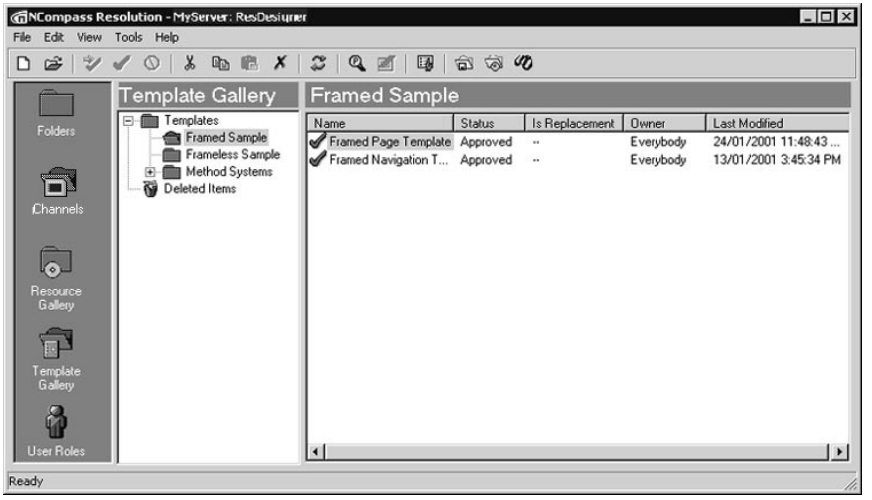
```
Dim pPostings
Set pPostings = Autosession.ThisChannel.Postings
Call pPostings.SortByDisplayName
```

Chapter 5

Working within the MSCMS 2001 Site Builder

Opening the Site Builder

Navigation templates are created and maintained within the MSCMS 2001 Site Builder. To open the Site Builder, select the Site Builder icon on your desktop, or select Site Builder from the Microsoft Content Management Server Start program group.



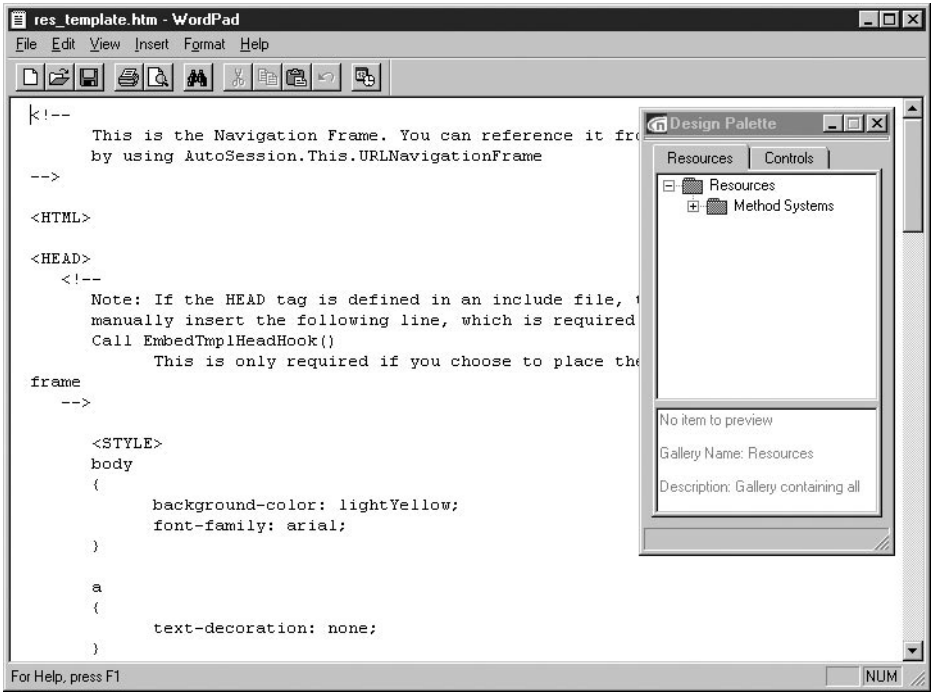
Sample Site Builder window

Configuring an HTML editor

About compatible HTML editors

We recommend using an editor designed for working with code, such as Microsoft FrontPage 2000. The only requirement for your HTML editor is that it doesn't modify code.

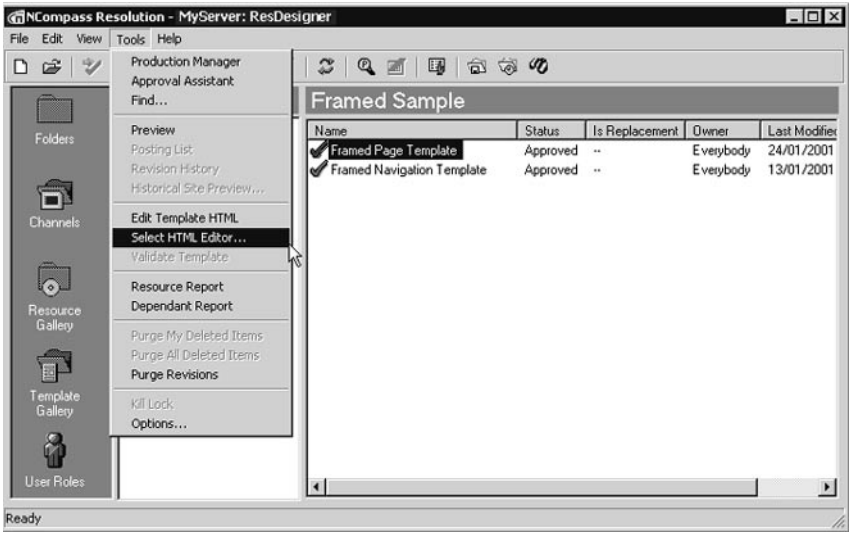
MSCMS 2001 works in a joint session with the configured editor to create or edit pages. When a file save is detected by MSCMS 2001, the template is automatically saved to the template gallery.



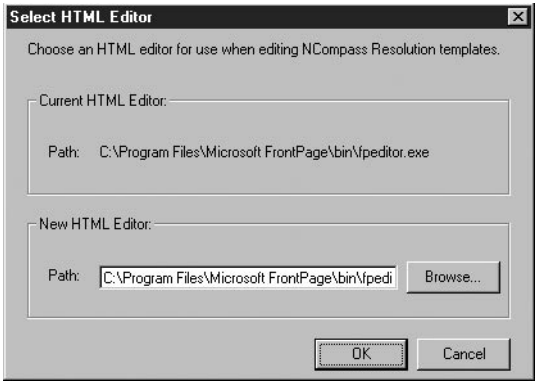
HTML editor and MSCMS 2001 Design Palette

Configuring your HTML editor

1 In the Microsoft Content Management Server window, choose the **Select HTML Editor** command from the **Tools** menu. The Select HTML Editor dialog box opens.



2 In the Select HTML Editor dialog box, type in the full pathname to your editing application, or use the Browse button to locate it.

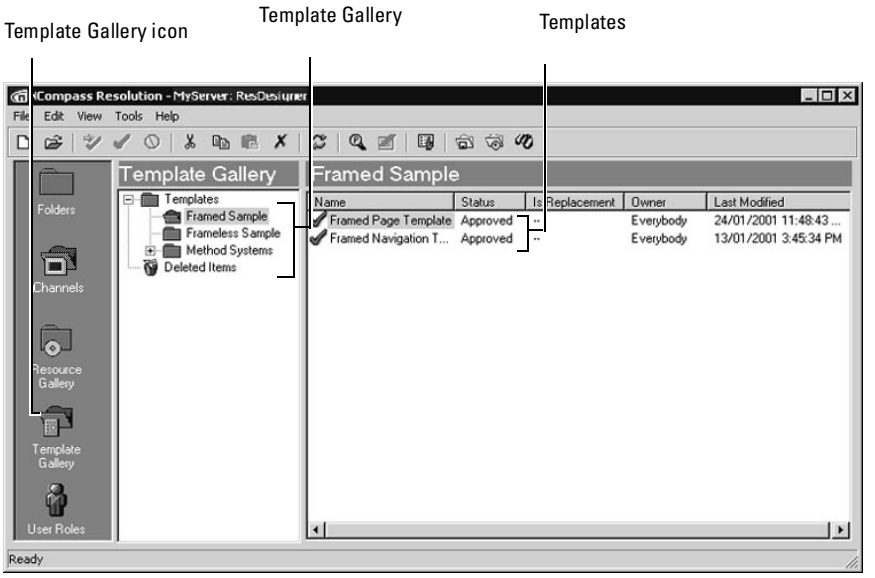


3 When finished, click **OK**.

The HTML editor can be re-configured at any time by repeating this setup procedure.

Locating navigation templates in MSCMS 2001

1 In the Microsoft Content Management Server window, display the template gallery hierarchy by selecting the Template Gallery icon.



Template Gallery hierarchy expanded, showing the page templates it contains

2 Click a gallery to display a list of templates in it. Navigation templates should be stored in a subgallery of the Navigation Templates gallery.

Note

You can work only in template galleries you have rights in. The icon that appears next to a template gallery's name in the Content Management Server window's list of items indicates what rights you have in the gallery. If the icon is blue, you can create and edit templates in the gallery. If the gallery's icon is dimmed, you have no rights to templates in that gallery. You can browse through dimmed folders (ones you do not have rights in) to reach blue colored folders you do have rights to.

About the template Design Palette

For more information about the template Design Palette, see “The template Design Palette” in Chapter 2, “Creating a Framed MSCMS 2001 Interface.”

Quick start to template creation

There are three ways to create a template:

- copy an existing template and paste it into a template gallery
- import an existing HTML or ASP file
- create a new one.

You may find it OK to start out by copying and editing an existing template that is close to your required functionality.

Copying a template

To copy a template:

1 Right-click the name of a template and select **Copy**.
2 Right-click the gallery you want to paste the template in and select **Paste**.
3 Single-click the template and re-name it.

Editing the HTML of an existing template

1 Right-click the template to be edited; from the right-click menu, select **Edit Template HTML**.
2 A new HTML file opens in your configured editor.
3 When you are finished editing, save the HTML file in your editor; the navigation template is updated in the MSCMS 2001 gallery.

   You are prompted to select an editor if you have not configured one yet. See “Configuring your HTML Editor” previously in this chapter for information on HTML editors.



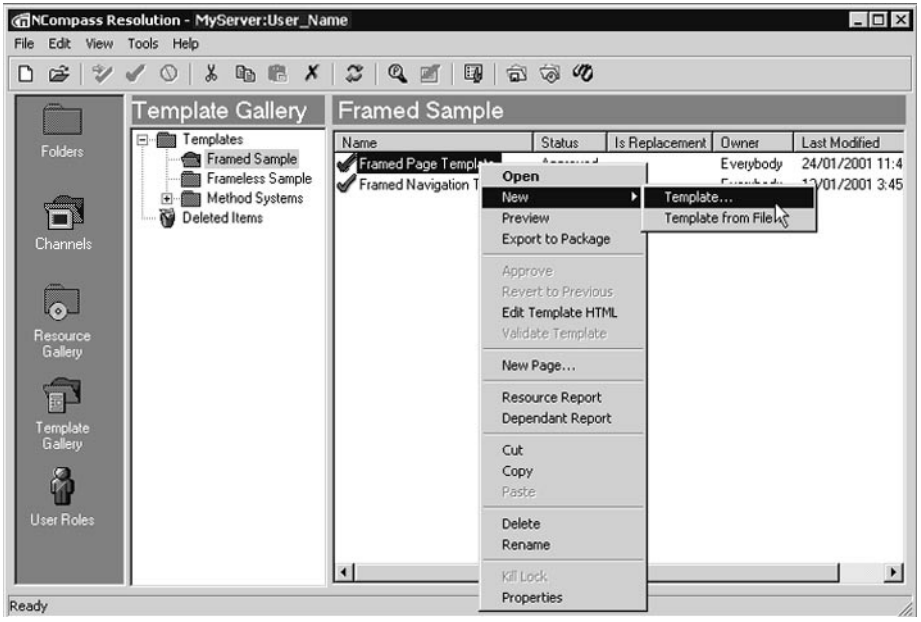
Note
Edited templates must be approved before they are available for use. See “Approving Templates” later in this chapter.

Creating a new template

Tip

We recommend starting out by editing one of the existing sample navigation templates. See "Editing the HTML of an existing navigation template" in this chapter. The Publishing API is documented where relevant throughout this manual, but for reference purposes, Chapter 4, "Using the MSCMS 2001 Publishing API" and the *MSCMS 2001 Publishing API Help* will be of the most use.

1 Right-click the gallery you want to create the template in, and select **New**, then **Template**.



2 In the Select Template Type dialog box, select **Navigation Template**. Click **OK**.

3 A new HTML file opens in the editor you configured.

4 When finished coding your template's functionality, save the HTML file in your HTML editor; the template Design Palette imports and saves the HTML file as an "Untitled" template in the gallery you created it in.

Important: Do not use "Save As" in the HTML Editor.

Assigning custom properties to your template

With MSCMS 2001, template designers and administrators can easily customize page template properties from within the Site Builder. Authors, editors, and moderators can display current template information in read-only format. From the Properties dialog box, template designers can:

- view custom properties currently assigned to their template
- add their own custom properties
- modify or delete existing custom properties.

Setting custom properties

All custom properties must be created through the Site Builder, but can be edited using the Web Author. Custom properties for a page are added to the page template by the template designer. From the Properties dialog box, template designers can:

- view custom properties currently assigned to their template
- add their own custom properties
- modify or delete existing custom properties.

Benefits of using custom properties

Custom properties are user-defined properties with a specific set of names and values associated with MSCMS 2001 channels and pages. Provided your custom properties are defined consistently and your Web programmer has added the applicable Application Programming Interface (API) search methods to the scripts, you can:

- create summary pages of posting and channel content
- create cross-reference between pages and channels
- categorize pages and channels based on content
- search all pages and channels
- facilitate customized scripting.

An example: categorizing your pages on the basis of content

You can set a range of values for a custom property that correspond to the types of pages on your site. For example, if you work in computer retail, you can define a list of values including laptops, PCs, and servers corresponding to the different types of computer equipment you stock. Authors then apply these values to their pages when they add content. Next, site programmers insert programming logic to do searches based on a custom property used to group pages together. Site users are presented with customized lists of pages based on product preferences.

Guidelines for working with custom properties

Sharing custom property values on a page

- For a posting, the set of custom properties, the default value, and the set of allowed values (if applicable) are inherited from the page template. Inheritance of custom properties works the same way as for placeholders
- If two postings share the same page (for example, connected postings), regardless of whether they share the same page template, and their respective page templates have a custom property with the same name (considered to be the same property), then any override value for this custom property that is set through one of the connected postings is automatically reflected in the other connected posting.

Best practices for adding custom properties

To add the same custom property on several page templates (or channels), follow these guidelines:

- Give the custom property the same name. It is important that custom properties be named consistently if they are to be considered the same property during searches or across page templates.
- Make the custom property the same type (Text versus Selection).
- If the custom property is a Selection type, be sure to add the same set of allowed values in all page templates (or channels). Spelling is important.
- Avoid adding custom properties in different templates (or channels) with the same name but different types or allowed values. This makes searching confusing and not as effective.
- Use the same case for names and allowed values if your database defaults to doing case-sensitive searches.

Custom properties and the API

For more information on API methods and properties, see the *MSCMS 2001 Publishing API Help*.

Adding a custom property to your template for the first time

1 Access the Properties dialog box as follows:

- Select your page template and choose **File > Properties**
- Select your template and click the **Properties** button
- Right-click your template and select **Properties**.

2 From the Properties dialog box, click the Custom tab.



The Custom tab of the page template Properties dialog box

3 Select **New**. The Add Custom Property window appears.

4 Enter the name of your custom property in the Property Name field.

Note

If your new property has the same name as an existing property for that particular template or channel, you receive an error message and cannot save your work until the name is changed.

5 Select a property type from the available drop-down list.

- If you chose Text property type, specify the default text in the Default Value field below, and proceed to step 6.
- If you chose Selection property type, you must first define a set of allowed values. See the next section for details. Once the list is defined, select a value from the list, and

click **Set As Default.** The value you selected then automatically appears in the Default Value field.

6 Click **OK**. The Page Template Properties window reappears. The property definition you just added now appears in the list.

Creating a set of allowed values

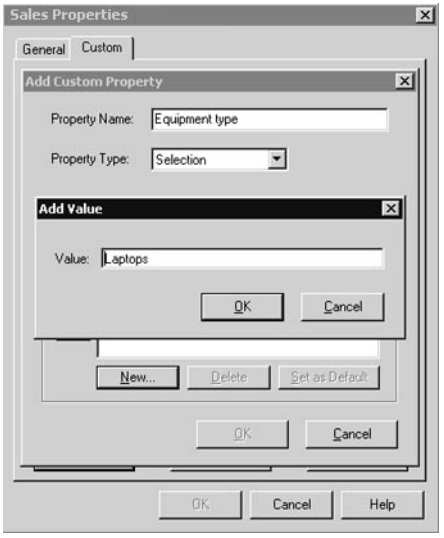
1 From the Properties dialog box, click the Custom tab. The Add Custom Property dialog box appears.

2 Specify "Selection" as the property type.

3 Enter a description for your new property in the Property Name field. For example if you are defining properties for the computer retail scenario cited earlier, the first property could be "Equipment Type." Other related properties could be "Price," "Price Range," "Features," and "Availability."

4 Click **New**. The Add Value dialog box appears.

5 Now, start defining a range of allowed values for your new property. Enter the first value in the Add Value dialog. For example, if you are defining values for the computer retail scenario cited earlier, the first value in your list of values for the Equipment Type property could be "Laptops." Others could be servers and PCs.



Tip
Each value must be unique within your list. An empty string value is permissible.

Modifying or deleting existing custom properties

To modify the custom properties currently assigned to your template:

1 From the Properties dialog box, click the **Custom** tab.
2 Select a property definition from the list displayed.
3 Click **Modify** or **Delete** as required.
   - If you clicked **Delete** the custom property disappears automatically from the list of values**.**
   - If you clicked **Modify,** proceed to step 4.
4 From the Edit Custom Property dialog box, modify your custom property as desired.
5 Click **OK**.

Note
When deleting a property on a template that is in use, you receive a warning message.

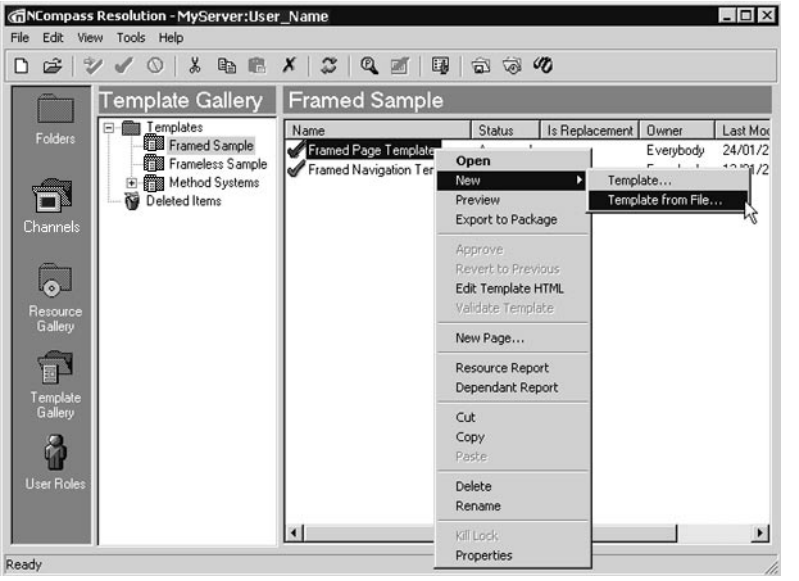
Tip
- If you are creating the same properties in multiple templates, we recommend creating them in the first template and then copying and pasting the template. This will enforce consistent names and allowed values for custom properties required for powerful searching
- If a posting or page is based on a template, and a custom property associated with the template is deleted, then the deleted custom property is not reflected in any posting or page.
- If a custom property for a page or channel cannot be edited, the most likely reason is that it is not in the same language as the operating system of the machine being used. For example, the property may have been created on a different machine with an operating system in a different language.
- Page and posting properties can also be edited and reviewed (but not set) using the Web Author. See Chapter 8, “Working from the MSCMS 2001 Web Author” for details.

Importing templates

You can import a suitable HTML (or ASP) file into a template gallery to add it as a template.

1 In the Microsoft Content Management Server window, display the template gallery hierarchy by selecting the Template Gallery icon.

2 Right-click the name of a gallery in the hierarchy and select **File** then **New** then **Template from File**. The Select Template dialog box opens.



3 Select a suitable HTML (or ASP) file to import from a local or network drive—the Select Template Type dialog box opens.

4 Choose to import the file as a navigation template. MSCMS 2001 saves the file to the gallery you started the process from.

Using external resources

Text and HTML files cannot be stored within the Resource Gallery. They must be stored in directories on the server.

Links to resources stored outside of the Resource Gallery require a root pathname (for example, "/mydir/mydir/mydoc/"). If you use relative paths, the target of the links may not be accessible under all circumstances.

If your site uses MSCMS 2001 Site Stager, wrapping links to external resources with the `ResolveURL()` function is necessary if you want the target of the link to be staged. Chapter 10 "Developing for MSCMS 2001 Site Stager" has complete instructions on applying `ResolveURL`.

MSCMS 2001 produces a Link Error dialog box when saving a navigation template with links to file formats that can't be imported into the MSCMS 2001. Click **OK** to continue to parse the file. The links are accessible to the Web browser that loads the template.

Setting the Frameset URL

To associate the navigation template with its frameset file, we need to set the template's Frameset URL property in the Site Builder. The navigation template can be assigned to only one Frameset URL at a time. We assign the Framed Sample Navigation Template to the frameset file we created earlier, Frameset.asp.

To assign the Frameset URL property:

1. Right-click the Framed Navigation Template in the Microsoft Content Management Server window, and select **Properties**.
2. Click the Publishing tab.
3. Select the **Framed** option.
4. In the Framed text box, enter /NR/Framed Sample/Frameset.asp. This is the virtual directory path to the frameset file we created. See Chapter 1, "Elements of an MSCMS 2001 Web Interface" for more information on how MSCMS 2001 virtual and physical directories map.

Previewing the template

Before approving the template, thereby making it available for use by channels, preview it in a Web browser.

You can view the template as it would appear in a browser to a user with subscription rights only, or as it would appear to a user with publishing rights.

See "Previewing the template" in Chapter 2, "Creating a Framed MSCMS 2001 Interface" for more information.

About template workflow

To prevent templates from "going live" while they are being created or edited, a simple workflow is followed. There are two statuses in the workflow: Saved and Approved.

