

Oracle® Fusion Middleware
Administrator's Guide for Dynamic Converter
11g Release 1 (11.1.1)
E10634-01

May 2010

Oracle Fusion Middleware Administrator's Guide for Dynamic Converter, 11g Release 1 (11.1.1)

E10634-01

Copyright © 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Mike Manier

Contributor: Ron van de Crommert, Alec Han

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation shall be subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License (December 2007). Oracle USA, Inc., 500 Oracle Parkway, Redwood City, CA 94065.

This software is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications which may create a risk of personal injury. If you use this software in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of this software. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software in dangerous applications.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

This software and documentation may provide access to or information on content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.

Contents

Preface	ix
Audience	ix
Documentation Accessibility	ix
Related Documents	x
Conventions	x
1 Introduction	
1.1 About Dynamic Converter	1-1
1.2 What's New	1-2
1.3 Basic Dynamic Converter Concepts	1-2
1.4 Dynamic Converter Process	1-2
1.5 Upfront Conversions	1-3
1.6 Forced Conversions	1-4
1.7 Fragment-Only Conversions	1-4
1.8 Caching and Querying	1-5
1.8.1 Caching of Timestamps	1-5
1.8.2 Metadata Changes	1-5
1.8.3 Timestamp Checking Frequency	1-6
1.8.4 Cache Interval	1-6
1.8.5 Cache Size	1-6
1.8.6 Cache Expiration Period	1-6
1.9 Special Conversions	1-7
1.9.1 Conversion of HTML Forms to HTML	1-7
1.9.2 Conversion of XML to HTML	1-8
1.9.3 Rendering Paragraphs as Graphics	1-8
1.10 Dynamic Converter Interface in Content Server	1-9
1.10.1 Dynamic Converter Admin Link	1-9
2 Configuring Dynamic Converter	
2.1 Before Using Dynamic Converter	2-1
2.2 Setting the Default Template	2-1
2.3 Setting Up Conversion Formats	2-2
2.3.1 Adding File Formats For Dynamic Conversion	2-2
2.3.2 Removing File Formats From Dynamic Conversion	2-3
2.4 Configuring Slideshow Template Files for PowerPoint Presentations	2-3

2.5	Removing Wireless Templates.....	2-5
-----	----------------------------------	-----

3 Template Rules

3.1	About Template Rules.....	3-1
3.2	Managing Your Template Rules.....	3-1
3.2.1	Adding a Rule.....	3-2
3.2.2	Deleting a Rule.....	3-2
3.2.3	Reordering the Rules.....	3-2
3.3	Assigning Metadata Criteria to a Rule.....	3-3
3.4	Choosing a Template for a Rule.....	3-3

4 Conversion Templates

4.1	About Templates.....	4-1
4.2	Template Types.....	4-1
4.3	Template Strategy.....	4-2
4.4	Checking In a Template.....	4-2

5 HTML Conversion Templates

5.1	About Templates.....	5-1
5.1.1	Creating a New HTML Conversion Template.....	5-2
5.1.2	Editing an Existing HTML Conversion Template.....	5-3
5.2	HTML Conversion Template Editor.....	5-3
5.2.1	Formatting Different File Types.....	5-4
5.2.2	Adding Document Properties.....	5-4
5.2.3	Adding Text Elements.....	5-5
5.2.4	Adding Navigation Elements.....	5-5
5.2.5	Configuring HTML Settings.....	5-5
5.2.6	Adding Output Markup Items.....	5-6
5.2.7	Adding Output Text Formats.....	5-6
5.2.8	Adding Format Mapping Rules.....	5-7
5.2.9	Adding Output Page Layouts.....	5-8
5.2.10	Previewing Your Content.....	5-8
5.2.11	Saving Your Template.....	5-9
5.3	Classic HTML Conversion Template Editor.....	5-9
5.3.1	Template Elements.....	5-10
5.3.2	Sample Classic HTML Conversion Templates.....	5-11
5.3.2.1	Academy.....	5-12
5.3.2.2	Acclaim CSS.....	5-12
5.3.2.3	Account.....	5-13
5.3.2.4	Adagio CSS.....	5-13
5.3.2.5	Administration.....	5-13
5.3.2.6	Analysis.....	5-14
5.3.2.7	Archive CSS.....	5-14
5.3.2.8	Blank.....	5-14
5.3.2.9	Business.....	5-15
5.3.2.10	Ceremonial.....	5-15

5.3.2.11	Courtesy	5-16
5.3.2.12	Executive	5-16
5.3.2.13	Introduction CSS.....	5-17
5.3.2.14	Lotus 1-2-3	5-17
5.3.2.15	Lotus Freelance	5-17
5.3.2.16	MS Excel.....	5-18
5.3.2.17	MS PowerPoint	5-18
5.3.2.18	Purple Frost	5-18
5.3.2.19	Retrofied! CSS	5-19
5.3.3	Migrating From Script Templates to Classic HTML Conversion Templates.....	5-19
5.3.3.1	Updating an Old Template	5-20
5.3.3.2	Sample of Newly Converted Template (From a Pre-6.0 Version).....	5-21

6 Classic HTML Conversion Layout Templates

6.1	About Classic HTML Conversion Layout Templates	6-1
6.2	Layout Template