Saved

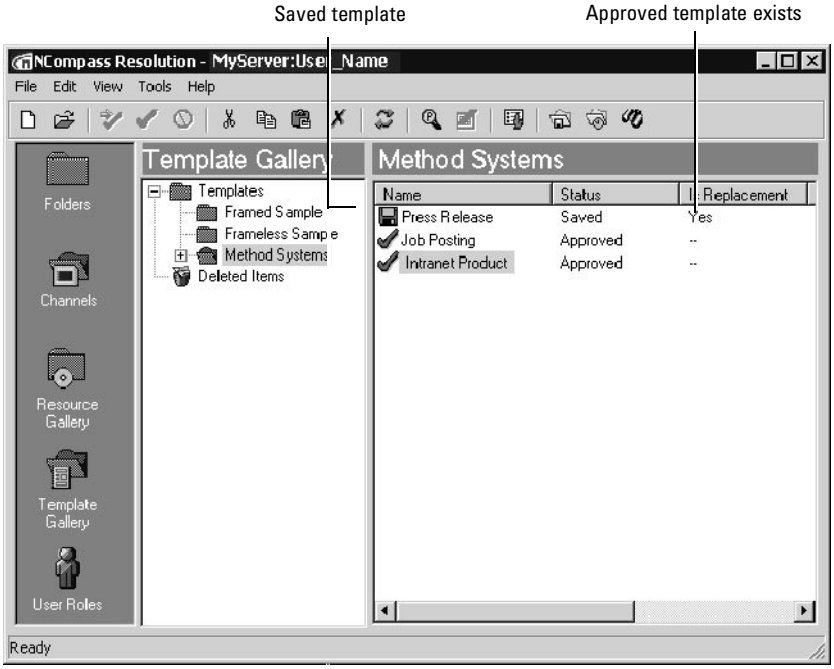
Saved templates are new templates that are not yet available for use by channels. When a navigation programmer decides the template is ready for use, they approve it, thereby changing its status to Approved.

Approved

Approved templates are available for use as a channel's navigation template.

Checking the status of a template

1 In the Microsoft Content Management Server window, display the template gallery hierarchy by selecting the Template Gallery icon.



Template Gallery in the Microsoft Content Management Server window

2 Select the name of a gallery in the hierarchy. A list of templates in the gallery (if any) appears on the right.

The icon beside each template's name and the Status column indicate the template's status.

An approved template that has been edited, but not re-approved yet, has a Saved status and a "Yes" in the Is Replacement column. The Approved version remains ready for use by authors, until the Saved version is approved.

Approving templates

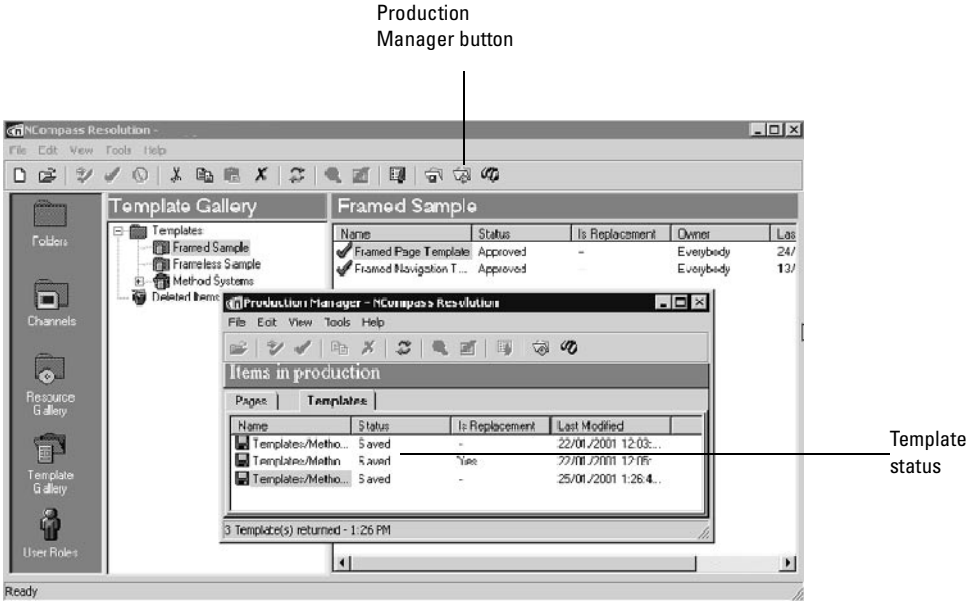
When you approve templates, they are ready for immediate use by authors. Edited templates must be approved before changes made to them take effect.

Using the Production Manager

The Production Manager is an efficient way to see which pages, postings, and templates have not been completed and approved. While it is possible to browse the container hierarchy to find templates you are working on, in most cases it will be easier to use the Production Manager. This is especially true if you are working on many items at one time. The Production Manager creates a list of all page templates whose status is Saved.

Approving a template

1 Click the **Production Manager** button on the Microsoft Content Management Server window toolbar (or right click a template, and select Production Manager). The Production Manager window opens and displays a list of your templates (including those in the Deleted Items container) whose status are marked as Saved.

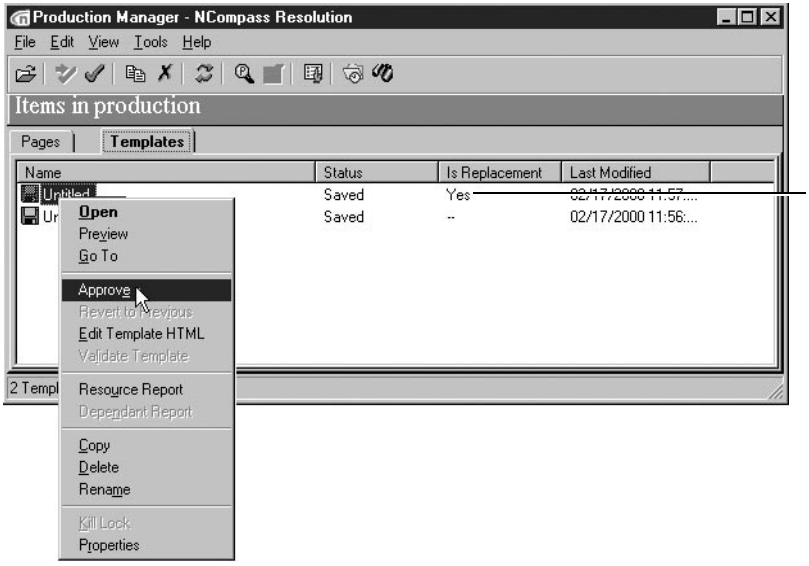


Production Manager window

Tip

Next to the Production Manager is the Approval Assistant. The Approval Assistant is used by editors and moderators to search for pages and postings awaiting their approval.

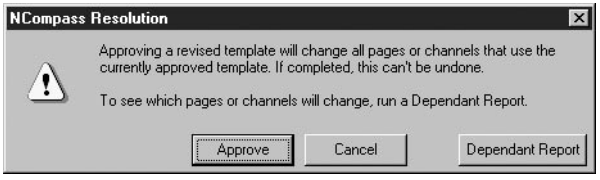
2 To approve a template, right-click its name in the Production Manager window and select **Approve**.



Approving Saved templates from the Production Manager window

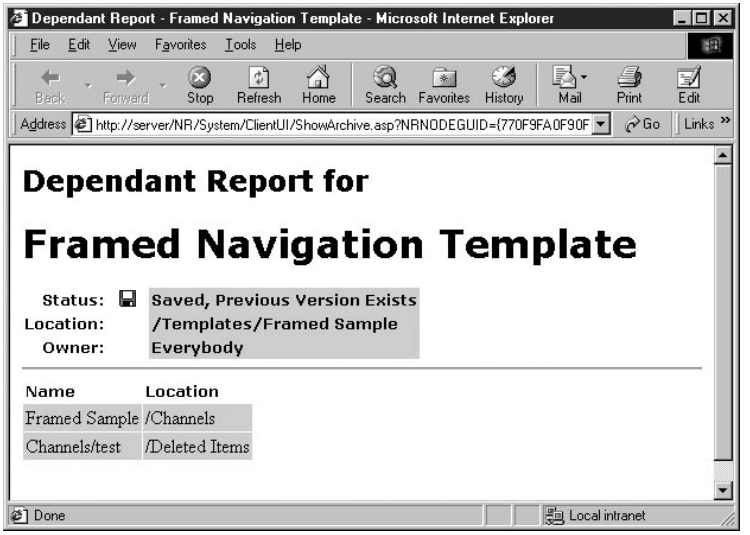
3 If the template has not been Approved before, the template is approved, and its status changes to Approved.

4 If the template has been Approved before, the Template Approval Warning dialog box opens, suggesting that you may want to do a dependancies check before you continue, in order to see what pages the changes might affect. To obtain a Dependant Report, click the **Dependant Report** button. Otherwise, click **Approve**.



Template Approval Warning dialog box

5 The Dependant Report shows channels (if any) that use the template. Click the **Preview** button beside the name of the dependant item to view it in a Web browser.



Dependent Report for a navigation template

6 When ready to approve the template, close the Dependant Report and click **Approve** on the Template Approval Warning dialog box. The template's status changes from Saved to Approved.

Tip
See "Previewing the Template" as described in Chapter 2 for instructions on viewing the Approved version (if any) of a Saved template.

Tip
You can also use the Find feature to search for Saved (or Approved) templates and approve them.

Reverting a Saved template to its Approved version

If an edited template has not been re-approved (it is still in its Saved state), you can cancel the edits and revert the template to its former Approved status.

1 If the Template Validation Failure dialog box is open, click **Cancel** to close it.

2 Right click the template in the Content Management Server window, and select **Revert to Previous**. The template's Saved version is removed and the template reverts to its Approved version.

Note
You cannot revert a template while the template Design Palette is open.

Generating a Dependant Report

Before modifying a navigation template with a status of Approved, a navigation programmer should do a dependency check to see what channels are using the template.

1 Find the template for which you want to generate the Dependant Report. You can do this in one of several ways: browsing through the template gallery; using the Production Manager; or using the MSCMS 2001 Find feature.
2 Right-click the template name and select **Dependant Report**.
3 A Web browser is launched, displaying a Dependant Report with a summary of the channels that are currently using the navigation template.

Deleting a template vs. purging a template

When a template is deleted, it is moved to the Deleted Items container where it is no longer available for use by channels. To make the deleted template available again, move it from the Deleted Items container back into a template gallery.

When a template is purged, it is permanently removed from the Deleted Items container.

! Caution
There is no similar distinction between deleting and purging in the Web Author. In the Web Author, pages are purged with a delete action. If an author deletes previously approved content, they cannot recover that content.

Deleting a template

Right-click the template to delete, and select **Delete**.

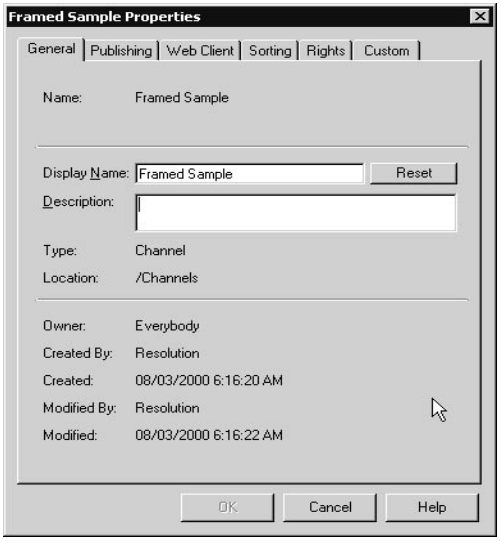
If the template is not in use by any channels, it is moved to the Deleted Items container. Otherwise, an error message appears, informing you of the existence of channels that depend on the template. These channels must be assigned another navigation template or deleted before the template can be deleted. Only administrators can modify properties or delete channels.

To see a list of the dependant channels and their locations, click the Dependant Report button on the error message dialog box. The report is displayed as an HTML page in your Web browser.

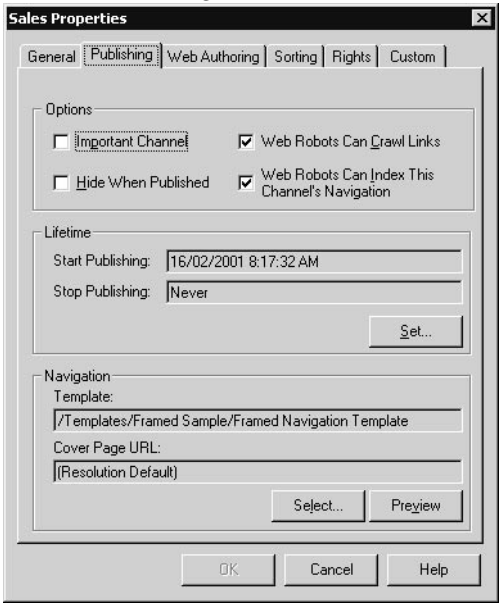
Setting a channel's Cover Page URL property

1 In the Content Management Server window, display the Channel hierarchy by selecting the Channels icon.

2 Right-click the channel, and select **Properties**. The channel's Properties dialog box opens.



3 Click the Publishing tab.



4 Click **Select**. The Select Channel Navigation dialog box opens.



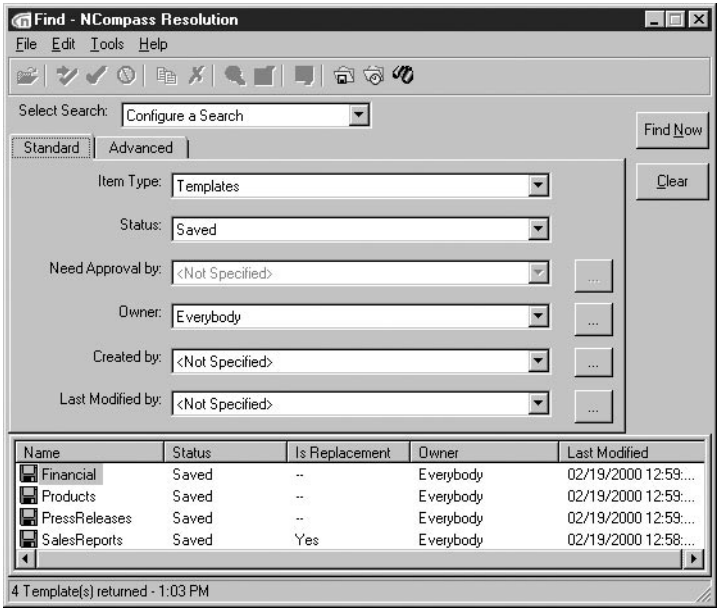
5 Type in an external URL, or select a channel or posting.

Note
If a subscriber does not have access to the channel or posting specified as the Cover Page URL, or posting or channel is deleted, the default cover page is returned.

Using the Find feature

To configure the MSCMS 2001 Find feature to search for templates:

1 Click **Find** on the Microsoft Content Management Server window toolbar. The Find window opens. Select the Standard tab on the Find window.



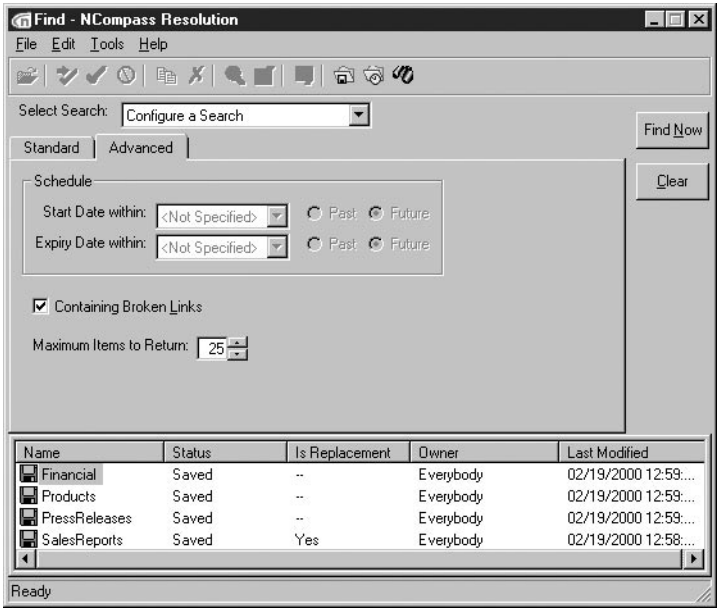
The Standard tab in the Find window

2 Click the **Item Type** drop-down list, and select **Templates**.

3 If you have used Find before, click the **Owner** drop-down list and select your name. If this is the first time you have used the Find feature, click the Browse button beside the Owner drop-down list. The Select User dialog box opens. On the Select User window, choose the appropriate domain, and select your name from the list of domain users. Click **OK**. For subsequent searches, your name appears in the Owner drop-down list.

4 On the Find window, click **Find Now**. A frame opens at the bottom of the window, listing all items matching your search. For example, the search described in the preceding steps returns all page templates and navigation templates that you own. Any templates in the Deleted Items folder matching the search criteria are also returned.

5 To search for templates that contain broken links, click the Advanced tab.



The Advanced tab in the Find window

6 Click the Containing Broken Links option to enable it, and click **Find**.

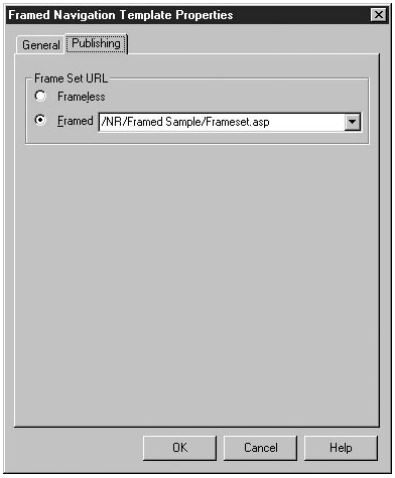
Chapter 6

Site Programmer's FAQ

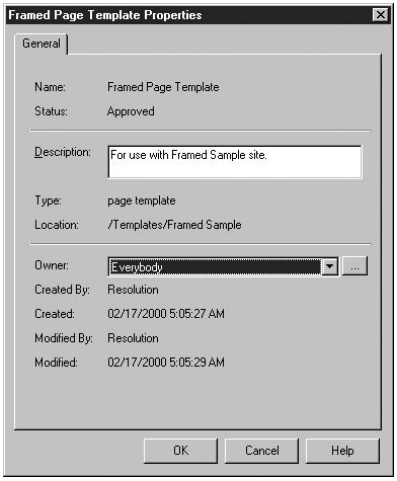
How to distinguish between page and navigation templates

To help distinguish between page and navigation templates, we recommend you store all navigation templates in subgalleries of a template gallery named "Navigation Templates."

If you are ever unsure whether a template is a navigation template or a page template, check the template's properties by right-clicking the template name and selecting **Properties**. Navigation templates include a Publishing tab with information for a Frameset URL; page templates do not require Frameset URLs.



Navigation template properties



Page template properties

What scripting languages are supported?

The MSCMS 2001 Publishing API can be called from any scripting language supported by Active Server Pages, including server-side JavaScript and VBScript.

How do I redirect to a different template?

See the section "Switching templates" in Chapter 4, "Using the MSCMS Publishing API" for information on implementing Template Switching.

Can I use Java or ActiveX with a template?

One disadvantage of using JavaScript, Java Applets or ActiveX for controlling navigation on the client side is that any of these may be manually disabled in the Web browser. By relying on the server-side VBScript and the MSCMS 2001 Publishing API, you retain more control over the generated pages. However, if you are using Site Stager, logic performed on the server may not be an option.

What does a template's status represent?

See the section see "About template workflow" in Chapter 5, "Working within the MSCMS 2001 Site Builder" for information on template statuses.

What is a frameless page template outline?

Frameless page templates must define the layout and source of navigation elements. To make page template designers' jobs easier, it is good practice to create an "outline template" that links to the navigation elements. Chapter 3, "Creating a Frameless MSCMS 2001 Interface" provides sample code for creating a frameless page template outline.

What is MSCMS guest user access?

Administrators can designate, during the server install or later using the Server Configuration Application, whether or not to allow guest user access. MSCMS 2001 guest access is separate from IIS guest access. When an administrator enables MSCMS 2001 guest access, they must also select a Windows 2000 user to use as the guest. The administrator must then assign the user to a subscriber rights group within MSCMS 2001 that specifies which channels the guest user has access to.

As the authentication scheme is currently set up, after trying to log on using a cookie from a previous (successful) login, Authenticate.inc tries to authenticate as the Default Guest user (see the section "Authentication" in Chapter 1, "Elements of an MSCMS 2001 Web Interface" for more information). You can also provide a logout link using the `DefaultGuestLoginURL` method. Following this link executes script in DefaultGuestLogin.asp, which logs you out. Then it determines whether guest access is enabled by querying the `Autosession` object's `IsDefaultGuestEnabled` property, and redirects to the requested item if it is (authenticating under the guest user), or redirects to AutomaticLogin.asp if it is not.

Where do the objects in Preview mode come from?

When using the Preview command with templates, a predefined hierarchy of simulated channels and postings is generated to provide reasonable return values for calls to Autosession and other Publishing API objects. See "About simulated channels" in Chapter 4, "Using the MSCMS 2001" Publishing API for more information.

How do I control browser caching?

Because MSCMS 2001 is based on the use of dynamically generated pages, there are some situations where a Web browser should be prevented from using cached versions of files. Use the following ASP code to instruct the browser to always retrieve the latest copy of the file from the server:

```
<%
Response.Expires= "0"
%>
```

If the browser logs out and then logs back in as a different user, the channels and pages they have access to may have changed. For this reason, navigation templates should also be refreshed every time a frameset is loaded.

Chapter 7

Introducing the MSCMS 2001 Web Author

What is the MSCMS 2001 Web Author?

Anyone anywhere in your organization with access to a Web browser can be allowed to create and distribute pages, using the MSCMS 2001 Web Author.

The Web Author is a set of server-side scripts that generate an editable version of an existing or new MSCMS 2001 posting. The Web Author allows authors, editors, and moderators to switch a posting from Live mode, to Edit mode, and to Authoring mode. With MSCMS 2001, resource managers can add, delete, edit, or replace resources.

In Authoring mode, placeholders are generated as editable HTML form elements. Authors add content or make changes to existing content in a posting, and submit the changes to MSCMS 2001 for approval by an editor and moderator.

Understanding the MSCMS 2001 publishing system

We recommend reading the *MSCMS 2001 Authoring and Publishing Guide* and the *MSCMS 2001 Site Administrator's Guide*, to gain a thorough understanding of the MSCMS 2001 publishing system. Following is a brief summary.

Content Management Server 2001 manages content by allowing you to:

- store content in a database
- ensure content is approved before it is published
- control access to pages
- control when pages are published.

Pages and postings

To understand pages and postings, it is important to understand how the Web Author and Site Builder work together when authors create content. When an author uses the Web Author to create a page of content and then submit it for approval, the result is a new posting. A posting includes a publishing schedule with start and expiry dates, and the channel to publish it on. When a posting is created in the Web Author, the page for that posting is stored in a folder. Folders and pages are visible only in the Site Builder.

Channels

Channels are used to store, organize, and manage access to content. Channels organize postings into a hierarchy, which behaves differently from the directory structure of traditional file-based Web sites in several ways. Entire branches of your site can be moved (or renamed) without breaking links between postings. Since navigation is dynamically generated and it is based on the channel hierarchy itself, all posting links immediately reflect your site's new structure.

Roles

MSCMS 2001 distributes publishing responsibility among several users. Each user is given a role. Users act within their role to either create content or manage the Web site.

Posting creation and approval roles

There are three key roles: authors, editors, and moderators on the content-creation side of MSCMS 2001. Authors, editors, and moderators should do their work in the Web Author.

- **authors:** create postings, enter content in placeholders, and submit the posting for approval by an editor. Authors do not need to know HTML.
- **editors:** approve or decline postings, based on their content
- **moderators:** approve or decline postings that are posted to channels. Moderators ensure that their channels contain relevant postings.

Site management roles

There are three key roles on the site-management side of Content Management Server 2001: administrators, template designers, and resource managers. Most, but not all, site management tasks are done using the Site Builder.

- **administrators:** create channels, assign users to groups, and give groups rights to containers (channels, folders, and galleries) using the Site Builder.
- **template designers:** create page templates used by authors to create pages, navigation templates (which control the navigational elements to a site), and design the interface to browser-based clients using the Site Builder. This is possible only through the Site Builder.
- **resource managers:** manage resource galleries, which contain images, videos and sound files for use by authors and page template designers. Resource managers can add, delete, edit, or replace resources in the Site Builder or Web Author environments.

Rights

Content Management Server 2001 administrators manage user rights and permissions with the Site Builder.

In the MSCMS 2001 Publishing API, a user's rights (what they can do) are determined by their role within Content Management Server 2001. Users can belong to more than one group.

Permissions to items in the Content Management Server 2001 system come from the container objects (channels, template galleries, resource galleries, and folders) that have been assigned to the rights group(s) the user belongs to.

About setting up publishing

Content Management Server 2001 publishing requires both the Web Author and the Site Builder. Authoring and approval takes place in the Web Author; site management is done in the Site Builder.

The Web Author operates from the site created in the Site Builder. In the Site Builder, a Content Management Server 2001 administrator sets the site's channel structure, and user rights and permissions. Before authors can use the Web Author to create pages, the Content Management Server 2001 administrator must use the Site Builder to do the following tasks:

- build the channel structure
- create a corresponding folder for each channel. Folders do not appear on your Content Management Server 2001 Web site, or in the Web Author.
- create template and resource galleries
- create rights groups, assign users to them, and assign the rights groups to containers (folders, channels, template galleries, and resource galleries).

These tasks are covered in detail in the *MSCMS 2001 Site Administrator's Guide*.

Location of the Web Author files

The Web Author files are in the directory, /NR/system/WBC/.

Getting started

Adding required tags to templates

Templates that are enabled for use with the Web Author must have the following tags:

- < head > and < /head > pair
- < body > and < /body > pair

Content Management Server 2001 requires these tags when it inserts Web Author scripts into templates stored in the MSCMS 2001. The following example shows the scripts (in italics) that Content Management Server 2001 adds:

```
<Tamil>
<head>

<% Call EmbedTmplHeadHook%>

</head>

<body>
```

```
<% Call EmbedBodyBeginHook%>

<% Call EmbedBodyEndHook%>

</body>

</html>
```

Enabling the Web Author

The Web Author user interface has two essential components that must be integrated in a Content Management Server 2001 site's templates:

- a link to switch between Live and Edit mode
- a set of links for Web Author authoring.



Note
Content Management Server 2001 automatically adds `Resolution.inc` (or `Resolution_RT.inc` for read-only sites) to all templates stored in the MSCMS 2001 database which in turn adds all necessary include files such as Authenticate.inc

Enabling the Web Authors on frameless sites

To enable the Web Author for a frameless site, add the following statements to all frameless page and navigation templates:

1 To display the link to switch between Live mode and Edit mode in the frame (this link appears only for users with Content Management Server 2001 publishing rights), add the following statement: `<% ShowSiteModeSwitch %>`.

   The `ShowSiteModeSwitch` sub-routine is defined in `SiteModeUI.inc` and can be customized.

2 To display the Web Author authoring console in the frame, add the following statement:

   `<%EmbedFlatHTMLConsole %>` or `<%EmbedFDroplistConsole %>`

   These sub-routines are defined in `ConsoleUI.inc`, and can be customized.

Enabling the Web Author on framed sites

For a framed site, you must add Resolution.inc and certain include files (shown in the following steps) to ASP files stored on your server. For example, you might want to include a banner file with links to common elements such as Help.

Framesets must use the URLs returned by `Autosession.This.URLNavigationFrame` and `Autosession.This.URLPostingFrame` to dynamically populate the frames that contain navigation templates and page templates from the Content Management Server 2001 system.

Query string properties can be passed to frames generated by external ASP files (files stored on the server, rather than templates within the Content Management Server 2001 system) by appending the value returned by `Autosession.This.QueryString` to their list of query string parameters of the URL. The information in the query string allows ASP files stored on the server to call `Autosession.This`, `Autosession.ThisPosting`, and `Autosession.ThisChannel`.

Add the following statements to templates used in the navigation and page frames:

1 To display the link to switch between Live mode and Edit mode in the frame (this link appears only for users with Content Management Server 2001 publishing rights), add the following statement:

   ```
   <% ShowSiteModeSwitch %>
   ```

   The `ShowSiteModeSwitch` sub-routine is defined in `SiteModeUI.inc` and it is customizable.

2 To display the Web Author authoring console in the frame, add the following statement:

   ```
   <%EmbedFlatHTMLConsole %> or <%EmbedFDroplistConsole %>
   ```

These sub-routines are defined in `ConsoleUI.inc`, and can be customized.

The following statements must be added to frames that use ASP files stored on the server:

1 Content Management Server 2001 automatically adds `Resolution.inc` to the top of your ASP file:

   ```
   <!--#include virtual="/NR/System/Access/Resolution.inc"-->
   ```

2 You must add EmbedTmplHeadHook immediately before the </head> tag:

   ```
   <% EmbedTmplHeadHook %>
   ```

   The resulting HTML should look like this:

   ```
   <!--#include virtual="/NR/System/Access/Resolution.inc"-->
   <html>
<head>
<% Call EmbedTmplHeadHook%>
</head>
   ```

3 To display the link to switch between Live mode and Edit mode in the frame (this link appears only for users with Content Management Server 2001 publishing rights), add the following statement:

```
<% ShowSiteModeSwitch %>
```

The `ShowSiteModeSwitch` sub-routine is defined in `SiteModeUI.inc` and can be customized.

4 To display the Web Author authoring console in the frame, add the following statement:

```
<%EmbedFlatHTMLConsole %> or <%EmbedFDroplistConsole %>
```

These sub-routine are defined in `ConsoleUI.inc`, and can be customized.

Enabling the Web Author for exceptional cases

You might use a template that references only ASP files and, therefore, does not contain opening and closing < head > and < body > tags. For example:

```
 <html>
<!-- #include virtual="/NR/Includes/HeadSection.inc" -->
<!-- #include virtual="/NR/Includes/BodyBegin.inc -->
<!-- #include virtual="/NR/Includes/BodyEnd.inc -->
</html>
```

In this case, you must add the following statements in the include files referenced by the template:

1 To follow our example, in the file named HeadSection.inc, you would add EmbedTempHeadHook immediately after the opening < head > tag:

```
<% Call EmbedTmplHeadHook %>
```

2 In the BodyBegin.inc file, you would add EmbedBodyBeginHook immediately after the opening < body > tag:

```
<% Call EmbedBodyBeginHook %>
```

3 In the BodyEnd.inc, you would add Add EmbedTemplBodyEndHook immediately before the < /body > tag:

```
<% Call EnbedTemplBodyEndHook %>
```

Working with templates that use forms

Content Management Server 2001 templates that contain < form > tags in their HTML will conflict with the Web Author's authoring forms when the template is used to author a page. The opening and closing form tags must be masked. After masking, the forms will still work on pages created from that template.

For example, let's look at a template that has the following form elements:

```
<html>
<head></head>
<body>
```

```
<form name="Search" method="post" action="xyz.asp">
Please input search string:
<input type="text">
<input type="submit">
</form>

</body>
</html>
```

You would need to add the following scripting (added scripting is in italics):

```
<html>
<head></head>
<body>

<% If Not IsAuthoringMode Then %>
   <form name="Search" method="post" action="xyz.asp">
<% End If%>

Please input search string:
<input type="text">
<input type="submit">

<% If Not IsAuthoringMode Then %>
   </form>
<% End If%>

</body>
</html>
```

Chapter 8

Working from the Sample Web Author

About customization

The Web Author has a modular structure that makes it easy to customize the user interface, without sifting through endless scripts to find the code you must modify. Using global constants to define the font faces and Cascading Style Sheets (CSS) to define styles makes it easier to make design changes, while ensuring the Web Author is compatible with newer and older Web browsers. This also means you can experiment with layout changes without the risk of breaking any scripts.

Understanding Web Author directory structure

The Web Author ASP files are organized into three directories: Site Interface, Customizables, and Internals.

External files

The Web Author files rely on a set of files within the server files, but are external to the Web Author files.

/NR/system/Access/

- **Resolution.inc:** Resolution.inc is Content Management Server 2001's main include file. Content Management Server 2001 automatically injects Resolution.inc (or Resolution_RT.inc for read-only sites) which in turn includes all include files such as Authenticate. inc.
- **Authenticate.inc:** used on all Content Management Server 2001 ASP files to verify that a user has rights to log on the Web Author, or access particular channels.

Customizable directory

/NR/system/WBC/Customizable/

The Customizable directory has the scripts that control presentation of the Web Author. The graphic-design customization of your site should be done using files in this directory.

The following three files are in the Customizable root directory:

- **WBC.css** contains all the style classes used by Content Management Server 2001.
- **OptionsClient.js:** defines client-side JavaScript values for setting options that affect aspects of the Web Author.
- **OptionsServer.inc:** defines server-side ASP values for setting options that affect aspects of the Web Author.

The Customizable directory contains four subdirectories.

- **SiteEmbeddedWBCLinks**: contains scripts that generate links to switch the site to Edit mode, and generate the authoring console.
- **Hooks**: contains scripts with skeleton procedures and functions that are triggered by a specific event, such as a posting being submitted. You can add custom procedures you want to run every time one of these events occurs on your site.
- **URL Generation Hooks:** contains files for functions that allow custom script to override all URL generation logic in the Web Author.
- **WBCFuncPages Configuration:** contains all the layout constants that control the Web Author dialog ASP pages.

Sub-directories within the Customizable directory:

/NR/system/WBC/Customizable/SiteEmbeddedWBCLinks/

Contains scripts that generate links to switch the site to Edit mode, and generate the authoring console.

- **SiteModeUI.inc:** displays the links that users click to switch between Edit mode and Live mode. What is shipped with Content Management Server 2001 supports only one type of switch using DefaultSiteModeSwitchUI.inc.

- **DefaultSiteModeSwitchUI.inc:** contains the default mode switch shipped with Content Management Server 2001. It is meant to be a sample implementation and can be customized to blend in with the customer site.

  Procedure exposed:
  `Sub ShowSiteModeSwitch()`

- **ConsoleUI.inc:** displays the authoring console site editing links as a group in a console-like user interface. All include files in the SiteEmbeddedWBCLinks directory are referenced by ConsoleUI.inc, except SiteModeUI.inc. What is shipped with Content Management Server 2001 supports only two versions of the console using `WBCFlatHTMLConsole.inc` and `WBCDroplistConsole.inc`.

- **WBCFlatHTMLConsole.inc:** displays a flat version of the console. It is meant as a sample implementation and can be blended with the customer site.

  Procedure exposed:
  `Sub EmbedFlatHTMLConsole()`

- **WBCDroplistConsole.inc:** displays an alternate version of the console. It is meant as a sample implementation and can be blended with the customer site.

  Procedure exposed:
  `Sub EmbedDDroplistConsole()`

- **Site Embedded WBC Links UI String Constants:** contains files that define user interface constants used in generating links Edit mode, Live mode, and the authoring console. The files are centralized here for ease of maintenance.

/NR/system/WBC/Customizable/Hooks/

- **TemplateHTMLHooks.inc:** is where to add your own site-wide custom scripts and hook functions for the head and body tags of templates.
  Functions and Procedures exposed:.

```
Sub OnHTMLHead():
Sub OnHTMLBodyBegin()
Sub OnHTMLFormBegin()
Sub OnHTMLFormEnd()
Sub OnHTMLBodyEnd()
```

- **WorkflowHooks.inc:** contains hook functions called before and after a posting is submitted, approved, or declined. The `OnBeforeXYZ` hook functions allow an operation to be cancelled.

  Functions and Procedures exposed:

```
Function OnBeforeWorkflowOp(pPosting, strAction)
Sub OnAfterWorkflowOp(pPosting, strAction)
```

- **PostingOpHooks.inc:** contains hook functions called before and after a posting is created, saved, or deleted. The `OnBeforeXYZ` hook functions allow an operation to be cancelled.

  Procedures and Functions exposed:

```
Function OnBeforeCreatePosting(pChannel, pTemplate)
Sub OnAfterCreatePosting (pNewPosting)
Function OnBeforeCreateConnectedPosting(pChannel,pTemplate,pExistingPosting)
Sub OnAfterCreatConnected Posting (pExistingPosting, pNewPosting)
Function OnBeforeDeletePosting(pPosting)
Sub OnAfterDeletePosting (pChannel)
Function OnBeforeUpdatePosting(pPosting)
Sub OnAfterUpdatePosting(pPosting)
```

- **ChannelOpHooks.inc:** contains hook functions called just before an existing channel is about to be updated. Returning a a value of False from this function cancels the updating.

```
Function OnBeforeUpdateChannel (pChannel)
Sub OnAfterUpdateChannel (pChannel)
```

- **PlaceholderHooks.inc:** contains hook functions called before and after a placeholder's content is emitted in the template. The `OnBeforeXYZ` hook functions allow the emitting to be cancelled.

  Procedures and Functions exposed:

```
Function ShouldRenderRichEditPlaceholder (pThis, pPlaceholder)
FunctionOnBeforeRenderWBCAuthoringToolbar (pThis, pPlaceholder)
Function OnBeforeEmit(pThis, strPhName, strEmit)
Sub OnAfterEmit(pThis, strPhName, strEmit)
Function OnBeforeUpdatePlaceholder()
Sub OnAfterUpdatePlaceholder()
```

- **ActiveXToolbarHooks.inc:** contains hook functions for the placeholder toolbar attached to the toolbar of the Internet Explorer browser.

  Procedures and Functions exposed:

```
Sub onToolbarInitialize (ByVal pToolbarInterface)
Sub OnToolbarStateInitialize (By Val strPlaceholderName, ByVal pState)
Sub OnToolbarUpdate (ByVal pActiveHtmlEditor, ByVal bEditingSource)
Sub OnToolbarEvent (ByVal pActiveHtmlEditor, ByVal bstrId, ByVal pItem,
By Val bEditingSource)
```

/NR/system/WBC/Customizable/URLGenerationHooks.inc/

- **CommonURLHooks.inc:** contains hook files that allow custom script for customizing or overriding URLs generated for different Web Author modes.

  Procedures and Functions exposed:

```
Sub OnCustomizeWBCEditSiteModeURL( ByVal pThis, ByRef strURL )
Sub OnCustomizeWBCLiveSiteModeURL( ByVal pThis, ByRef strURL )
Sub OnCustomizeWBCAuthoringModeURL( ByVal pThis, ByRef strURL )
Sub OnCustomizeWBCPreviewModeURL( ByVal pThis, ByRef strURL )
```

- **EditSiteOpsHooks.inc:** contains custom scripts for customizing or overriding URLs generated for Web Author Edit Site Mode links for postings.

  Procedures and Functions exposed:

```
Sub OnCustomizeConnectedPostingQuickJumpURL( ByVal pThis, ByRef strURL)
Sub OnCustomizeDeleteURL( ByVal pThis, ByRef strURL )
Sub OnCustomizePostingCopyURL( ByVal pThis, ByRef strURL )
Sub OnCustomizePostingMoveURL( ByVal pThis, ByRef strURL )
Sub OnCustomizeSubmitURL( ByVal pThis, ByRef strURL )
Sub OnCustomizeApproveURL( ByVal pThis, ByRef strURL )
Sub OnCustomizeDeclineURL( ByVal pThis, ByRef strURL )
Sub OnCustomizePageRevisionReportURL( ByVal pThis, ByRef strURL )
```

```
Sub OnCustomizePageRevisionByDateURL( ByVal pThis, ByRef strURL )
Sub OnCustomizePostingPropertiesURL( ByVal pThis, ByRef strURL )
Sub OnCustomizeChannelPropertiesURL( ByVal pThis, ByRef strURL )
```

- **AuthoringModeHooks.inc:** contains custom scripts for customizing or overriding URLs generated for the Web Authoring Mode links.

  Procedures and Functions exposed:
  ```
  Sub OnCustomizeAuthPreviewURL(ByVal, pPosting, ByRef, strlURL)
  Sub OnCustomizeAuthSaveNewDigURL(By Val, pPosting, ByRef, strURL)
  Sub OnCustomizeAuthPreviewConnectedDigURL(ByVal pPosting, ByRef strURL)
  ```

- **PostingCreationHooks.inc:** contains scripts for customizing or overriding URLs generated for the posting creation links in the Web Author's Edit Site mode.

  Procedures and Functions exposed:
  ```
  Sub OnCustomizePostingCreationDlgURL(ByRef strURL)
  Sub OnCustomizeCreatePostingDirectLinkURL(ByVal pChannel, ByVal pTemplate,
  ByRef, strURL)
  ```

- **TaskAssistantHooks.inc:** contains script for customizing or overriding URLs generated for the Task Assistant links in the Web Author's Edit Site mode.

  Procedures and Functions exposed:
  ```
  Sub OnCustomizeProductionManagerURL(ByVal pThis, ByRef strURL)
  Sub OnCustomizeApprovalAssistantURL(ByVal pThis, ByRef strURL)
  ```

- **ResourceManagerHooks.inc:** contains script for customizing or overriding URLs generated for the Web Author's Resource Manager link in Edit mode.

  Procedures and Functions exposed:
  ```
  Sub OnCustomizeResourceManagerURL(ByVal pThis, ByRef strURL)
  ```

/NR/system/WBC/Customizable/WBCFuncPagesConfiguration Files/

- **LayoutConstant:** contains one file called GeneralLayoutConstants.inc for all margin and spacing constants used in all WBCFuncPages.

- **UIStringConstants:** contain all the user interface (UI) strings used in the Web Author. They are centralized here for ease of modification and localization.

- **MLSDisplayConfig:** contains all the display configuration settings used by Web Author dialog boxes to support multilingual display. They can be customized to affect behavior of these Web Author dialog boxes.

Sub-directories within the Internals directory

The Internals Directory is divided in two parts. The first part contains all the internal functions and subroutines used to build the user interface of the Web Author. These are described in this chapter.

The second part consists of scripts that are not meant for customization by site designers and do not directly require the scripts in the Customizable directory. Therefore, they are not documented. Under no circumstances should users modify these files.

/NR/System/WBC/Internals/LinkSupport/

LinkSupport has commonly used server-side procedures that generate URLs needed by scripts in the SiteEmbeddedWBCLinks directory. For example, EditSiteOps.inc returns the URL to the page displaying the Edit mode of a channel or posting.

- **SiteMode.inc:** toggles between Live and Edit modes of the Web Author. It is used by SiteModeUI.inc.

  Functions exposed:
  ```
  Function IsSwitchToEditSiteVisible
  Function IsSwitchToLiveSiteVisible
  Function GetEditSiteURL()
  Function GetLiveSiteURL()
  ```

- **PostingCreation.inc:** supports the construction of links an author clicks to create a page when the site is in Edit mode. The create page links in the sample Web Author are named: “Create New Page,” “Create Connected Page,” and “Select this Template." PostingCreation.inc is used by PostingCreationUI.inc.

  Functions exposed:
  ```
  Function IsNewConnectedPostingLinkVisible
  Function GetCreateConnectedPostingDlgUrl()
  Function IsNewPostingLinkVisible()
  Function GetCreateSimplePostingDlgURL()
  Function GetCreatePostingDirectLinkURL(pChannel, pTemplate)
  ```

- **EditSiteOps.inc:** supports the construction of links available when the site is in Edit mode. It is used by EditSiteOpsUI.inc.

  Functions exposed:

```
Function AreEditOpsVisible()
Function IsConnectedPostingJumpLinkVisible()
Function IsMoveLinkVisible()
Function GetConnectedPostingQuickJumpUrl()
Function GetAuthoringUrl(pPosting)
Function GetDeleteUrl()
Function GetPostingPropertiesUrl()
Function GetChannelPropertiesUrl()
Function GetSubmitURL()
Function GetApproveUrl()
Function GetDeclineUrl()
Function GetPostingRevisionReportUrl()
Function GetPostingHistoricalSiteUrl()
Function GetPostingCopyUrl()
Function GetPostingMoveUrl()
```

- **TasksAssistant.inc:** used by other Web Author scripts to construct Task Assistant links. TasksAssistant.inc is used by TasksAssistantUI.inc.

  Functions exposed**:**

```
Function AreTaskAssistantsVisible()
Function GetApprovalsURL()
Function GetProductionManagerURL()
```

- **ResourceManagement.inc:** supports the construction of Resource Manager links available when the site is in Edit mode. It is used by EditSiteOpsUI.inc

  Functions exposed:

```
Function IsResourceManagementLink Visible()
Function GetResourceMannagerURL()
```

- **AuthCommands.inc**: used by AuthCommandsUI.inc to generate authoring links when the Web Author is in authoring mode.

  Functions exposed:

```
Function AreAuthCmdsVisible()
Function AreAuthCmdsFunctional()
Function IsAuthPreviewConnectedPageVisible()
Function GetAuthPreviewConnectedDlgURL()
Function GetAuthSaveURL()
```

```
Function GetAuthSaveAndExitURL()
Function GetAuthCancelURL()
```

- **LinksSupport.js:** provides client-side JavaScript to support URL links generated by other files in the LinksSupport directory.

  Functions exposed:
  ```
  function WBC_onAuthPreviewPostingWithURL(bFunctional,strPreviewPostingURL)
  function WBC_onAuthSave (bFunctional, strURL, strAction)
  function WBC_onAuthCancel (bFunctional, strURL)
  ```

- **CommonURL.inc:** used by other Web Author scripts to generate commonly used URLs.

  Functions exposed:
  ```
  Function GetWBCLiveSiteModeUrl()
  Function GetWBCEditSiteModeUrl()
  Function GetWBCAuthoringModeUrl (pThis)
  Function GetWBCPreviewModeUrl(pThis)
  ```

/NR/System/WBC/Internals/Shared/

Shared has Server-Side Includes that provide variables and procedures shared by most other files in the Web Author. For example, scripts that return the number of items in a list, or issue messages and alerts, use the variables and procedures in Shared.

- **WBCSite.inc:** contains server-side functions for inserting various Web author components into templates.

  Procedures and Functions exposed
  ```
  Sub EmbedTmplHeadHook()
  Sub EmbedBodyBeginHook()
  Sub EmbedBodyEndHook()
  Function IsTmplHeadHookCalled()
  Function IsBodyBeginHookCalled()
  Function IsBodyEndHookCalled()
  ```

- **WBCCommonShared.inc:** contains server-side functions that are used commonly by ASP pages and Web Author-enabled templates

  Procedures and Functions exposed:
  ```
  Sub EmbedWBCHeadSection()
  Function GetInitializedFormReaderSafe(bUploadValidate)
  ```

- **CommonServer.inc:** defines the global variables and common procedures shared by server-side scripts.

  Procedures and Functions exposed:

```
Function GetCurrentObject
Function IsStagedOrCrawled()
Function IsWBCVisible
Function IsAuthoringMode()
Function IsExistingPosting (pPosting)
Format Date (date Val)
IIf(b, t, f)
Min (x, y)
Max (x, y)
Function AppendQueryString (strBaseUrl, strQueryString)
Sub SetROMPropagationPolicyforWBCMode()
Sub DisableROMPropagationPolicyforWBCMode()
Sub GenerateRoundTripFormElements()
Sub WriteAsNonbreakingBlock(strHTML)
Function Bool2LCaseStr(bASPBoolean)
```

- **CommonClient.js:** shared by all client-side script files, CommonClient.js defines commonly used client-side constants and procedures.

  Functions exposed:

```
function WBC_isIE()
function WBC_isNetscape()
function WBC_getBrowserVersion()
```

  .

- **Windows.js:** contains client-side functions for use in the Web Author that deal with the JavaScript Window object.

  Functions exposed:

```
function WBC-GetFuncWin (strFunc, pTargetWin)
function WBC_openWindowOnFocus (strURL,strWinTarget,strWinFeatures)
function WBC_getPoppedUpWindowFeatures (strWindowFeature, pWindowBasedOn)
```

- **Dates.js**: contains client-side functions for use in the Web Author that deal with dates.

```
function WBC_writeUTC2Local(strDate)
function WBC_convertUTC2Local(pDateUTC)
function WBC_writeUTC2Local(strDate)
function WBC_convertLocal2UTC(pDateLocal)
function WBC_writeDate(pDate)
function WBC_formatDate(pDate)
function WBC_fixJavaScriptYear(nYear )
function WBC_addLeadingZero(nNum)
```

Optimizing performance for run-time sites

This section summarizes the modifications made to Web Author include files to improve performance at run time.

Note
If a file is not listed below, it is supported in a run-time site without modification.

Files with new run-time versions

The following table describes the list of files that also have run-time versions to optimize performance.

Design-time version	Run-time version
Resolution.inc	Resolution_RT.inc
PlaceholdersSupport.inc	PlaceholdersSupport_RT.inc
WBCCommonShared.inc	WBCCommonShared_RT.inc
WBCSite.inc	WBCSite_RT.inc
CommonServer.inc	CommonServer_RT.inc

Several ASP scripting functions have had their behavior modified to suit run-time only sites.

Empty functions

The following ASP scripting functions will be available, but will be stubbed out functions for run-time sites. These modifications ensure compatibility with templates that were Web Author enabled using WBC SiteModeSwitch and/or WBC Console.

- ShowSiteModeSwitch
- EmbedEditSiteConsole
- EmbedSimpleEditConsole
- EmbedDroplistConsole
- EmbedDropDownEditSiteConsole

Note
To functions, IsSwitchToEditSiteVisible and IsSwichToLiveSiteVisible, always return a value of False. The remaining five become empty stubs in run-time sites.

Functions modified for runtime

The following ASP scripting functions have been modfied to omit design-time code:

- EmbedTmpleHeadHook
- EmbedBodyBeginHook
- EmbedBodyEndHook
- EmbedTmplBottom
- EmbedWBCHeadSection
- EmitPlaceholder

Unsupported functions

The ASP scripting functions that follow are not supported in sites configured for run time only.

> **!** Caution
> The scripting functions listed below are not part of the include file structure for run time sites. If your site calls them, a scripting error may result.

- All functions used by SiteMode.UI.inc to toggle between Live and Edit modes of the Web Author: the contents of SiteMode.inc
- All functions used by the Web Author to construct links on the console: the contents of PostingCreation.inc, EditSiteOps.inc, TaskAssistant.inc, ResourceManagement.inc, AuthCommands.inc, LinksSupport.js, and CommonURL.inc
- All Hook functions except PlaceholderHooks.inc and TemplateHTMLHooks.inc
- All option flags in OptionsClient.js used to control Web Author dialog boxes.

About CommonServer_RT_inc

Only the following functions from CommonServer.inc are retained in the run-time version:

- GetCurrent Object()
- IsStagedOrCrawled()
- IsWBCVisible()
- IsExistingPosting(pPosting)
- FormatDate (dateVal)
- IIf(b, t, f)
- Min(x, y)
- Max (x, y)
- AppendQueryString (strBaseUrl, strQueryString)
- IsAuthoringMode()
- Bool2CaseStr(bASPBoolean)

The following functions from CommonServer.inc are not supported in the run-time version: CommonServer_RT.inc: ResetPropateParam()

- ResetPropateParam()
- SetRomPropagationPolicyForWBCMode()
- GenerateRoundTripFormElements()
- WriteAsNonbreakingBlock(strHTML)

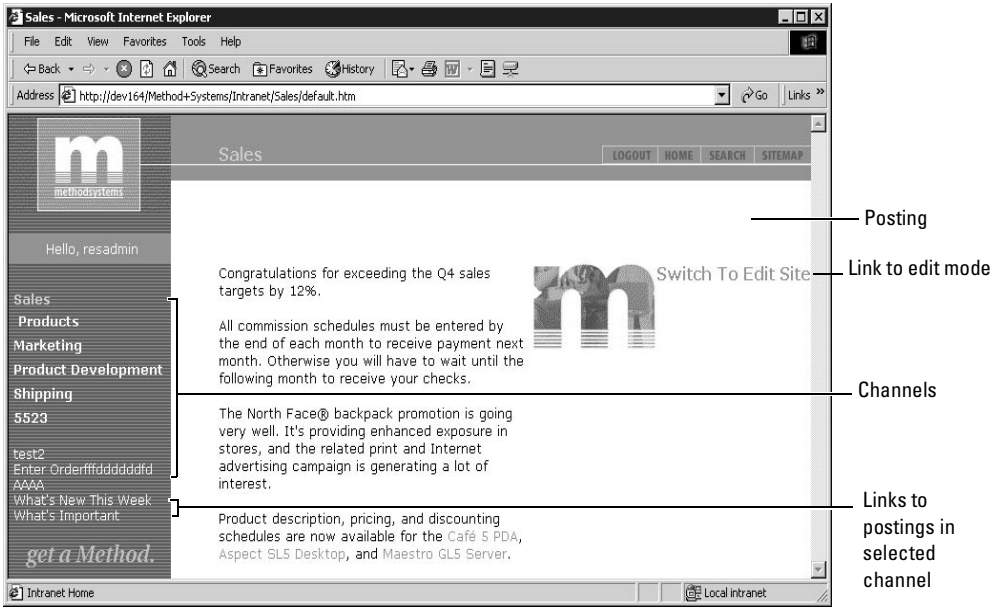
Most constants in CommonServer.inc are removed in the run-time file CommonServer_RT.inc. Only a few essential ones remain. Compare the actual files for details.

Understanding the authoring process

Following is a run-through of the Web Author authoring and approval process. The screen shots are examples only. You can change the location and style of links to suit the needs of your own browser-based authoring.

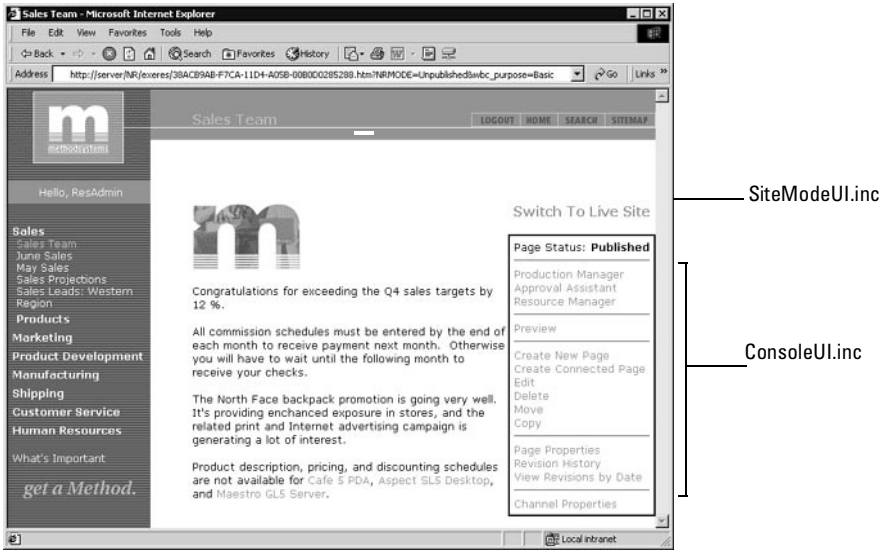
Authoring

1 Once an author logs on to your site, the "live" (run-time) version of the site appears. If the user has authoring rights (authenticated by the `Autosession` object), the "Edit Site" link is also visible.



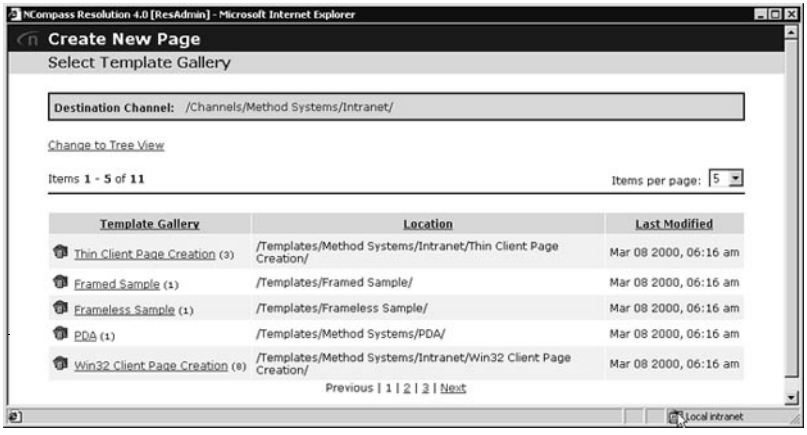
Sample Web Author in Live mode. The current user has authoring rights in this channel.

2 The author clicks the "Switch to Edit Site" link. The Web Author switches to Edit mode. Note that the authoring console shown here is just one way to display the authoring and editing links from the SiteEmbeddedWBCLinks directory (/NR/system/WBC/Customizable/). Alternatively, you could embed the authoring and editing links in a banner cell.



Sample Web Author in Edit mode

3 The author clicks the Create New Page link in the Authoring console. The Template Gallery opens.

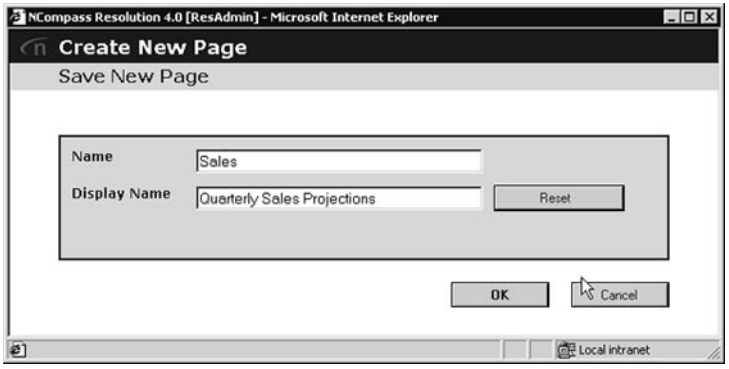


Template galleries in sample Web Author

4 The author selects a template gallery, which opens. The main browser window loads the selected template as a form in authoring mode.

Note
Content and formatting options are determined by the placeholder's properties. A template's placeholder properties are set in the Site Builder. See the *MSCMS 2001 Page Template Designer's Guide* for details.

5 The author adds content to the template's placeholders. When done, the author clicks the Save link. The Save New Page browser window opens.



The Save New Page browser window in sample Web Author

6 The author enters the following information:

- a name for the posting
- a display name.

Note
Display Name defaults to Name if no value is entered. Clicking the Reset button sets the Display Name value to Name.

7 If the author clicks **OK** the posting is Saved. If the author clicks **Submit**, the posting's status changes to Waiting for Editor Approval. If no editor or moderator has been assigned to the channel, the posting is auto-approved. Otherwise, the posting is saved, but not approved.

8 If the author edits the posting after it has been approved by an editor and moderator, the author must resubmit the posting for approval. The authoring console indicates that the posting the author has edited and resubmitted has a "live" (published) version.

Tips for working with postings in the Web Author

- You can review and edit values of existing custom properties using the "Page Properties" option from the Web Author's authoring console, but you cannot create or delete custom properties using the Web Author. Use the Site Builder.
- Custom properties for a posting come from the template on which the posting is based.
- When editing a posting, you have three options to choose from if you have not saved your changes:
  - clicking **Save** from the authoring console saves your changes and keeps you in authoring mode
  - clicking **Save and Exit** saves you changes and takes you out of authoring mode
  - clicking **Exit** takes you out of authoring mode but does not save any unsaved changes. It does not display a confirmation message.
- When authoring a posting, you no longer have access to other console options such as Submit.

- We recommend using "bread crumbs" rather than your browser's "Back" button or keyboard shortcuts to navigate.

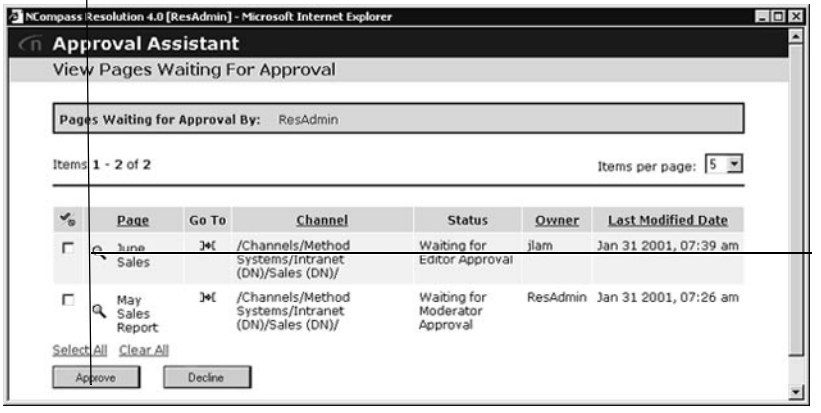
Approving: Editor

1 An editor clicks the Approval Assistant link.



Sample Web Author in Edit mode

2 The Approval Assistant opens, showing postings (pages) submitted to channels where the editor has rights.



Sample Pages Waiting for [Editor] Approval page in sample Web Author

3 The editor can approve or decline the posting. Once approved, its status changes to Waiting for Moderator Approval. If no moderator has been assigned to the channel, the posting is automatically published.

4 If the editor clicks Submit, not Approve, the page waits for the editor's approval. Instead of using the Approval Assistant, an editor can access postings by navigating to them using channel navigation. If the editor has rights, they will see links on the console for Submit, Approve, or Decline the posting.

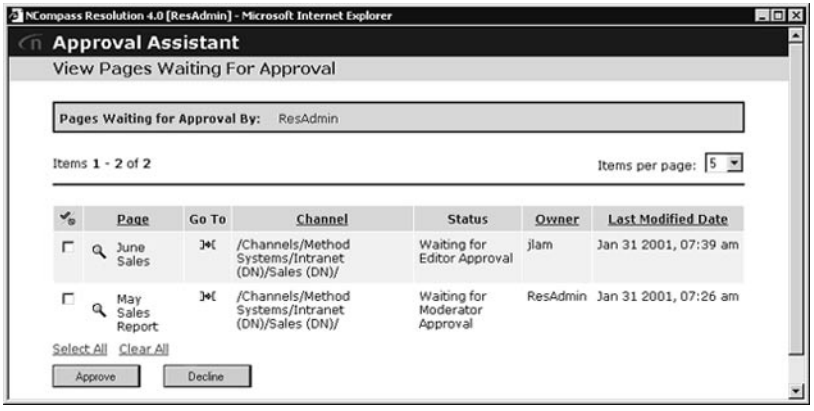
   If the editor edits the posting and submits it, the pages will still await editor approval. Alternatively, the editor can click **Approve** to approve the posting.

Approving: Moderator

If `Authenticate.inc` determines the moderator has moderator rights only (no authoring or editing rights), then `TasksAssistantUI.inc` displays the Approval Assistant link only, as shown in the example below.

1 A moderator clicks the Approval Assistant link.

2 The Approval Assistant opens, showing postings (pages) submitted to channels where the moderator has rights.



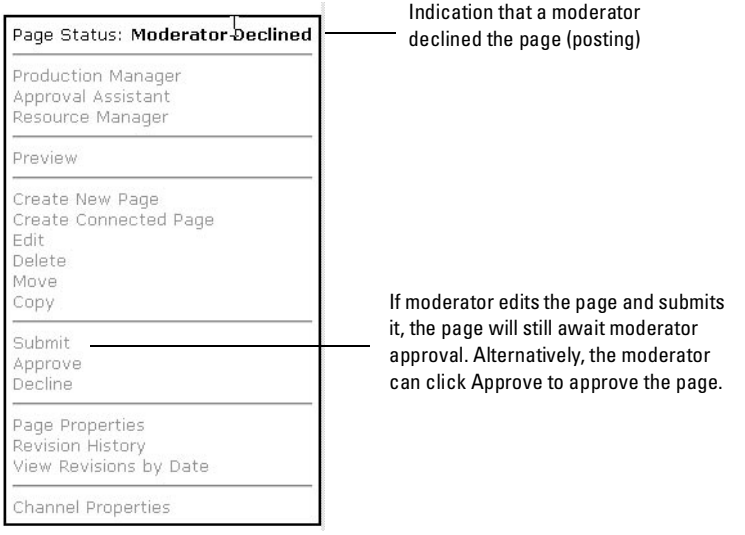
Pages Waiting for [Moderator] Approval page in sample Web Author

3 The moderator can approve or decline the posting. Once approved, its status changes to Approved, and it is available to subscribers as of its publishing start date.

4 If the moderator clicks Submit, not Approve, the page awaits the moderator's approval.

5 If a moderator also has authoring rights, all postings they author and submit to their channel are auto-approved.

6 If a moderator declines the posting, the authoring console indicates this.



Making changes to an Approved posting

If an author changes a published posting's name or description, the posting must be re-approved by an editor. Once re-approved by an editor, the posting is immediately published. It does not need to be re-approved by a moderator.

If an author changes a published posting's start date, expiry date, or priority, the posting must be re-approved by a moderator. It does not need to be re-approved by an editor.

When a published posting is copied into the same channel or a different channel, it reverts to Saved and needs editor and moderator approval before it can be republished.

When a published posting is moved to a different channel (by either an editor or author) it does not revert to Saved and need editor approval. Instead, it goes straight to Waiting for Moderator approval.

States of pages and postings

Publishing states

Postings of Content Management Server 2001 pages can have various publishing states.

Status	Description
New	The current version of the posting has not been stored permanently by the server. The current version may represent an entirely new posting or a new Unapproved version of an existing Approved version. A posting version enters this state when the posting itself is first created, or when a user attempts to modify an existing posting that has only an Approved version.
Saved	The current version of the posting has been stored permanently by the server, but it has not been submitted for approval or approved.
Waiting for Editor Approval	The current version of the posting has been saved and submitted. It is waiting for editor approval.
Editor Declined	The current version of the posting has been saved and submitted, but an editor has declined it.
Waiting for Moderator Approval	The current version of the posting has either been approved by an editor or did not require an editors approval. The posting version is now waiting for approval from a moderator.
Moderator Declined	The current version of the posting has been approved by an editor or did not require editor approval, however it has been declined by a moderator.
Approved	The current version of the posting has been approved by both an editor and a moderator, or it did not require approval by either. The posting object's StartDate is greater than the current date and time, so the posting is not a visible part of the 'Published' site.
Published	The current version of the posting has been approved by both an editor and a moderator, or it did not require approval by one or both. The current date and time is between the posting object's StartDate and ExpiryDate so the posting is visible as part of the 'Published' site.

Status	Description
Expired	The current version of the posting has been approved by both an editor and a moderator, or it did not require approval by one or both. The posting object's ExpiryDate is less than the current date and time so the posting is not a visible part of the 'Published' site.
Deleted	The brief status of a posting between the time it is deleted to the time it is removed from the database. The Deleted status is not be displayed in the Web Author.

Approved and unapproved posting versions

Because Content Management Server 2001 can store previous versions of a posting, a published version of the posting can remain available to people browsing your site while an author, editor, or moderator re-edits the page. In this case, two versions of the page exist: the approved (Published) version, and the unapproved (Waiting for Approval) version. When the unapproved version of the page is approved by an editor, it replaces the old approved version. At that point the page has an approved version only, until the next time it is re-edited.

Chapter 9

Customizing Your MSCMS 2001 Web Author

Customization example #1: re-designing the interface

Using a frameless site as an example, we will see how to re-design the look of the Web Author interface. Our examples assume that you have used the Default Site Mode Switch and the Flat HTML version of the console as the basis for customization.

Customizing the Site Mode link

1 Open DefaultSiteModeSwitchUI.inc located in WBC/CustomizableSiteEmbeddedWBCLinks/WBCSiteModeSwitch/DefaultSiteModeSwitch.

2 Locate the following subroutines: `Sub ShowSwitchToEditSiteLink()` and `SubShowSwitchToLiveSiteLink()`.

3 Customize the links generated in these subroutines.

4 Change the definition of `IDS_UI_SWITCH_TO_EDITSITE` and `IDS_UI_SWITCH_TO_LIVESITE` to an `<img>` tag if required.

These constants are defined in the following subdirectory: WBC/Customizable/SiteEmbeddedWBCLinks/UIStringConstants/SiteModeUIStrings.inc.

5 Save your file. Your changes should appear in your site.



The Switch To Live Site link customized using the ShowSiteModeSwitch () function in DefaultSiteModeSwitch.inc.

Sample Web Author, with redesigned banner and channel includes. The next step is to re-design the authoring console.

Customizing the authoring console

This example assumes that the site designer is using the Flat HTML version of the Web Author console as the basis for customization.

1 Open `WBCFlatHTMLConsole.inc` located in WBC/Customizable/SiteEmbeddedWBCLinks/WBCConsole/FlatHTMLConsole.

2 Locate the subroutine `Embed FlatHTMLConsole()`. It relies on smaller subroutines defined in files in the same directory as helper functions to generate individual console links.

3 Customize the links generated in `EmbedFlatHTMLConsole()` as well as the smaller helper subroutines.

4 Change the definitions of the console link HTML in WBC/Customizable/SiteEmbeddedWBCLinks/UIStringConstants/WBCConsoleUIStrings.inc if required.

5 Save the file. Your changes should appear in your site.



Customized authoring console with EmbedFlatHTMLConsole() in FlatHTMLConsole.inc. See Chapter 8 for details.

Sample Web Author, with redesigned authoring console

Customization example #2: multilingual support

If a template is enabled for a language other than English, this means:

- placeholder objects are fully readable and editable in that language
- all standard authoring and editing functions such as copying, pasting, importing, saving, deleting, previewing, and publishing can be done in a language other than English, if that language is installed on the server
- all aspects of the user interface embedded in the site (for example, Console and SiteMode Switch) and the navigation can be displayed in a language other than English, if that language is installed on the server.

Summarizing the procedure

Enabling new or existing page or navigation templates to support multilingual authoring in both the Site Builder and the Web Author involves the following steps:

- template designers must edit template HTML to set placeholder properties such as the fonts and character sets supported
- authors and editors must use the new DisplayName property in the scripting of navigation links intended to appear in a language other than English. Content Management Server 2001 supports the DisplayName property for postings, channels, and resources.

Language and character sets supported

Content Management Server 2001 currently supports any language or writing system that is part of the Unicode 2.0 standard. Languages and character sets available include the following:

Japanese	Arabic	Thai
Korean (hangul)	Hebrew	Vietnamese
Chinese Traditional	Greek	Turkish
Chinese Simplified	Baltic (Estonian, Lithuanian, Latvian)	
Cyrillic (Russian, Bulgarian, Ukrainian, Serbian)	Eastern European (Czech, Polish, Hungarian, Slovak, Slovenian, Romanian, Croatian)	

Editing template HTML to enable multilingual support

To enable multilingual support on the Web Author, template designers must:

- specify a CodePage directive at the top of the Template ASP
- set the proper Charset value in the HTML < meta > tag. The CodePage directive and Charset meta tag must be inserted directly using the template HTML.
- ensure the font used in the HTL is compatible with the specified language. For example, a Japanese template should not use Arial as the font since it does not support the Japanese character set.

! Caution
If the Codepage directive and the Charset value are pulled in using the include file or generated dynamically through server-side or client-side scripting, the Content Management Server 2001 multilingual support feature will not function properly.

Sample page template scripting

The following is a code sample to enable page template scripting in Japanese. Key lines of code are italicized. Explanatory notes follow at the end.

```
<%@ CodePage=932 %>
<HTML>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=Shift_jis">
<STYLE>
body
   {
   background-color: lightYellow;
   }
   a
   {
   text-decoration: none;
   }
</STYLE>
<TITLE></TITLE>
</HEAD>
<BODY>
<h1><%= Autosession.This.Name %></h1>
<img src="file:///C:/TEMP/ctlA767B720-71C5-11D1-88BE-006097B58A00.gif" height="51"
width="500" alt="Multipurpose">
<br>
<img src="file:///C:/TEMP/ctlA767B720-71C5-11D1-88BE-006097B58A00.gif" height="48"
width="500" alt="SingleImage">
<br>
<img src="file:///C:/TEMP/ctlA767B720-71C5-11D1-88BE-006097B58A00.gif" height="48"
width="500" alt="SingleAttachment">
```

```
<br>
<img src="file:///C:/TEMP/ctlA767B720-71C5-11D1-88BE-006097B58A00.gif" height="51"
width="500" alt="SinglelineText">
</BODY>
</HTML>
```

Notes

- <%@ CodePage=932 %> is the Posting Frame. You can reference it from the frameset by using the `Autosession.This.URLPostingFrame` method.
- If the `<BODY>` tag is defined in an include file, then you must manually insert the following line, which is required for Web Author integration: `Call EmbedBodyBeginHook()`.
- If the `<HEAD>` tag is defined in an include file, you must manually insert the following line, which is required for Web Author integration: `Call EmbedTmplHeadHook()`.

Setting aliases for channels and postings

With Content Management Server 2001, the DisplayName property provides an easier means for enabling multilingual navigation. Content Management Server 2001 supports the use of the DisplayName property for postings, channels, and resources to display text for navigation links. Hierarchical URLs are unaffected by the DisplayName property.

Sample navigation scripting

The following is a code sample to enable navigation scripting in UTF-8. Italics are used to emphasize key lines of code and additional comment lines are provided for reference:

```
<%
Sub PrintChannelNavigation(pThisObject, pChannel, strIndent)
   ' create ASPUtils object which will be used for various misc functions
   ' that are either buggy or not supported in the built-in ASP objects
   Dim pASPUtils
   Set pASPUtils = GetInitializedASPUtils()

   Dim strDisplayName
      ' the display text has to be HTMLEncoded to take care of characters
      ' such as "<" which could cause confusion in the HTML
      strDisplayName = pASPUtils.HTMLEncode(pChannel.DisplayName)

   If pThisObject.GUID <> pChannel.GUID Then
      ' we need to use pASPUtils.MBCSWrite() to write to the client the
      ' content of strDisplayName in order to do the translation from Unicode in
      ' ASP to the specific CodePage necessary for client. In this case, the
      ' client will receive UTF-8 encoded content (65001)
   %>
      <div class="channel"><%= strIndent%><a href="<%= pChannel.URL %>"><% Call
pASPUtils.MBCSWrite( strDisplayName, 65001) %></a></div> <%
   Else
```

```
      ' we need to use pASPUtils.MBCSWrite() to write to the client the
      ' content of strDisplayName in order to do the translation from Unicode in
      ' ASP to the specific CodePage necessary for client. In this case, the
      ' client will receive UTF-8 encoded content (65001)
   %>
      <div class="channel"><%= strIndent%><% Call pASPUtils.MBCSWrite(
strDisplayName, 65001) %></div><%
   End If

   Call PrintChannelPostings(pThisObject, pChannel, strIndent +
"   ")

   Dim pSubchannel
   For Each pSubchannel In pChannel.Channels
      Call PrintChannelNavigation(pThisObject, pSubchannel, strIndent +
"   ")
   Next
End Sub
```

Multinational characters and fonts in channel, posting, and resource names

Template designers, editors, and authors can enter multinational character sets and fonts in the Display Name and Description fields and custom properties for postings, channels, and resources. Content is editable in both the Site Builder and Web Author environments as follows:

	Channels	Postings and resources
Site Builder	Editable to enable the use of multinational font and character sets, but restricted to the operating system language of the client (for example, Japanese on a Japanese operating system).	Editable to enable the use of multinational font and character sets, but restricted to the operating system language of the client (for example, Japanese on a Japanese operating system).
Web Author	Restricted to the language used in navigation template.	Postings are restricted to the language of the template they are based on. Resources can use any multinational characters that are supported in their Web browsers.

We strongly recommend that if you use multilingual characters and fonts in custom properties for templates and channels, you restrict yourself to one language per channel or template. If, for example, you have custom properties in more than one language per channel, it is possible to overwrite correct data with useless data.

If a custom property for a page or channel cannot be edited, the most likely reason is that it is not in the same language as the operating system of the machine being used. For example, the property may have been created on a different machine with an operating system in a different language.

A font set must be installed on the server for it to support serving a Content Management Server 2001 page in that language. For more information, search the Microsoft site (`www.microsoft.com`) under the topic of International Language Support.

For information about Windows 2000, select Control Panel then Regional Options. Choose the appropriate language pack from those listed in the International Language Support section.

Best Practices

Working with font styles for different browsers

To change the font styles for your Web Author interface, you can adjust the Web Author's Cascading Style Sheets (CSS). The CSS are in WBC.css in the Customizable folder.

Site-wide template customization

Code you have written that customizes your Content Management Server 2001 site should go in the `TemplateHTMLHooks.inc` file. The empty `TemplateHTMLHooks.inc` file (WBC/Customizable/Hooks/TemplateHTMLHooks.inc) is specifically to provide a safe place to store custom code. TemplateHTMLHooks.inc, which replaces NR/Shared/ResTmplCustom.inc, is backed up during an upgrade. Migrate code from ResTemplCustom.inc to TemplateHTMLHooks.inc.

Optimization tip

To improve performance, especially on larger Content Management Server 2001 systems, try to reduce the number of API calls you use that retrieve information from the database to create objects.

The following example retrieves three collection objects for each iteration of the For loop. Each collection must be retrieved from the database and sorted. Problems can arise if the contents of the database are changed by other users between the calls.

```
<%
Dim pChannel

Set pChannel = Autosession.ThisChannel

' a new collection is retrieved each time the loop's
' terminating expression is evaluated
For i = 0 to (pChannel.Postings.Count - 1)
   ' each iteration retrieves and sorts two more collection objects
%>
   <P><%= pChannel.Postings.Item(i).Name %>
   <P><%= pChannel.Postings.Item(i).Description %>
<% Next %>
```

To increase performance and avoid problems, store a snapshot of the collection in a variable and use the variable within the For loop. Using VBScript's "For Each (object) In (Collection)" syntax would result in additional performance improvements in this example.

```
<%
Dim strName, pChannel, pPostings, pPosting

Set pChannel = Autosession.ThisChannel

' collection is retrieved only once
Set pPostings = pChannel.Postings

iCount = pPostings.Count
For i = 0 to (iCount - 1)
   Set pPosting = pPostings.Item(i)
%>
   <P><%= pPosting.Name%>
   <P><%= pPosting.Description%>
<% Next%>
```

Handling Publishing API Errors

The server is a multi-user system that has access to several network resources, including a database server such as Microsoft SQL Server and a Microsoft IIS Web server.

Examples of actions that cause errors include trying to:

- view Content Management Server 2001 objects the current user does not have rights to
- view Content Management Server 2001 pages before logging in
- change read-only properties
- change read/write properties while not in the Autosession 'Update' mode
- access or set properties of objects that have been deleted.

The Publishing API provides two important tools for handling errors: the Autosession object's SuppressExceptions property, and the Error object. The Autosession.SuppressExceptions property determines whether the server Publishing API COM objects will raise COM exceptions when it encounters an error during the current ASP session. The Error object is used in some circumstances (described later on) to access information about whether an error has occurred, and if so, what the cause is.

Errors when using VBScript

We recommend that if you are using VBScript as the server-side scripting language in your templates and server ASP files, you should allow the server to raise COM exceptions by setting `Autosession.SuppressExceptions` to false.

COM exceptions are handled with VBScript's built-in Err object. After Content Management Server 2001 encounters an error, the Number and Description properties of the VB Err object will contain, respectively, the identification number and description of the error that occurred. The description will sometimes include information about the particular parameter that caused the problem. The standard 'On Error Resume Next' statement will prevent the errors from halting the script. The Err object is silently reset and control will pass to the next statement in your ASP. The Autosession Error is still set and can still be used to get internal details when debugging.

Note that you need to convert VBScript's Err.Number property by performing a bitwise AND with the hexadecimal value &H0000FFFF before comparing its numerical value with the Error.ID values described in the *MSCMS 2001 Publishing API Help*.

```
<%
Dim pError ' Autosession's error object
Dim lErrorID ' for Autosession's Error ID
Dim lErrorNumber' for VB Err's Number
Dim lErrorNumberConv ' for Err.ID value after a bitwise AND

On Error Resume Next

' ... set properties that throw errors

Set pError = Autosesion.LastError

' get the Autosession's Error ID
lErrorID = pError.ID

' get the built-in VB Err number
lErrorNumber = Err.Number

' after bitwise And, it should be equivalent to the Error.ID
lErrorNumberConv = lErrorNumber And &HOOOOFFFF

' lErrorNumberConv is equal to lErrorID
%>
```

Errors when using JavaScript

With earlier versions of the Microsoft JavaScript engine shipped with version 4 of IIS, server-side scripts often stopped after encountering a COM exception. If you are using the older JavaScript engine, you may want to suppress the server's COM exceptions by setting the `Autosession.SuppressExceptions` to true. When errors are encountered the server will not raise a COM exception.

This is where the Content Management Server 2001 Error object will be of use: although exceptions are not raised, the Content Management Server 2001 Error object will still contain information about any errors that occurred. This gives JavaScript programmers a way to check whether an operation was successful before proceeding, even when COM exceptions are suppressed.

The Autosession.LastError property returns an Error object that will contain the ID, Description, and InternalDetails properties of the last error. If an error is found, remember to reset the Autosession's record of the last error by calling `Autosession.ClearLastError`.

```
var pError;
// change a read-write property
pPosting.Name = "New Name";
pError = Autosession.LastError;

if (pError.ID == 0)
{
   pPosting.Commit();
} else {
   Autosession.ClearLastError();
   // ... any error-handling you wish to perform
}

// change a second read-write property
pPosting.Description = "New description...";
pError = Autosession.LastError;
// last error (if any) will not prevent this
// second property from being committed
if (pError.ID == 0)
{
   pPosting.Commit();
} else {
   Autosession.ClearLastError();
   // ... any error-handling you wish to perform
}
```

One last issue to note is that the Autosession.RollBackOnSessionEnd property, regardless of whether it is set to true or false, will roll back all uncommitted changes from the current session if the last error caused by a Content Management Server 2001 object has not been handled. Your scripts should always handle errors, but in case you forget, the API will make sure the database is not compromised by trying to commit any failed changes.

Handle all errors using either the Err object or the Content Management Server 2001 Error object (the latter followed by a call to `Autosession.ClearLastError`) to prevent losing your changes when you have set the RollBackOnSessionEnd property to false.

Changing, updating, and committing objects

Changing Read/Write properties

There are several read/write properties for Posting and Placeholder objects that make it possible to write ASP-based Web Authors with the API. Content creators can make new postings and edit placeholder content from a Web browser.

An outline of the steps for changing a read/write property include

1. Making sure you are in the Autosession Update mode. Modes persist for the lifetime of the session, and are, in most cases, automatically propagated from one session to the next. If you are not in the correct mode, you must explicitly get the query string for a new posting or channel in Update mode, and then provide a link to an ASP script using that query string (or redirect the user). See the *MSCMS 2001 Publishing API Help* for the QueryStringModeUpdate property for more information about switching to the Autosession 'Update' mode. If you are not in the correct mode, you get an error when you try to set a read/write property.
2. Get a variable that references the object you want to modify. This reflects the properties and values of information stored in the database. The Autosession mode will, in some cases, partly determine which version of the object is returned. The user's rights will also influence the version that is returned.
3. Determine the new value you want to set the property to. The sources for this information include: user-defined form fields; databases; information returned by another application.
4. Set the object's read/write property to the new value.
5. Check for errors using either the VB Err object, or the Content Management Server 2001 Error object (see the topic on error handling for more information about when to use each of these).
6. Either commit or roll back the changes to permanently store them in the database.

Committing the changes

You commit changes using the Autosession.CommitAll method which causes the server to update the information in the database to match the property changes made in ASP. `Autosession.CommitAll` commits changes to all objects that have been modified in the current session.

Rolling back the changes

An alternative to committing the changes is to roll them back. This might be necessary if one or more errors occurred, and you wanted to undo all changes that were made.

For example, if you changed the StartDate successfully, but resetting the ExpiryDate failed, you would probably want to return both values to their original state rather than committing only one of the changes to the database.

`Autosession.Rollback All` undoes changes made to the ASP object since the last time `Autosession.CommitAll` was called. `Autosession.RollbackAll` rolls back changes to all objects that have been modified in the current session.

Automatically committing or rolling back changes

Content Management Server 2001 provides a way of automatically either committing or rolling back changes at the end of each ASP session. The Autosession.RollbackOnSessionEnd property, which has an implicit default value of false, determines whether uncommitted changes are kept or discarded.

If `Autosession.RollbackOnSessionEnd` is false, Content Management Server 2001 will commit all changed objects at the end of each session. The only circumstance that prevents it from committing changes is if there was an unhandled error. If there are unhandled errors, all changed objects are rolled back to their last committed state, regardless of the RollbackOnSessionEnd property value.

If `Autosession.RollbackOnSessionEnd` is true, all uncommitted changes are rolled back and discarded at the end of the session.

In most sites, the RollbackOnSessionEnd property should be set in one of the include files referenced by all Content Management Server 2001 templates and server files on your site such as the TemplateHTMLHooks.inc file.

Whether or not you set `RollbackOnSessionEnd` to true or false will depend on how explicit you want your site's programmer to be when modifying Content Management Server 2001 objects. If uncommitted changes are rolled back at the end of the session, you force template programmers to explicitly commit their changes. If uncommitted changes are committed at the end of the session, template programmers don't have to explicitly commit changes, with the built-in safeguard that if unhandled errors have occurred, Content Management Server 2001 will roll back the changes rather than commit them.

Chapter 10

Developing for MSCMS 2001 Site Stager

MSCMS 2001 Site Stager compatibility

Site Stager is an application for creating a static HTML version of a dynamic Content Management Server 2001 site, including content, navigation, and links.

If your Content Management Server 2001 system will be staged at any point, consider the following when creating or modifying navigation templates or modifying system resource files.

- In all ASP script files used by Content Management Server 2001, the `Autosession` object's `ResolveURL()` method must be wrapped around all non-Content Management Server 2001 references to resources to be staged. Examples of these resources are style sheets, linked scripts, ASP files, Java Applets, ActiveX objects, and other HTML pages.



Note
If an HTML file contains a link to a resource that needs to be staged, the HTML file must be converted to an ASP file first, and then `ResolveURL()` must be put around the link to make it "stageable." If this is not done, Site Stager ignores the link and the resource pointed to is not staged.

- Do not apply `ResolveURL()` to links to targets external to the Content Management Server 2001 system that contain one or more links. If `ResolveURL` is applied, Site Stager stages the target of that link but any links in the staged target itself won't work. Not applying `ResolveURL()` to the link allows users to follow the link to a 'live' version of the target, complete with functioning links.
- Site Stager saves the output HTML pages generated by the server-side scripts, but without special modifications, the server-side scripts themselves are saved as part of the generated files.
- To successfully stage the product of a script that contains a mode check, `Or Autosession.IsModeStaging` must be added to the mode check. For example:

```
If Autosession.IsModePublished Or Autosession.IsModeStaging Then
   ' execute for both runtime and staging modes...
End If
```



Note
See the end of this chapter for samples of scripting for staging.

- To stage ActiveX objects or Java applets referenced in Content Management Server 2001 templates and server-side scripts, the objects or classes must be packaged in .cab files (for ActiveX objects) or .jar files (for Java Applets). In the case of server-side scripts, `ResolveURL()`must be applied to the references, unless the files are stored in a resource gallery. Note that .jar files stored in a resource gallery only function if the name of the file is the same as the name of its initial class.

Examples of using external .cab and .jar references with ResolveURL()

The HTML code must be converted to ASP for `ResolveURL()` to be effective.

The external .cab file reference is as follows:

```
<object id="UserControl1"
classid="CLSID:FD68B14F-442C-11D1-9B8B-DAFE8593593D"
codebase="<%=Autosession.ResolveURL _
("somedir/Demo1.CAB#version=1,0,0,0")%>"
align="baseline" border="0" width="349" height="231">
</object>
```

The external .jar file reference is as follows:

```
<applet code="PieChart.class" archive="
<%=Autosession.ResolveURL _
("somedir/survey.jar")%>"
width="300" height="300">
     <param name=excellent value=
        "<%= g_GradeArray(1,2) %>">
     <param name=good value="<%= g_GradeArray(2,2) %>">
     <param name=fair value="<%= g_GradeArray(3,2) %>">
     <param name=poor value="<%= g_GradeArray(4,2) %>">
</applet>
```

- To prevent staging of dynamic Content Management Server 2001 resources, such as Subscription pages, help pages, query (Whatsnew.asp), and login pages that no longer function once they are staged, any reference to these should be contained within the following check:

  ```
  If Not Autosession.IsModeStaging Then
     ‘ execute for non-staging modes only
   End If
  ```

  The `If Not` allows the following ASP scripting to execute in modes other than staging. This provides for the functionality of these dynamic Content Management Server 2001 resources in the context of non-staged sites. Other navigation items, such as the Help and Login/Logout buttons, can be disabled from staged sites using this conditional statement.

- Any references to URLs not on the same Web server as the server must be fully qualified, and the URLs must be accessible from the server on which the staged site is deployed. An example of a fully qualified URL is:

  ```
  <a href=”http://www.microsoft.com/default.asp”>
  ```

- When saving a template open in an HTML editor, it imports some local resources specified in the HTML file into the Content Management Server 2001 system. The following list of tags are resource types that get imported:

  As value of the ARCHIVE attribute: APPLET, OBJECT
  As value of the BACKGROUND attribute: BODY
  As value of the CODEBASE attribute: OBJECT
  As value of the CODE attribute: APPLET
  As value of the DATA attribute: OBJECT
  As value of the OBJECT attribute: APPLET
  As value of the SRC attribute: IMG

- Those resources that are not imported are left as is. If links to them are not fully qualified URLs, they might not be accessible after staging. In addition, the final staged site's HTTP server must have access to the targets of these links.

- For successful staging, navigation templates must not contain any code that creates an Autosession object using `CreateObject()`; rely instead on the Resolution.inc file to supply an Autosession object to work with.

Site Stager redirection

If you use Response.Redirect to redirect browsers, the URL to which you are redirecting should not be a relative URL. Rather, it should use a full path, such as /files/recent/headline.html. Redirecting to relative URLs will lock the Site Stager in an infinite loop.

If redirecting to URLs returned by properties, such as `Posting.URL`, you must manually modify the URL before redirecting to it. In the Staging Autosession mode, Content Management Server 2001 wraps staging tags (< NCOMPASSSTAGINGSERVER > ... < NCOMPASSSTAGINGSERVER >) around all URLs it generates. The files included with the server include a VBScript function to strip off staging tags. The StripOffStagingTags function in the Server\IIS_NR\Shared\Template.inc file can be used to make generated URLs compatible with redirects. The following example shows how this is done.

```
=' Removes <NCOMPASSSTAGINGSERVER> tags from urls. These tags appear
' in staging mode. This function is most commonly used to remove
' staging tags from redirection URLs in Staging mode.
' (this function is from the intranet_common.inc file)
Function StripOffStagingTags(strUrl)
   StripOffStagingTags = Replace(Replace(strUrl, "<NCOMPASSSTAGINGSERVER>", ""),
"</NCOMPASSSTAGINGSERVER>", "")
End Function
```

```
' check mode - if Autosession.IsModeStaging is true, you must
' modify header response redirects in the following manner:
If Autosession.IsModeStaging Then
   Dim pPosting
   Dim strURLWithStagingTags
   Dim strURLNoStagingTags

   ' set pPosting to the posting you want to redirect to
   ' ...

   ' get URL - in Autosession staging mode, generated
   ' URLs are wrapped in staging tags
   strURLWithStagingTags = pPosting.URL

   strURLNoStagingTags = StripOffStagingTags(strURLWithStagingTags)

   ' redirect to cleaned-up URL
   Call Response.Redirect(strURLNoStagingTags)
End If
```

Sample script for staging

The template designer uses the following tags to mark up script that will be staged. There are a pair of tags created for staging script, and a pair of tags for staging include statements and the files to which they refer. There is one tag used to specify the suffix of staged output files.

```
<!-- RESX STAGE_SCRIPT -->
<!-- RESX RUN_AND_STAGE_SCRIPT LANG="VBSCRIPT" -->
<!-- RESX STAGE_INCLUDE -->
<!-- RESX RUN_AND_STAGE_INCLUDE -->
<!-- RESX STAGE_EXT="" -->
```

Tag: <!-- RESX STAGE_SCRIPT --><!-- /RESX STAGE_SCRIPT -->

Usage

```
<!-- RESX STAGE_SCRIPT -->
[script content to be staged]
<!-- /RESX STAGE_SCRIPT -->
```

Sample

The template source is as follows.

```
<%
'This stuff is not staged.
Response.Write "Testing..."
%>
<HTML>
<HEAD></HEAD>
<BODY>
<!-- RESX STAGE_SCRIPT -->
<%
Response.Write "Hello World"
%>
<!-- /RESX STAGE_SCRIPT -->
</BODY>
</HTML>
```

The appearance in staged script is as follows.

```
Testing...
<HTML>
<HEAD></HEAD>
<BODY>
<%
Response.Write "Hello World"
%>
</BODY>
</HTML>
```

Explanation

Anything between the tags is copied intact to the staged output. There is no validation of the content between the tags so you can put anything between the tags.

Be aware of the following in regards to staged script.

- The default scripting language of the template must be VBScript. By default this is the case unless a script-directive (`<%@ LANGUAGE=JScript %>`) is placed at the top of the script.
- Case is important when specifying the tag.
- The staged script should not access the MSCMS 2001 Publishing API, assuming the site is staged to a non-server.
- The default extension of the staged file is ".asp" unless specified otherwise (see the `<!-- RESX STAGE_EXT = "" -->` tag explanation).
- Staging tags and script to be staged cannot be nested inside a set of scripting delimiters. The following is an example of bad script.

```
<HTML>
<HEAD></HEAD>
<BODY>
<%

<!-- RESX STAGE_SCRIPT -->
<%
Response.Write "Hello World"
%>
<!-- /RESX STAGE_SCRIPT -->

%>
</BODY>
</HTML>
```

- Enclose the script to be staged in a pair of scripting delimiters.
- The scripting tags are not validated. An error in specifying the tags could result in none of the tagged content making it to the staged file.

Tag: <!-- RESX RUN_AND_STAGE_SCRIPT LANG="VBSCRIPT" --> <!-- /RESX RUN_AND_STAGE_SCRIPT -->

Usage

```
<!-- RESX RUN_AND_STAGE_SCRIPT LANG="VBSCRIPT" -->
<!-- /RESX RUN_AND_STAGE_SCRIPT -->
```

Sample

The template source is as follows.

```
<%
Response.Write "Hello World 1"
%>
<!-- RESX RUN_AND_STAGE_SCRIPT LANG="VBSCRIPT" -->
<%
   Response.Write "Hello World 2"
%>
<!-- /RESX RUN_AND_STAGE_SCRIPT-->
```

This is how the script appears when in ASP as run by the server.

```
<%
Response.Write "Hello World 1"
%>
<%
   If Autosession.IsModePublishing Then
      Response.Write "Hello World 2"
   End If
%>
```

This is the appearance in staged script.

```
Hello World 1
<%
   Response.Write "Hello World 2"
%>
```

Explanation

Does the same thing as `<!-- RESX STAGE_SCRIPT-->` and also leaves a copy of the script in the template to be executed during publishing mode.

Notes

- The LANG attribute is mandatory.
- Currently VBScript is the only supported language, which means, in order to have script run and be staged, it must be in VBScript.
- In the staged file, any output from the run-time version of the script appears after the script in the staged file.
- The script must be enclosed in scripting delimiters for it to be executed properly during publishing mode.
- Currently, there is error reporting if an unsupported language is specified. See notes for `<!-- RESX STAGE_SCRIPT -->`
- Any error in specification of the tag results in the tag and its contents not appearing in the staged output.
- The default scripting language of the template must be VBScript. By default this is the case unless a script-directive (< %@ LANGUAGE = JScript % >) is placed at the top of the script.
- The script's output shows up in Published mode only. That is, it will not show up in previews of any page based on the template.

Tag: <!-- RESX STAGE_INCLUDE --><!-- /RESX STAGE_INCLUDE -->

Usage

```
<!-- RESX STAGE_INCLUDE -->
[include statement(s) to be staged]
<!-- /RESX STAGE_INCLUDE -->
```

Sample

The template source is as follows.

```
<!-- RESX STAGE_INCLUDE -->
<!-- #include virtual="/includes/inc1.inc" -->
<!-- /RESX STAGE_INCLUDE -->
<HTML>
<HEAD></HEAD>
<BODY></BODY>
</HTML>
```

The appearance of the script in ASP as run by the server is as follows.

```
<HTML>
<HEAD></HEAD>
<BODY></BODY>
</HTML>
```

The format of the staged include statement depends on whether the “Store Resources Separately” option is selected or not. If this option is selected, and a virtual directory “/resources” is specified on the destination system, then the appearance of the staged script is as follows.

```
<!-- #include virtual="/resources/inc1.inc" -->
<HTML>
<HEAD></HEAD>
<BODY></BODY>
</HTML>
```

If the "Store Resources Separately" option is not selected, the appearance of the staged script is as follows.

```
<!-- #include file="inc1.inc" -->
<HTML>
<HEAD></HEAD>
<BODY></BODY>
</HTML>
```

Explanation

Used to stage #include files. To stage #include files, do not place the #include statements with the script to be staged. The script to be staged should be enclosed by script staging tags, and the include should be marked up with its own tag. When the #include statement is staged, the #include statement itself will be rewritten depending on whether resources are stored separately from the pages or not.

- If the "Store Resources Separately" option is select, all include files will be stored in the resources directory and the #include file will be written as `<!-- #include virtual=".." -->` with the virtual directory as set by the user.
- If the "Store Resources Separately" option is not selected, the include files will be stored next to the first staged file that included it. The #include statement will be written as `<!-- #include file=".." -->` and the path is a relative path to the #include file.

Notes

- Multiple #include statements can be placed between the tags.
- The tags disappear from the staged content.
- The suffix of the include file should not be .asp or .jsp because only the output of the script will be staged, not the script itself. It should be .inc to ensure success in staging the entire contents of the #include file.
- The path to the #include file must be an absolute path without any server name, for example "/includes/inc1.inc".
- The path to the #include file is assumed to be a valid path on the server being staged
- The #include statement to be staged must be of the format `<!-- #include virtual=".." -->` as the format `<!-- #include file=".." -->` will not work with Site Stager.

Tag: <!-- RESX RUN_AND_STAGE_INCLUDE --><!-- /RESX RUN_AND_STAGE_INCLUDE -->

Usage

```
<!-- RESX RUN_AND_STAGE_INCLUDE -->
[ include statement]
<!-- /RESX RUN_AND_STAGE_INCLUDE -->
```

Sample

The template source is as follows.

```
<!-- RESX RUN_AND_STAGE_INCLUDE -->
<!-- #include virtual="/includes/inc1.inc" -->
<!-- /RESX RUN_AND_STAGE_INCLUDE -->
<HTML>
<HEAD>
</HEAD>
<BODY>
</BODY>
</HTML>
```

The appearance in ASP as run by the server is as follows.

```
<!-- #include virtual="/includes/inc1.inc" -->
<HTML>
<HEAD>
</HEAD>
<BODY>
</BODY>
</HTML>
```

The appearance in staged script is as follows.

```
<!-- #include <virtual or file>="<url staged and resolved>" -->
[any possible output from /includes/inc1.inc]
<HTML>
<HEAD>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Explanation

Places the #include statement in the staged file and leaves a copy in the template for processing during publishing mode. See the notes for `<!-- RESX STAGE_INCLUDE -->`.

Tag: <!-- RESX STAGE_EXT="" -->

Usage

`<!-- RESX STAGE_EXT="<ext>" -->` where `<ext>` is the extension the posting based on this template is to be staged under. The extension should not contain the '.' character.

Sample

The template source is as follows.

```
<!-- RESX STAGE_EXT="cfm" -->
<HTML>
<HEAD></HEAD>
<BODY>
</BODY>
</HTML>
```

Effect

Any postings based on this template are staged with a suffix of "cfm." If the name of the posting was "p1," the posting is staged as "p1.cfm."

Explanation

Use this tag to specify the suffix of the file that the posting will be staged to. This tag is inserted in the template to effect this change during staging and will not show up in the content during publishing mode. To specify an extension on a per-posting basis and not a per-template basis, insert the sequence `<br><!-- STAGE_EXT="ext" -->` in the HTML source of a placeholder. Be aware however, that the tag appears in the content during publishing mode.

Keep in mind the following:

- This tag disappears from the staged content.
- By default, if a template contains scripted marked for staging then the staged file is a .asp file by default. If a JSP include is found then its extension is .jsp. If an ASP include is found then its extension is .asp. If the STAGE_EXT tag is found then it overrides the other values.

Tips for better scripting

1. Limited use of API values in staged script

While the staged script cannot make use of the Publishing API, there is a way to stage numerical and string properties retrieved from the API so they can be used in the staged script.

The technique is to output script that consists of assigning API literal values to variables. These variables can then be used inside the staged script. Following is an example of the template source using this technique.

```
<!-- RESX STAGE_SCRIPT -->
<%
Dim username
username = <!-- /RESX STAGE_SCRIPT -->
<%
Response.Write """" + Autosession.Username + """"
%>
<!-- RESX STAGE_SCRIPT -->
Response.Write username
%>
<!-- /RESX STAGE_SCRIPT -->
```

The staged output is as follows.

```
<%
Dim username
username = "WinNT://NCOMPASS-LABS/RESADMIN"
Response.Write username
%>
```

2. Specifying Extensions on a per-page basis

The `<!-- RESX STAGE_EXT="ext"-->` tag is normally used in template source but can also be inserted in placeholder content. To do this, the author must be able to modify the HTML source of the placeholder and insert the sequence `<br><!-- RESX STAGE_EXT="ext" -->`. The < br > tag ensures that the < resx > tag is not stripped from the HTML source when it is saved.

Chapter 11

Implementing Full-Text Search

Setting up full-text search

Industry-standard Web and intranet indexing software packages such as Microsoft Site Server are designed to produce a local index of your site. Instructions for setting up search engines to index Content Management Server 2001 sites are provided in the *MSCMS 2001 Site Administrator's Guide*. This chapter covers:

- adding scripting to Content Management Server 2001 files and templates to allow Web crawlers to index a Content Management Server 2001 site
- linking from a Content Management Server 2001 file or template to the index generated by the search engine.

Tip
For more information on setting up full text search, see the *MSCMS 2001 Site Administrator's Guide*, the Content Management Server 2001 site (www.microsoft.com/cmserver), and your product's reference manuals.

Supported search engines

Content Management Server 2001 sites have been tested with Microsoft Site Server version 3.

Redirection

For a Web crawler to be able to index a page, it must get to that page through a series of hyperlinks, which originate ultimately from the designated root URL(s). Sometimes when a user is browsing a site, the browser is redirected to an alternate page when trying to access a given URL. For the Web crawler to follow this same redirection, there are some special considerations. (Note that the following items are true for any Web site, not just a site hosted by Content Management Server 2001.)

HTTP Header Redirection

If the browser is redirected using HTTP header redirection, the Web crawler can follow the link to the new URL. When using ASP pages, this type of redirection is achieved by using the Response object's Redirect method.

```
<%
response.redirect ("/ct/redirected.html")
%>
```

Script-based redirection

Another type of redirection is client-side, script-based redirection. In this approach, the Web browser executes a script block that contains instructions to redirect to another location. The Web crawler usually can't follow these redirections.

If script-based redirection is necessary, use empty anchor tags (as shown) to make the Web crawler aware of the target of the redirect.

```
<A HREF ="http://servername/ct/redirected.html"></A>
```

Web browsers do not display anything for an empty anchor tag, but Web crawlers read the tag and index the indicated URL in their crawls.

Tip
Check the Web crawler's log files to see which URLs are being disqualified due to configuration settings.

Setting the title of a page template

Most search engines display the title of the pages they index. For framed and frameless sites, this title is created by the page template's <TITLE> tags. To use the posting name as the document's title, the Autosession object can be used within the page template:

```
<HTML>
<HEAD>
   <TITLE><%=Autosession.ThisPosting.DisplayName%></TITLE>
</HEAD>
```

Considerations for framed sites

The Web crawler's default setting is to index all pages that it can access. We recommend you exclude the navigation frames from the index. The <META> tag can be used to exclude these frames from indexing, while allowing the Web crawler to follow any links they may contain.

```
   <META NAME="robots" CONTENT="noindex,follow">
```

This tag can be generated automatically for framed channels by specifying the appropriate options from the Properties dialog box.

Web crawlers index and create links to the individual frames within a frameset. Anyone using the search engine can then select a link to an individual frame. This causes the frame to load in the entire browser window, outside of its original frameset. This behavior occurs when any framed site is indexed by a Web crawler, and is not specific to a Content Management Server 2001 framed site. The sample searching scripts supplied with Content Management Server 2001 solve this problem. The sample scripts are discussed later in this chapter.

Enabling Web robot access

Note
Not available on frameless channels.

By selecting check boxes on the Publishing tab of the channel or posting dialog box you can specify the following:

- whether a channel or a posting should be indexed by Web robots
- whether hyperlinks in a channel or a posting should be followed by Web robots

To enable Web robot access from within the Site Builder:

1 Access the Properties dialog box by one of the following methods:
   - Right-click a channel or posting and select **Properties**
   - Select a channel or posting and click the **Properties** button
   - Select a channel or posting and then choose **File > Properties**.

2 Click the **Publishing** tab.

3 Set the "Web Robots Can Crawl Links" box to checked

4 Uncheck the "Web Robots Can Index This Channel's Navigation" check box.

5 Click **OK**.

The Hidden flag

If a channel or posting has been marked as hidden by checking the 'Hide When Published' check box, this means the channel or posting is hidden from the navigation in the Web Author. This does not mean that your ability to access the given channel or posting is restricted in published mode.

For example, an author can create a hyperlink to the hidden posting from another posting or channel, and, by following this hyperlink, the posting or channel can be accessed. Similarly, if a site user knows the URL of the 'hidden' item and they have rights to view it, they can navigate to it.

The hidden flag should not be used to prevent users from viewing a channel or posting. If you do need to prevent users from viewing, then change their user rights.

Using consistent URLs

If the Web crawler is configured to filter out URLs from external sites, you must ensure that the hyperlinks to pages on the internal site are specified with consistent URLs. This is especially important when a server uses more than one name.

As an example, if the name of the server you are indexing is `http://foo/`, then use the same name consistently in your URLs, or alternatively don't use the server name at all. If the Web crawler finds a link starting with `http://www.foobar.com/` (an alternate name for the same server), it is considered a different site and is not indexed.

Linking to the search page

Your Content Management Server 2001 software contains a navigation template that allows full text searching using Microsoft Site Server. The template named Search includes Search.inc, which is in the following directory on your server:

```
/NR/Method SystemsSupport/Includes/Search/Search.inc
```

Chapter 12

Content Connector Directory Structure and Framework

Overview

Microsoft Content Management Server 2001 Content Connector for Commerce Server 2000 is an optional feature of Microsoft Content Management Server 2001. This integration of content management and the personalization capabilities of Commerce Server 2000 gives users the ability to create a seamless environment for building and maintaining personalized Web sites.

Commerce Server handles the back-end infrastructure like credit card transactions, shipping considerations, and inventory management, while Content Management Server 2001 delivers Web site design, self-service authoring, template-based publishing, and site management. The combination of these two products means users can cross-reference editorial content with the e-commerce portion of Commerce server in a dynamic content-driven site.

The next few chapters in this guide outline the main building blocks of the Content Connector feature.

Understanding Content Connector directory structure

Content Connector builds on both the Commerce Server 2000 site directories and the Microsoft Content Management 2001 Web Author directories.

Site files

When a Content Connector site is unpacked, some directories are created in the virtual directory that is created for the site under the **Inetpub\wwwroot** folder. The following are significant subdirectories:

Subdirectory	Description
Include	These include files define the main functions that are included and called by the sample site's templates.
SolutionSite	These include files encapsulate Commerce Server 2000 Retail site functionality used by the Content Connector sample site.
Unpack	This contains all the code that will be unpacked to the wwwroot file when the site is moved. In the case of the shipped sample site, this includes the global.asa that Content Connector requires in wwwroot.
WBC_Customizable	This is the sample site's version of **<MCMS 2001 server>\httpexec\WBC\Customizable** and is mapped in IIS as the WBC Customizable virtual directory during site unpacking. If any changes have been made to a pre-existing site's MSCMS 2001 Web Author console, or if any changes are required to the console that ships with the sample site, this is where they are made. See the section "Customizing the authoring console" in Chapter 9, "Customizing Your MSCMS 2001 Web Author" in this guide for information on how to modify the Web Author console.

Web Author Extensions

In the Web Author directories, Content Connector installation adds the following files under the **< MCMS 2001 server > \IIS_NR\ContentConnector\CS2000\Framework** directory.

The framework directory contains a few ASP files used for click tracking and including the framework in templates and ASP files. It also contains the following subdirectories:

Subdirectoriy	Description
WBCExtensions	Contains some ASP files like WBCLauncher.asp (called from the Business Desk to launch MSCMS 2001 when creating, editing, or deleting pages).
Classes	ASP files defining the classes of the Framework API.
ChannelPicker	A Web application used to choose an MSCMS 2001 channel for creating a profiled posting from the Business Desk. Used for creating profiled postings from the Business Desk.
Customizable	ASP files used to define the constants used in Web Author extensions included in the Content Connector sample site console.
Include	A style sheet used in the sample site, plus the include file for Web Author extensions in the sample site.
PageProfile	A Web application that allows you to change the Content Connector page profile. Note that this is an extension to the Web Author page properties application.
ResolutionLogin	This directory contains the Web application for doing the MSCMS 2001 login action from the Business Desk.
TemplatePicker	A Web application used to choose an MSCMS 2001 template for creating a rich product posting from the Business Desk or Web Author. This is called when a product category has specified a template gallery instead of a single template for creating products of that category. See Chapter 14, "Creating and Mapping Content Connector Containers" in the MSCMS 2001 Site Administrator's Guide for more information.

Understanding the Content Connector Framework API

The Content Connector Framework API is an intermediate function module that links the functionality of Commerce Server 2000 and Content Management Server 2001. Through the Framework API, the application developer can implement Content Connector's main functionality. This is done by including some Framework API class files that are also ASP files into the Content Connector ASP files or MSCMS 2001 templates that are calling the objects' properties and methods.

Important Framework API objects, properties, and methods

For a complete listing of Content Connector Framework API objects, refer to the *MSCMS 2001 Content Connector for Commerce Server 2000 Framework API Help* file. The following is a listing of important objects and their properties and methods.

NRCSSession object—an object extending the MSCMS 2001 Autosession object

Important properties	Brief description
ThisPosting	Get the current framework posting object.
ThisCategory	Return the NRCSCategory object from the URL query string or for the current product posting.
ThisCatalog	Return the NRCSCatalog object from the URL query string or from the current product posting.
ProductContainerRoot	Used to get the NRCSProductContainerRoot object.
CatalogRootChannel	This property uses the path in the Content Connector resource for this site to find the channel where catalog channels and rich product postings will be created by default.

Important methods	Brief description
SelectContent	This method returns a SimpleList of any NRCSProfiledPostings by executing the personalized content object (PCO) pipeline.

NRCSPosting object—an object that is used to deal with the framework postings.

Important properties	Brief description
Posting	Returns the MSCMS Publishing API Posting for this NRCSPosting object.
URL	Returns the URL to the Publishing API Posting for this NRCSPosting.
IsProductPosting	Used to verify what type of framework posting this is.
IsRichProductPosting	
IsSimpleProductPosting	
IsProfiledPosting	

NRCSProfiledPosting object—postings that have a Commerce Server 2000 profile. All NRCSPosting properties are supported, plus some additional properties.

Important properties	Brief description
ContentProfile	Returns the Commerce Server profile for this NRCSProfiledPosting

Including the Framework API

To use all the Framework API components, you include a file named ConnectorApp_include.asp which is located under `<MSCMS 2001 server>\IIS_NR\ContentConnector\CS2000\Framework`. This file instantiates the framework API session for the current page.

Make sure global.asa includes connector global initialization files and functions.

Chapter 13

Personalizing a Content Connector Site

Personalization

There are three ways to target content within a Content Connector site. They correspond to the Content Management Server 2001 objects Posting, Placeholder, and Channel.

The most important personalization feature gives users links to the most appropriate pages for their current context and profile. This is posting-level personalization and is done using personalized content objects, which is a Content Connector feature.

Another way to provide users with content that is appropriate to them based on their profile is through page fragment targeting. This is placeholder-level personalization because it happens within a page and it requires user profiling provided by Commerce Server 2000 in addition to template scripting in Content Management Server 2001.

Users can be provided with access to secure content that is not publicly available. This is done in Content Management Server 2001 through controlling access to the channels that users are allowed to see. See the section "Personalization using MSCMS 2001 security" in this chapter for more information.

Using personalized content objects

The most powerful method of personalization in Content Connector is the personalized content object. It combines the content management capabilities of Content Management Server 2001 with the content selection framework of Commerce Server 2000 and is made available in Content Connector. See Chapter 3, "Personalizing Your Site Content" in the *MSCMS 2001 Content Connector for Commerce Server 2000 Business Desk User's Guide* for more information on personalized content objects.

The site programmer is responsible for displaying personalized content and to:

- Enable authors to create profiled pages that can be used both as a source of personalized content and as a target for context-based rules.
- Agree with the business manager on a partition of the site that makes sense of the site objectives.
- Embed personalized content objects in templates for pages which require personalized content to appear.
- Display the results of invoking the personalized content object.

Creating profiled pages

To target pages to particular contexts or users, the Content Management Server 2001 pages must have a profile. For more information on how to create profiles and how to author pages using profiles, see Chapter 2, "Creating Page Profiles" in the *MSCMS 2001 Content Connector for Commerce Server 2000 Business Desk User's Guide*. This section explains how the site programmer can use a template to create profiled pages.

A profiled page is one created from a template with the following custom property:

```
b_AllowsContentProfile:TRUE
```

Page groups and page positions

For personalization rules to be effective, the business manager and site programmer must agree to a partition of the site. For example, in the Method Systems sample site, the home page contains two personalized content object pipelines. The first returns a list of product specials appropriate to the current user's product preference and the second returns a list of new articles.

The site programmer does not need to know what the rules are for selecting which product pages or news articles will be in the list, but they do need to know that the first is a list of product pages and the second is a list of new articles. The business manager does not need to know how the specials or new articles are going to be displayed, but they do need to know that both specials and articles are going to appear on the home page. This intersection of knowledge is expressed as page groups and page positions.

The business manager defines which personalized content object rules apply for which page groups and positions. The site programmer specifies a page group and position when requesting selected content from the personalized content object pipeline.

Embedding a personalized content object in a template

The site programmer needs to invoke a personalized content object in a template. To get a list of Framework ProfiledPostings, pass the appropriate PageGroup and PagePosition for the page as arguments to SelectContent.

```
Set listNRCSProfiledPostings = g_pNRCSSession.SelectContent _
   (strPageGroup, strPagePosition, nMaxPostingsToReturn)
```

If the site programmer needs to sort the list of postings or filter them in some way, they can do so by using the NRCSPersonalizedContentObjectSelector object instead of the session's SelectContent method.

Sorting the returned postings

Normally, the list of postings returned by the SelectContent method is sorted first by their score and second by their start date. Refer to the *MSCMS 2001 Framework API Help* for more information about using this property.

```
pNRCSPersonalizedContentObjectSelector.Sort
```

Filtering the returned postings

Normally, the list of postings returned by the SelectContent method is not filtered and all postings that match are returned. Refer to the *MSCMS 2001 Framework API Help* for more information about using this property.

```
pNRCSPersonalizedContentObjectSelector.Filter
```

Displaying personalized content object results

Once the list of postings is obtained, the template script needs to display them. How this is done will depend on what the content is expected to be. For example, on the home page of the MethodSystems sample site, a list of product specials is obtained and only the first one is displayed. On the products page, the same list is obtained, but all of the returned postings are displayed.

Product special—home page

The following script displays a single product posting in detail. Note that it relies on knowing that the posting being returned by the personalized content object is a rich product posting. On this site rich product postings all have a placeholder with a known name that contains an image of the product.

```
'get one product posting from PCO
```

```
Set listNRCSProductPostings = g_pNRCSSession.SelectContent (strPageGroup,
strPagePosition, 1)
Set pNRCSProductPosting = listNRCSProductPostings.Item(0)
Set pPosting = pNRCSProductPosting.Posting

'Display the product's image
Set colPlaceholders = pPosting.Placeholders
For Each pPlaceholder in colPlaceholders
   If pPlaceholder.Name = IDS_PH_PRODUCT_IMAGE Then
      Set pPlaceholderMatched = pPlaceholder
      Exit For
   End If
Next
strProductImageURL = pPlaceholderMatched.XImageSRC

'Display the Commerce Server information for the product
Set pProduct = pNRCSProductPosting.Product
Set rsProperties = pProduct.GetProductProperties

'Iterate through each property
For Each fldProperty In rsProperties.Fields
   Call CS_RenderProductPropertiesInSpec (fldProperty.Name, fldProperty.Value)
Next
```

Product special—product page

The following script displays links to multiple product postings returned by the same personalized content object as the previous example. Note that this script relies on the same naming convention as the previous script.

```
'Get a list of postings from PCO
Set listNRCSProductPostings = g_pNRCSSession.SelectContent_
   (strPageGroup, strPagePosition, nMaxPostingsToReturn)

<table>
For Each pNRCSProductPosting in listNRCSProductPostings

   'Get the product image
   For Each pPlaceholder in pNRCSProductPosting.posting.Placeholders
      If pPlaceholder.Name = IDS_PH_PRODUCT_IMAGE Then
         Set pPlaceholderMatched = pPlaceholder
         Exit For
      End If
   Next
   strImageURL =
   CS_AppendQueryStringToURL (pPlaceholderMatched.XImageSRC,
```

```
   "thumbnail=true&maxsize=" & ID_THUMBNAIL_SIZE)

   <tr><td>
   <img src="<%=strImageURL%>">
   </td><td>
   <a href="<pNRCSProductPosting.URL%>"
   <%=pNRCSProductPosting.Displayname%></a>
   </td></tr>
Next
</table>
```

Page fragment targeting

By checking a property on the current user's profile, you can personalize the content inside a template. An example of this in the Method Systems sample site is in the rich product template where particular placeholders are targeted at various "partner" types.

The following code displays content of a placeholder when the login Commerce Server user is a Government Partner type:

```
<% If CS_ShowPartnerContent (IDS_PARTNER_TYPE_GOVERNMENT) Then%>
   <img src="placeholder" height= "150" width="300" alt="Government Content">
<% End If%>
```



Note

The img src placeholder is obtained by dragging a placeholder from the Controls gallery while editing a template in an HTML editor. For more information, see "The template Design Palette" in Chapter 3, "Creating a Frameless MSCMS 2001 Interface" in this guide.

Targeting does not have to be confined to targeting a single placeholder. It is possible to create entire sections of templates that are only visible to users with certain properties in their profiles.

Personalization using MSCMS 2001 security

Without Content Connector and Commerce Server 2000, Content Management Server 2001 allows some limited forms of personalization. These are based on the security model and allow site programmers to design the site so that only certain users are allowed to see certain parts of the site.

The first method for doing this is to set up subscriber rights groups that have exclusive access to some channels in the site. For an example of this, see the OEM News channel in the Method Systems site. Any news articles placed in that channel will only be visible to members of the CS2000 OEM Subscribers rights group.

For more elaborate personalization based on channels, you can create channels based on interests and share postings across those channels. (To share postings across channels, use "Connected Postings." With Connected Postings, a single user can have access to multiple channels. An example of this personalization method would be if you had a posting called "Hiking with Children." People with access to the "Hiking" channel and the "Camping with Children" channel would both see that posting.

Chapter 14

Accessing the Content Connector Product Catalog

Accessing the product catalog

The Commerce Server 2000 product catalog can be accessed through an interface provided with that product. We have extended the capabilities of that interface and integrated it with the Content Management Server 2001 Publishing API within the Content Connector Framework API.

Managing the product catalog

For more information about how to use the product catalog in your site, see Chapter 4, "Managing the Product Catalog" in the *MSCMS 2001 Business Desk User's Guide*.

Product catalog interfaces

The main framework objects involved in accessing the product catalog are as follows:

NRCSProductContainerRoot—a top level object that contains all the catalogs in the site.

Important properties	Brief description
ProductContainerChildren	Use this property to get a list of all the catalogs in the site.

NRCSCatalog—represents a Commerce Server catalog.

Important properties	Brief description
Catalog	Use this property to get the Commerce Server Catalog for a framework catalog.
ProductContainerChildren	Use this property to get the root categories for the catalog.

NRCSCategory—represents a Commerce Server product category.

Important properties	Brief description
ProductPostings	Use this property to get the product postings for any product in this product container.
Category	Use this property to get the Commerce Server Category for a framework category.

Important properties	Brief description
Catalog	Use this property to get the Commerce Server Product Catalog to which the corresponding Commerce Server Category for this framework category belongs.
ProductContainerChildren	Use this property to get the categories that are children of this category.

NRCSSimpleProductPosting—a single posting that is equivalent to product.asp in a Commerce Server 2000 site. A simple product posting can be used to display any product that does not have a rich product posting associated with it.

Important properties	Brief description
Catalog	Use this property to get the Commerce Server Catalog for the current product.
Product	Use this property to get the Commerce Server product object for the current product.
RichProductChannel, RichProductTemplate	Use these properties to get the designated channel and template to create a rich posting for the current product.

NRCSRichProductPosting—a posting that represents a single product in a Commerce Server catalog.

Important properties	Brief description
Catalog	Use this property to get the Commerce Server Catalog for the current product.
Product	Use this property to get the Commerce Server product object for the current product.

Creating product templates

This section outlines how site programmers can create the necessary templates for simple and rich product pages.

Creating simple product pages

A simple product page is one created from a template with the following custom property:

```
TemplateType: SimpleProduct
```

Creating rich product pages

Rich product pages are created from a template with the following custom properties:

```
TemplateType: RichProduct
b_AllowsEditInBrowser: TRUE
b_AllowsContentProfile: TRUE
```

Scripting product templates

To create either simple or rich product templates, do the following:

- Include the Framework
- Access the following Commerce Server API objects
  - Product Catalog: `g_pNRCSSession.ThisCatalog.Catalog`
  - Product Category: `g_NRCSSession.ThisCategory.Category`
  - Product: `g_pNRCSSession.ThisPosting.Product`

Example script for displaying product specificaton

```
Set pProduct = pNRCSProductPosting.Product
Set rsProperties = pProduct.GetProductProperties

' Iterate through each property
For Each fldProperty In rsProperties.Fields
   Call CS_RenderProductPropertiesInSpec (lfdProperty.Name, fldProperty.Value)
Next
```

Scripting product navigation

In the Method Systems sample site, product navigation is done through the catalog hierarchy. Catalog navigation is not necessary, but it is useful because products are sometimes in multiple categories and because there may be products that do not have an MSCMS 2001 page assigned to them.

To script catalog navigation, you should:

- Include the Framework
- Access the following Framework API objects:
  - Framework Catalog: `Set pNRCSCatalog = g_pNRCSSession.ThisCatalog`
  - Framework Category: `Set pNRCSCategory = g_pNRCSSession.ThisCategory`

Once you have the catalog and category of the current product you can generate links to other pages of interest using the following properties and methods.

To get this	Use this
Parent category or list of parent categories	`Set listNRCSContainerParents = pNRCSCAtegory.ProductContainerParents`
List of catalogs	`Set rsCatalogs = mscsUserCatalogs ()`
List of children categories for the current category	`Set listNRCSChildrenCategories = pNRCSCategory.ProductContainerChildren`
List of root products for the current catalog	`Set listNRCSProductPostings = pNRCSCatalog.ProductPostings`
List of children products for the current category	`Set List NRCSProductPostings = pNRCSCategory.ProductPostings`

Listing Product Items

Once you have obtained a list of products, you can access each of them and get specific properites when creating a page.

```
For each pNRCSProductPosting in listNRCSProductPostings

   Set pProduct = pNRCSProductPosting.Product

   If pNRCSProductPosting.IsRichProductPosting Then
      strName = pNRCSProductPosting.Posting.DisplayName
   Else
      strName =
pProduct.GetProductProperties.Fields(IDS_PRODUCT_PROPERTY_NAME).Value
   End If
```

```
   strDescription =
pProduct.GetProductProperties.Fields(IDS_PRODUCT_PROPERTY_DESCRIPTION).Value
   StrPrice =
pProduct.GetProductProperties.Fields(IDS_PRODUCT_PROPERTY_LIST_PRICE).Value

Next
```

Product Search

Products can be accessed through a search function as well as by navigation through a category hierarchy. There is an example of this in the Method Systems sample site.

To search for a product on a Commerce Server product catalog and translate the result to direct links to product pages:

Use `MSCSCatalogManager.FreeTextSearch` to perform a product search.

Convert ADO recordset result to a list of Framework postings using Framework API method `g_pNRCSFactory.GetPostingsByProductRS`.

Chapter 15

Accessing Other Commerce Server 2000 Objects

Accessing other Commerce Server 2000 objects

Commerce Server support pages and processing scripts

MSCMS 2001 Content Connector sites express Commerce Server functionality like product search, supporting pages like the Basket page, and processing scripts like basket operations. Any of these can be implemented in a stand-alone MSCMS 2001 page or as an external ASP script. The advantage of implementing them as MSCMS 2001 pages is that site navigation scripts are easier to generate and less dependent on static URLs that may break.

The general method for implementing a site function as an MSCMS 2001 page is to build a template with the necessary logic and create a single page for it. The page may contain user interface elements or just be a script page that does some function. If the page is for a processing purpose, such as adding an item to a basket, ensure users cannot get or navigate to it other than through the intended triggered action. If the page does not contain placeholders, it is not necessary to set it so that it cannot be edited, but it is recommended that the following custom property be set:

```
b_AllowsEditInBrowser: FALSE
```

In the Method Systems sample site, product search is embedded in the search result MSCMS 2001 page. All non-UI pages such as product ordering operations are implemented as external ASPs. Most UI pages such as the basket page are implemented as MSCMS 2001 pages. The sample site does not give examples of some of the functions available in the Commerce Server Retail Site, such as order tracking. These functions could be handled as either external ASPs or as MSCMS 2001 pages, depending on site requirements.

Analyzing Content Connector Sites

The main Content Connector function that site managers may want to analyze is the personalized content object (PCO) pipeline. To tune the PCO rules, you must know which rules were successful in getting users to click through to the pages that were being suggested to them. This can be done through the Commerce Server Analysis module in the Business Desk. This is similar to the sample reports created for analyzing campaigns. The information is written by Content Connector to the site logs so analysis can be done.

Content Connector does not use the Commerce Server 2000 pipelines to log events exactly as it does for campaigns. The campaigns **redir.asp** uses the RecordEvent pipeline to record events in a special way to the IIS log, but that pipeline is too specialized to campaign items for handling a pipeline serving personalized content.

Since PCOs serve up links to other pages on the site, the IIS log will contain a normal entry for the request, for example:

```
/NR/ContentConnector/CS2000/Framework/ClickTracking_redir.asp?NRCS_PostingID={123-
ABC-
```

```
...}&NRCS_PostingName=Article&NRCS_Score=1.2&NRCS_PageGroup=News+Summary&NRCS_PagePo
sition=Headlines&NRCS_Url=%2fCS2000%2fNews%2fArticle%2ehtm
```

The PCOs do not need to use the explicit event recording pipeline because the above URL appears in the log and the user is redirected to the posting itself, for example, `"/CS2000/News/Article.htm"`

In the properties of the DTS Web log import task you can add references to the query parameters of interest. For example: `NRCS_PostingId, NRCs_PostingName, NRCS_Score, NRCS_PageGroup, NRCS_PagePosition, NRCS_Url.`

When the log is imported there are entries in the data warehouse query string table that can be used to write report queries about the Names, URLs, Scores, GUIDs, and for things that users have clicked on in the site.

Integrating Content Management 2001 authentication with user profiles

For an explanation of how the authentication system works in the Method Systems sample site, see Chapter 13, "Managing Content Connector User Authentication" in the *MSCMS 2001 Site Administrator's Guide*.

The Method Systems sample site has an example of how to reconcile Content Management 2001 authentication with Commerce Server 2000 SQL-based user profiles. The example provides limited security for profiles of authenticated users, but it is expected that most important data will be protected by MSCMS 2001 security and the user's Active Directory password.

Securing profile passwords

Site programmers may change the generation of the profile password to something more secure. To do this, re-implement the functions:

`cs_FormatProfilePasswordForResolution(string_user)` and `cs_FormatProfileNameForResoltuion(string_user)`.

The argument to both functions is the login name for the authenticated Content Management Server 2001 user that you generate a login id and password for. The current algorithm will give them a modified user name and a standard password. The sample site authentication sequence uses these functions to create corresponding SQL-based profiles for each MSCMS 2001 authenticated user. Our login page does not allow users to log on to the site using the generated user name and password. They must use the Active Directory user name and password to access this generated profile.

appendix A

Query String Parameters

General query string parameters

The following is a summary of the query string parameters that affect how a page is generated in MSCMS 2001. These parameters are passed in the query string to the template.asp.

Five name/value pairs are passed as MSCMS 2001 Publishing API or Web Author generated query string parameters:

- NRMODE: {Published, Unpublished, Update, Staging}
- NRHISTORICALTIME {Some date/time value}
- nr_emit: {View (default), ThinEdit, Win32Edit}
- wbc_purpose: {Basic, Author, PreviewAuthor}
- wbc_acceptcontent: {None (default), PostProps}}

MSCMS 2001 Publishing API generated query strings

This is not an exhaustive list of the query strings generated and used by the MSCMS 2001 Publishing API. It is the list of query string parameters used specifically for the Web Author.

NRMode

Indicates the MSCMS 2001 Publishing API's Autosession mode. It controls the mode the API will operate in for the current ASP page.

- Published: Live mode
- Unpublished: the state of the Web Author after the user clicks the Edit Site link
- Update: the MSCMS 2001 is updated. For example, the Web Author is in Update mode when a user changes a posting's name.
- Staging: the ASP is being run for the purpose of staging.

Script should use API methods to determine the mode instead of examining the query string directly. Use the following Publishing API methods to check the mode:

- Autosession.IsModePublished
- Autosession.IsModeUnpublished
- Autosession.IsModeStaging
- Autosession.IsModeUpdate.

See the *MSCMS 2001 Publishing API Help* for more information.

NR_HISTORICALTIME

Indicates any expression that can be interpreted as a date. It can also be a value of data type "Double" that represents a date, as a result of using a conversion function such as CDbl() to convert a Date value in VBScript.

When specified, this parameter renders the page at the time specified. This is what you would see if you preview a specific page at a specific time in the past. It is currently supported only for postings.

- numbers that look like dates
- strings that look like dates
- dates returned from functions.

A date expression is limited to numbers or strings in any combination that can represent a date from January 1, 100 to December 31, 9999.

Dates are stored as part of a real number. Values to the left of the decimal represent the date; values to the right represent the time.

This query string is used by the MSCMS 2001 Publishing API to automatically determine the object `Autosession.This` should return. When the query string is present, `Autosession.This` will return the corresponding historical version of the object. Script should not examine this query string directly.

Web Author generated query string parameters

nr_emit

Indicates the HTML to generate for a placeholder in various modes.

- View (default): generate placeholder content viewing (read-only)
- ThinEdit: represent placeholders as HTML 3.2 form elements when the Web Author is in Authoring mode
- Win32Edit: represent a placeholder in the Site Builder's Authoring window.

Script should use the following constants defined for the query string name and values.

- Const IDS_QPARAM_NREMIT = "nr_emit"
- Const IDS_EMIT_VIEW = "View"
- Const IDS_EMIT_THINEDIT = "ThinEdit"
- Const IDS_EMIT_WIN32EDIT = "Win32Edit"

wbc_purpose

Indicates the Authoring Console's visibility and mode. These properties are added to all generated URLs using the Autosession.PropagateParameter method.

- Basic: the site is in Edit mode
- Authoring: the page is now in authoring mode.
- PreviewAuthoring: the template is previewed in Web Author authoring mode from the Site Builder's Authoring window.

Script should use the following constants defined for the query string name and values.

- Const IDS_QPARAM_WBCPURPOSE = "wbc_purpose"
- Const IDS_WBCPURPOSE_BASIC = "Basic"
- Const IDS_WBCPURPOSE_AUTHORING = "Authoring"
- Const IDS_WBCPURPOSE_PREVIEWAUTHORING = "PreviewAuthoring"

wbc_acceptcontent

Indicates what content should be set on the posting object.

- None (default): no properties or content to set
- Phs: setting placeholder content

Script should use the following constants defined for the query string name and values.

- Const IDS_QPARAM_WBCACCEPT = "wbc_acceptcontent"
- Const IDS_ACCEPTCONTENT_PHS = "Phs"

Common UI scenarios and query strings

The following table summarizes the common UI scenarios in the Web Author and the corresponding query string parameter values for each case.

UI Scenario	NRMODE	NR_ HISTORICAL TIME	NR_EMIT	WBC_PURPOSE	WBC_ ACCEPT CONTENT
Run time (Live mode)	Published	-	-	-	-
Site Builder's Authoring Window	Unpublished	-	Win32Edit	-	-
Preview of template in Site Builder's Authoring window	Unpublished	-	ThinEdit	PreviewAuthoring	-
Preview of posting in Web Author (authoring Console is not displayed)	Unpublished	-	-	-	-
Web Author is in Edit mode	Unpublished	-	-	Basic	-
Preview of a template selected for creating a new page	Unpublished	-	-		-

UI Scenario	NRMODE	NR_ HISTORICAL TIME	NR_EMIT	WBC_PURPOSE	WBC_ ACCEPT CONTENT
Web Author is in Authoring mode (author is authoring a page)	Unpublished	-	ThinEdit	Authoring	-
Preview of page the author is authoring in Web Author	Unpublished	-	-	-	Phs
Viewing of versioned content	Published/ Unpublished	Some date value	Any of the possible values	-	-

Glossary

Note that comprehensive glossaries of Microsoft terminology can be viewed at the following Web site: `www.msdn.microsoft.com/siteguide/glossaries.asp`.

A

Administrator, Content Management Server 2001

The Content Management Server 2001 (MSCMS 2001) administrator's job is to set up and manage publishing in the MSCMS 2001 environment. Administrators have full privileges (including the right to create, edit, and approve pages and postings) in all MSCMS 2001 containers.

Administrator, Windows 2000

A Windows 2000 administrator is a user with administrative rights on the computer where the MSCMS 2001 server and database are installed. It is typically the responsibility of the Windows 2000 administrator to install and configure Content Management Server 2001, and set up user accounts. The Windows 2000 administrator will not necessarily be the same person functioning as a Content Management Server 2001 administrator.

API – Application Programming Interface

In the same way that most applications will have a user interface (often a graphical one), many applications will also present an Application Programming Interface (API) with which other applications can interact. The API consists of the functions, messages, data structures, data types, and statements that programmers can use in creating applications that run under Windows 2000.

Approval Assistant

In the MSCMS 2001 Site Builder only. The Approval Assistant is a feature, accessible with the Approval Assistant button on the MSCMS 2001 window toolbar, that generates a list of all postings, pages, or templates waiting approval.

(See also *Production Manager*.)

approve, approval

Publishing Content Management Server 2001 pages follows a two-part approval process. Pages are approved by editors for accuracy of content. Postings for the views of these pages are approved by moderators for relevancy to the channels to which they are posted.

(See also *auto-approve*.)

authentication page

A Web page you might need to pass through in order to log on to a Microsoft® Content Management Server 2001 site with a Web browser. On the authentication page, you will need to select a valid 2000 domain and enter your user name and password, or an LDAP (Lightweight Directory Access Protocol) Organizational Unit (OU).

author

Content Management Server 2001 authors create pages in folders, and submit postings in channels. The role of author is one of the formal roles in Content Management Server 2001.

(See also *role*.)

Authoring window

In the MSCMS 2001 Site Builder only. Pages are authored in the MSCMS 2001 Authoring window, which displays a view or views of a page, to which you add content such as text or images.

auto-approve

When an author submits a page to a folder that does not have an editor, Content Management Server 2001 automatically approves the page. Pages are also automatically approved when submitted by editors for the folder, or by administrators. Postings for approved pages are auto-approved (on an individual basis) if they are:

- created by an administrator, or created in a channel where the creator has moderator rights
- created in a channel where no moderator is configured.

(See also *approve, approval*.)

B

business manager, Content Connector

The business manager is responsible for creating page profiles, personalized content objects, and for managing rich content created for Web sites.

C

channel

In typical implementations of Content Management Server 2001, the channel structure forms the navigational architecture of the Web site and can be thought of as a site map. A channel is designed to control both how the browsing user navigates through the site and how these users are restricted from accessing various parts of the site, depending on their authorization level.

container

Virtual storage spaces (folders, channels, or galleries) used to organize pages, postings, resources, and templates. Pages are stored in folders; postings are stored in channels; resources and templates are stored in galleries.

content

Text, an image, or any embeddable item that can be used to fill the placeholders in a page.

content expressions

Content expressions are used in Content Connector. They are a combination of properties, Boolean and relational operators, and values that determine if a page profile is selected or not. Content expressions are combined to make content groups.

(See also *content groups*.)

content groups

Content groups are used in Content Connector. They are objects that define and score by relevance a set of page profiles that share some common properties. Personalized content objects specify one content group to define the set of page profiles to be returned by the personalized content object pipeline. Content groups are created by combining content expressions and scoring them.

(See also *content expressions*.)

cover page

Refers only to Content Management Server 2001 channels that use frames. The cover page is the page that appears when a subscriber clicks the channel name in their Web browser. A cover page typically displays a brief outline of the type of information posted to the channel. For frameless channels, the channel navigation template is displayed when the user clicks a channel link, which then takes the place of a cover page.

D

decline

The alternative to approving a page or posting is to decline it. Editors and moderators can decline pages or postings that are inaccurate or otherwise unsuitable for the channel they are intended to be viewed on. Declining changes the status of the page or posting from "Waiting" to "Declined." An author can then make changes, and resubmit.

default content

Default content is an image, or text, or a combination of images and text that a page template designer has added to a placeholder. This indicates to authors what type of content the

placeholder should be filled with. For example, in a placeholder designated to display the title for a page, the page template designer might enter "Place title here" as the placeholder's default content.

Dependant Report

(See *Report, Dependant*.)

description

Descriptions should be entered in the Properties dialog box of each page, folder, channel, gallery, template, and resource. Descriptions are displayed in ToolTips and are valuable for providing quick but meaningful identification of these items and their intended uses.

destination directory, staging

The destination directory is the directory that receives the HTML or ASP "snapshots" of your Content Management Server 2001 site when it is staged with the MSCMS 2001 Site Stager.

destination server

The destination server is the MSCMS 2001 server that is importing the contents of an object package.

E

editor

The main responsibilities of an editor are approving or declining pages based on accuracy and overall suitability, revising them as required, and managing pages in all folders where the editor has editing rights.

F

fixed resource

To prevent content providers from editing or deleting a given resource on any page based on a particular template, the template designer can make it a "fixed" resource. There are two ways to create fixed resources:

1. A template designer can create a link to the resource in the template's HTML.
2. A template designer can place the resource in a placeholder, and clear the placeholder's "Content editing" property.

folder

The folder hierarchy is defined according to how an organization needs to control the authoring and editing process. A folder is a storage space (a container) where authors and editors store

pages. If there are one or more editors configured for a folder, pages in that folder must be approved by one of those editors (or by an administrator) before they can continue through the publishing process. If there is no editor configured for a folder, Content Management Server 2001 auto-approves pages for that folder when authors submit them. Pages for administrators and for editors in folders that they edit, are always auto-approved.

framed site

A Web site that uses HTML frames throughout to display its hierarchies of channels and postings.

frameless site

A Web site that displays its hierarchies of channels and postings without using frames.

G

gallery

A gallery is an MSCMS 2001 storage space (a container) for either templates or resources (such as images). Templates and resources are stored in separate galleries.

Guest User, Content Management Server 2001

If permissions for guest access have been set, Content Management Server 2001 uses this account for anonymous site visitors. Guest users must be granted rights to channels and postings, just like any other MSCMS 2001 subscriber.

I

IIS – Internet Information Server

Microsoft's Web server software that runs on the Windows 2000 platform.

Initial Content Management Server 2001 Administrator

The "Administrator, Content Management Server 2001" created by the DCA when creating or upgrading a database. Initially, this is the only user who can log on to Content Management Server 2001 through the MSCMS 2001 Site Builder.

(See also *Administrator, Content Management Server 2001*.)

internal resource

A resource that exists only within the context of a specific page, as opposed to a resource that is stored in the Resource Gallery and is available to all pages.

K

kill lock

A "lock" prevents more than one user from modifying an object (such as a page or template) at the same time. Note that locks are not used with the MSCMS 2001 Web Author. The system will automatically place a lock on an object while a user modifies it. When a user, other than the one who has the lock, opens a locked item, the item will open in read-only mode. Using the Kill Lock function, an administrator can remove a lock placed on an object by the system.

L

label

A timestamp applied to Content Management Server 2001 objects (templates, pages, and resources) that identifies that object at a specific time in the past (a revision).

M

Microsoft® Content Management Server 2001 system

A Microsoft Content Management Server 2001 (MSCMS 2001) system is the MSCMS 2001 server, all the MSCMS 2001 Site Builders and Web Authors that connect to the server, and the database. Content Management Server 2001 can also include multiple MSCMS 2001 servers doing different functions, such as development, staging, or production.

moderator

Moderators are responsible for postings in all Content Management Server 2001 channels in which they have moderating rights, with the main responsibilities being:

- approving (or declining) postings, based on relevancy and suitability for the channel's subscribers
- revising posting schedules (if required)
- managing all postings in any channel where they have moderating rights.

MSCMS 2001 page profile definition

The MSCMS 2001 page profile definition used in Content Connector specifies the set of properties and attributes in a page profile. The MSCSM 2001 page profile definition is managed in the Profile Designer module in the Business Desk. This definition can be extended to include more properties.

(See also, *page profiles*.)

MSCMS 2001 window

The Content Management Server 2001 window contains the hierarchies of containers (folders, channels, and galleries) where authors, editors, moderators, resource managers, template designers, and administrators do their work. From this window, authors can open the Authoring window, and template designers can open the Template Design window.

(See also *Authoring window, Template Design window.*)

multilingual support (MLS)

Multilingual support provides the ability to display textual data in a language other than the standard Western European languages.

N

native flags

Native flags allow customers to define specific properties of objects (such as postings, channels) within Content Management Server 2001. These flags include the ability to hide pages at run-time, control Web Robot access, hide page content, and display multilingual navigation links.

navigation template

Navigation templates are Active Server Pages (ASP) files created in MSCMS 2001 template galleries and assigned to channels. The template determines the interface displayed when the channels are displayed in a Web browser. Only template designers and administrators can create and edit navigation templates.

(See also *cover page, page template.*)

O

object

Content Management Server 2001 data objects are maintained in the MSCMS 2001 database and can be manipulated by users. An object can be a container such as a folder, channel, gallery, or user role, or an object such as a page, posting, template, or rights group.

object package

A file that contains objects exported from a Content Management Server 2001 system using the Site Deployment Manager. Object packages have the .ROP file extension and .ROP files can be imported to another MSCMS 2001 system.

(See also *Site Deployment Manager.*)

object property

The object property allows users to define customized pieces of data which can be added to Content Management Server 2001 objects. This allows the site generally to be much more extensible to allow greater scripting and searching capabilities.

P

page

Pages in Content Management Server 2001 are created by adding content, such as text or images, to the placeholders on templates. The MSCMS 2001 pages are different from most Web pages in that different views can be generated from a single page and each view of the page can be posted to a different channel. Each view is the HTML page dynamically assembled and delivered when a subscriber selects an MSCMS 2001 posting. Because pages can use several views, each one based on a different template, the content of the views can be customized to suit different audiences.

(See *placeholder, view.*)

page groups

Page groups are used in Content Connector. Personalized content objects use page groups to target specific locations on a site. Page groups are defined in the Business Desk using the Reference Tables module.

(See *page positions, personalized content objects.*)

page positions

Page positions are used in Content Connector. They are identifiers defined in the Business Desk and used to provide additional context filtering when selecting content with personalized content objects.

page profile

Used in Content Connector, a page profile is a collection of properties describing a Content Management Server 2001 page. During personalization, page profiles are scored by content groups in order to determine what set of profiles belong to the group. There is one page profile definition for each site and it is defined in the Commerce Server Business Desk Profile Definition module for that site. Profiles of individual pages can be created and managed from either the MSCMS 2001 Web Author or the Commerce Server Business Desk.

(See also *content expressions, MSCMS 2001 page profile definition.*)

page revision

A page revision or historical revision is a "snapshot" of a Content Management Server 2001 object (page, posting, template, or resource) that can be previewed and reverted to. An individual revision of an object also maintains links to all dependencies that were in effect when the revision was approved.

page status

(See *status*.)

page template

Content Management Server 2001 page templates create a view of a page that includes one or more placeholders in a pre-configured layout. Authoring a page involves adding content (text, images) to the placeholders. When creating a page, an author can add more views to it. This way an author can quickly create different versions of the page to suit different audiences. Only template designers and administrators can create or make changes to templates.

(See also *navigation template, view, placeholder.*)

personalized content object (PCO)

Personalized content objects are used in Content Connector. They deliver collections of page profiles to Content Management Server 2001 pages. When a user requests a page that executes a personalized content object pipeline, information about the current user and the current context within the Web site are sent to the pipeline. Based on your targeting and filtering definitions, personalized content objects currently active in the Web site are evaluated to determine which ones are suitable for the current user in that particular context. The content from the personalized content objects is scored for relevance, blended, and returned to the page.

placeholder

Placeholders are ActiveX® controls or predefined areas of a view where you add content to create a page using Content Management Server 2001. Placeholders may contain default content, put there by template designers to show authors the type of content a placeholder accepts, or to suggest an appropriate use for the placeholder. The page template designer can put restrictions on placeholders for authoring.

(See also *default content, page template, view.*)

post, posting

A page must be posted to a channel in order for it to be published. A posting defines which view of a page is to be published and when to publish it. (Actual publication is subject to the Content Management Server 2001 approval process.)

"Posting" can also refer to the process by which a page is posted.

(See also *approve, view.*)

posting status

(See *status*.)

priority, important

If you create postings or channels that contain timely information you want read as soon as possible, you can assign an Important priority status to these postings or channels. Navigation programmers can create pages that display only Important priority postings, and can sort lists of channels and postings so that important objects are shown at the top of the list.

Production Manager

The Production Manager feature displays a list of all pages, postings, or templates owned by you that are New, Saved, Waiting for Approval, Approved, Declined, or Expired.

profile, staging

(See *staging profile.*)

proto-template

A proto-template is a page template opened in an HTML editor in conjunction with the template Design Palette. When the template is saved in the editor, the Design Palette saves the template to the template gallery the HTML editor was launched from.

R

Report, Dependant

Generating a Dependant Report on an object (such as a resource, page template, or navigation template) displays a list of all objects that are using it.

Report, Resource

Generating a Resource Report on a page or template displays a list of all resources it uses. When you replace a resource, all pages that use that resource will have that resource replaced; therefore it is important to do a Resource Report on items before replacing them so you can determine what the overall effect is of replacing that resource.

resource

A file that is added to a template or a page from somewhere other than a gallery. Once nothing is referencing it, it is deleted by background processing.

Resource Gallery

A container used within Content Management Server 2001 to organize and control access to resources.

Resource Gallery item

Any file that is stored within a Content Management Server 2001 resource gallery. To build a page, a user adds resources to placeholders in a view. Template designers can also create references to resources within a template's HTML code.

resource manager

Resource manager is one of the formal Content Management Server 2001 roles and is someone who maintains and updates resources in the galleries where they have rights. Only the resource managers and administrators have read and write access to the files in a resource gallery and can therefore add, replace, preview, and delete resources.

Resource Report

(See *Report, Resource*.)

rich product page

Used in Content Connector, a rich product page is a Content Management Server 2001 page linked to a specific product in a Microsoft® Commerce Server 2000 catalog. It stores its rich HTML and multimedia content in MSCMS 2001 and displays it on the site along with the product properties from Commerce Server 2000.

(See also *simple product page.*)

rights

A Content Management Server 2001 user's rights are derived from their membership in one or more rights groups and the roles (subscriber, author, editor) they have in the containers (folders, channels, galleries) assigned to those rights groups. All containers you have rights in or rights to have colored icons.

- The rights group(s) you belong to determines which containers you have rights in.
- The role of the rights group(s) you belong to determines the types of tasks (create, modify, approve) you can do with items in a container.

(See also *rights group, role.*)

rights group

A rights group is a group of users assigned to one of the Content Management Server 2001 predefined roles. Members of a rights group hold privileges to containers, based on the role the rights group represents. Authors, for example, can create pages in a folder, while editors can both

create and approve pages in folders. Access to specific containers (folders, channels, galleries) is set for a rights groups by an MSCMS 2001 administrator.

(See also *rights, role*.)

role

In Content Management Server 2001, your role (subscriber, author, editor, moderator, template designer, administrator, resource manager) determines what actions you can carry out in containers. The different roles and the tasks allowed by those roles, are key components of the collaborative content management process.

(See also *rights*, *rights group*.)

S

server, MSCMS 2001

The MSCMS 2001 server is a machine that hosts the Microsoft Content Management Server 2001 software. This server is connected to an MSCMS 2001 database and interacts with MSCMS 2001 Site Builder or Web Author software.

simple product page

Used in Content Connector, a simple product page is a single MSCMS 2001 page for displaying product information for Commerce Server 2000 products. Products without a rich product page automatically use the designated simple product page. You have the option to make simple product pages visible on the live site or visible only in edit mode.

(See also, *rich product page*.)

Site Builder, MSCMS 2001

A client-side, WSIWYG (what-you-see-is-what-you-get) authoring environment that operates on a Windows 32-bit platform. The MSCMS 2001 Site Builder is used to develop pages and navigation templates, develop code with the API, and do site administration tasks. The MSCMS 2001 Web Author should be used for all authoring, editing, and moderating functions.

site programmer (developer), Content Connector

The site programmer is responsible for creating and editing Content Management Server 2001 templates and solution site files for Content Connector sites.

Site Stager, MSCMS 2001

A Content Management Server 2001 application that takes HTML or ASP "snapshots" of part or all of your Web site, thereby allowing you to host this content from another directory or server.

MSCMS 2001 Site Stager allows you to reduce browser traffic on your MSCMS 2001 site, and to host your site on non-Windows 2000 platforms.

Stage As User

The "Stage As User" is a Windows 2000 user account whose container rights determine which channels and postings are staged by the MSCMS 2001 Site Stager. The "Stage As User" isn't a real user. It is a dedicated Windows 2000 user account that an administrator must create for staging to occur. Creating the Stage As User involves creating a Windows 2000 user account, and adding its user name to Content Management Server 2001 subscriber or administrator rights group. When Site Stager stages a site, it stages the channels assigned to members of this rights group.

staged site

The MSCMS 2001 Site Stager allows you to create a static HTML site that can be ported to various platforms. A staged site consists of HTML "snapshots" taken of the channels and postings or views of pages of the Content Management Server 2001 site to be staged. The contents of the staged site can be hosted on a Web server for users who want to browse published content on your site.

staging schedule

The staging schedule, which can be set from the Profile Properties dialog box, determines when a particular staging profile is staged.

staging profile

A staging profile is a collection of all the information the MSCMS 2001 Site Stager needs to stage a site. A profile contains a name (to distinguish it from other profiles), the directory of the site to be staged, the channels to be staged, and a schedule that sets the staging frequency. You can have more than one staged site by creating a different profile for each one.

Site Stager, destination directory

The directory on the machine that receives the HTML "snapshots" of a Content Management Server 2001 site being staged.

Site Stager, Site Map URL

The MSCMS 2001 Site Stager Site Map URL points to a file (sitemap.asp) that generates a description of the Content Management Server 2001 system to be staged.

status

The status or state of a page (in the MSCMS 2001 Web Author) or a page or posting (in the MSCMS 2001 Site Builder) indicates how far along it is in the Content Management Server 2001 publishing process. You can check the status of pages and postings in the status column next to its name in the MSCMS 2001 window's list frame. As well, there is a column titled Backup Exists that further explains status.

Subscriber

Content Management Server 2001 subscribers have access through their Web browser to view MSCMS 2001 pages in the channels to which they're subscribed.

system administrator, Content Connector

The system administrator's job in Content Connector is to install Content Connector, customize and deploy the Method Systems sample site, and to create Content Management Server 2001 user groups, channels, and resource gallery.

T

template

Content Management Server 2001 uses two types of templates, page templates and navigation templates. Page templates create a view of a page that includes one or more placeholders in a pre-configured layout. Authoring a page involves adding content, for example text or images, to the placeholders. When creating a page, an author can add more views to it by applying different templates to it. In this way, an author can quickly create different versions of the page to suit different audiences.

Templates have no life-cycle and can be considered to be in either a checked-in (Saved) or checked-out (Approved) state.

Only template designers and administrators can create or make changes to MSCMS 2001 templates. Most references to templates in Content Management Server 2001 documents are to page templates.

(*See page template. See also navigation template, placeholder, view.*)

Template Design Palette

The template Design Palette opens when an Edit Template HTML or New Template command is selected. The Design Palette manages the joint session between Content Management Server 2001 and the configured HTML editor. The Design Palette provides access to items in the Resource Gallery as well as placeholders that can be dragged and dropped into proto-templates. When the HTML editor saves the template, the Design Palette detects the save action, and imports the new version of the template back into the database.

Template Design window

A template designer opens templates in the Template Design window, where placeholder properties can be configured, default content can be added, and hyperlinks can be inserted.

Template Gallery

A container used to control and manage access to templates.

template status

You can check the status of a template by looking at the status column next to its name in the Content Management Server 2001 window's list frame. The status indicates the availability of the template for use.

(See also *status*.)

V

view

A page in Content Management Server 2001 is a set of one or more views and the content that appears in those views. Content can be shared between all views of a page or any subset of them, or can appear in one only. A view can be considered to be an instance of a page template.

(See also *page, template*.)

W

Waiting

The status of a page after it has been submitted for approval and before it's approved (or declined). The status of a posting after the page it's for is approved and before the posting itself is approved (or declined).

(See also *status*.)

Web Author, MSCMS 2001

The MSCMS 2001 Web Author allows authors, editors, and moderators to create and approve pages from a Web browser. The MSCMS 2001 Web Author is a Web application that uses server-side and client-side scripting to generate an editable version of an existing or new posting.

Index

A

Access folder 7
ActiveXToolbarHooks.inc. 117
adding a description
  for multilingual display 143
  to a template gallery folder 21
administrators 104
Approval process
  editor 130
  moderator 131
Approved 88
  See also published states
approving templates 30, 46, 86
ASP (Active Server Pages)
  in Access folders 7
  navigation templates 5
ASP Server.HTMLEncode method 61
Authenticate.inc. 113
authentication
  for Content Connector sites 202
  process 7, 8
authoring controls 24, 42
authoring process. *See* publishing process, authoring
AuthorModeHooks.inc. 118
authors 104
Autosession object 16, 98, 126

B

broken links. *See* Find feature
browser caching 99
Business Desk 201
business managers 185

C

Cascading Style Sheets (CSS). *See* WBC.css
ChannelOpHooks.inc 116
ChannelPicker directory 177
channels
  about simulated channels 58
  assigning a template 31, 46
  hiding from navigation in Web Author 171
  internationalization (multinational characters) 143
  setting explicit order for children 66
  viewing in a Web browser 33
Classes directory 177
CommonClient.js 122
CommonServer.inc. 122
CommonURL.inc. 121
CommonURLHooks.inc 117, 118
configuring an HTML editor 71
ConsoleUI.inc 115
Content Connector
  directory structure 176
  Framework API 178
  framework directory 177
  overview 175
  processing scripts 201
  product search 197, 201
  sample scripts 195, 196
controlling browser caching 99
controls
  authoring 24
  in templates 24, 42, 183, 185
copying templates 75
cover pages 7
creating
  framed interfaces 13
  frameless interfaces 37
  page template outlines 49
  profiled pages for user targeting 184
  rich product pages 195
  simple product pages 195
custom properties
  an example 77
  assigning to templates 77, 195
  benefits 77
  best practices 78
  for profiled pages 184
  setting 77
  sharing 78

Customizable directory 115, 176, 177
customization
  authoring console 115, 138, 176
  redesigning the interface 137
  storing customized code 144

D

Dates.js. *See* Web Author, JavaScript files
Deleted Items container 89
Deleted. *See* publishing states
deleting templates
  compared with purging 89
  implication for dependant channels 89
descriptions
  adding multinational characters and fonts 143
  adding to a template gallery folder 21
development process
  for personalizing your site content 184–188
  framed tutorial interface 15
  frameless tutorial interface 39
DisplayName property
  default values 128
  enabling multilingual navigation 139
  relationship to hierarchical URLs 141
  support for multinational characters and fonts 143

E

EditOpsHooks.inc. 117
Editor Declined. *See* publishing states
editors
  overview of approval process 130
  tasks defined 104
EditSiteOps.inc 120
errors
  handling API errors 145
  handling JavaScript errors 147
  handling VBScript errors 16, 146
Expired. *See* publishing states
external resources 83

F

Find feature 92
fonts
  enabling styles for the Web Author 144
  with custom properties and multilingual display 143
forms
  masking for Web Author 108
framed interfaces
  creating 13
  overview 5
framed sites
  enabling Web Author 106
frameless interfaces
  creating 37
  overview 38
frameless sites
  enabling Web Author 106
frameset
  creating the sample frameset 17
frameset URL
  setting for a framed template 28, 83
Framework API
  including (instantiating) 179
  list of properties and methods 178–179
  overview 178
framework directory 177

G

guest user
  access 98
  in authentication process 8

H

hidden flag 171
Hooks directory 116
HTML editor
  configuring 71
  selecting 22, 73

I

icons 74
Important.asp. *See* Publishing API
ImportantItems method. *See* Publishing API, Searches
importing templates 82
Include directory 177
indexing for the Web 169
Internals directory
  Link Support 119
  Shared 121
internationalization. *See* DisplayName property
ItemCollection object. *See* Publishing API

J

JavaScript
  files
    CommonClient.js 122
    Dates.js 122
    LinkSupport.js 121
    OptionsClient.js 114
    Windows.js 122
  preventing unauthorized JavaScript 61
  use with switching templates 64

L

LinkSupport.js 121

M

manual login page 8
moderator approval. *See* publishing process
Moderator Declined. *See* publishing states
moderators
  overview of approval process 131
  tasks defined 104
multinational characters. *See* internationalization

N

navigation templates
  about statuses 84
  assigning to a channel 31, 46
  copying 75
  creating for framed interface 19
  creating for frameless interface 40
  deleting compared with purging 89
  editing 75
  importing 82
  location 74
NewChannels (method) 59
NewPosting (method) 59
NRCSCatalog object 193
NRCSCategory object 193
NRCSPosting object 179
NRCSProductContainerRoot object 193
NRCSProfiledPosting object 179
NRCSRichProductPosting object 194
NRCSSession object 178
NRCSSimpleProductPosting object 194

O

objects
  changing read/write properties 148
  commiting 148–149
  rolling back changes 149
optimization for Web Author 123–125, 144
OptionsClient.js. 114
OptionsServer.inc 114

P

page fragment targetting 188
page groups 184
page positions 184
page profiles 184
page template outlines 49

PageProfile directory 177
personalization. *See* personalized content objects
personalized content objects
  analyzing campaigns 201
  displaying results 185
  embedding in a template 185
  pipeline 201
  responsibilities of business manager 185
  responsibilities of site administrators 201
  responsibilities of site programmer 184
PostingCreation.inc 119
PostingCreationHooks.inc 118
PostingOpHooks.inc 116
postings
  and personalized content objects 185
  approved and unapproved versions 134
  hiding from navigation in Web Author 171
  tips for working with in Web Author 129
previewing templates 58
product catalog
  list of framework objects and properties 193–194
  listing product items 196
  performing searches 197
  product navigation 195
product search (Content Connector) 197, 201
Production Manager 86
Published. *See* publishing states
Publishing API
  and the Content Connector Framework API 193
  ASP Server.HTMLEncode method 61
  introduction to 4
  Searches 60
  sorting 65
  supported scripting languages 97
publishing process
  authoring 126
  editing a posting 132
  moderator approval 131
  overview 103
  setting up 105
publishing states
  Approved 133
  Deleted 134
  Editor Declined 133
  Expired 134
  Moderator Declined 133
  New 133
  Published 133
  Saved 133
  Waiting for Editor Approval 133
  Waiting for Moderator Approval 133

R

read/write properties 148
Resolution.inc 57, 113, 123
ResolutionLogin directory 177
ResolveURL 153
ResourceManagement.inc 120
ResourceManagerHooks 118
Revert to Approved command 88
rich product
  pages 195
  postings 185
  template 188
rights 74
roles
  administrators 104
  business manager 185
  resource managers 104
  site manager 104
  site programmer 184
  template designers 104
rolling back changes 149

S

search page
  linking to 172
  sample files 172

Searches object 60
searches. *See* Find feature
server files 7
setting up publishing 105
Shared directory 121
Shared folder 7
simple product pages 195
Site Stager
  .cab files (ActiveX) 154
  .jar files 154
  avoiding use of CreateObject() 155
  IsModeStaging mode check 153
  references to external URLs 154
  resource types 155
  server-side scripts 153
  use of ActiveX 153
  use of Java applets 153
  use of ResolveURL() 153
  use with switching templates 64
SiteEmbeddedWBCLinks directory 114, 115
SiteMode.inc. 119
SiteModeUI.inc. 115
SolutionSite directory 176
sorting
  a channel's children by their properties 66
  a channel's children explicitly 66
  postings returned by the SelectContent method 185
status of templates 85, 88
submitting templates 86
supported scripting languages 97
switching templates
  alternate templates 63
  client-side JavaScript 64
  JavaScript 64
  logic 63
  server-side ASP 64, 65
  use with Site Stager 64

T

TaskAssistantHooks.inc 118
template Design Palette 24, 42
template switching feature. *See* switching templates
TemplateHTMLHooks.inc 116, 144
TemplatePicker directory 177
templates
  alternate 63
  approving 30, 46, 86
  assigning custom properties 77, 195
  creating for product pages 195
  creating profiled pages 184
  editing 88
  embedding a personalized content object 185
  previewing 58
  reverting to approved 88
  status 85
  switching
    client-side JavaScript 64
    server-side ASP 65
  tags for Web Author 105
  user targetiing 188

U

Unpack directory 176
URL Generation Hooks directory 114
URLs
  about 57
  hierarchical 141
  propagating parameters 57
user profiles 202
user targeting 183

W

Waiting for Editor Approval. *See* publishing states
Waiting for Moderator Approval. *See* publishing states
WBC.css. 114
WBCCommonShared.inc. 121
WBCDroplistConsole.inc 115
WBCExtensions directory 177
WBCFlatHTMLConsole.inc 115
WBCSite.inc 121

Web Author
best practices 144
changing objects 148
committing objects 148
Customizable directory
Hooks 114
list of sub-directories 115
modifications for Content Connector 176
SiteEmbeddedWBCLinks files 114
URLGenerationHooks files 117
WBCFuncPagesConfiguration files 118
customizing
about 113
best practices 144
for internationalization 139
storing customized code 144
enabling
for exceptional cases 108
for framed sites 106
for frameless sites 106
errors
handling API errors 145
handling JavaScript errors 147
handling VBScript errors 146
extensions for Content Connector 177
external files 113
forms 108
Internals directory
LinksSupport 119
Shared 121
JavaScript files 122
CommonClient.js 122
Dates.js 122
LinkSupport.js 121
OptionsClient.js 114
Windows.js 122
location of files 105
optimization 123–125
read/write properties 148
required template tags 105
rolling back changes 149
Server Side Includes
ActiveXToolbarHooks.inc 117
AuthCommands.inc 120
Authenticate.inc 113
AuthorModeHooks.inc 118
ChannelOpHooks.inc 116
CommonServer.inc 122
CommonURLHooks.inc 117, 118
ConsoleUI.inc 115
DefaultSiteModeUI.inc 115
EditOpsHooks.inc 117
EditSiteOps.inc 120
OptionsServer.inc 114
PlaceholderHooks.inc 117
PostingCreation.inc 119
PostingCreationHooks.inc 118
PostingOpHooks.inc 116
Resolution.inc 113, 123
ResourceManagement.inc 120
SiteMode.inc 119
TaskAssistantHooks.inc 118
TemplateHTMLHooks.inc 116, 144
WBC.css 114
WBCCommonShared.inc 121
WBCDroplistConsole.inc 115
WBCFlatHTMLConsole.inc 115
WBCSite.inc 121
WorkflowHooks.inc 116
tips for working with postings 129
updating objects 148
Web indexing 169
Windows.js. *See* Web Author, JavaScript files
WorkflowHooks.inc 116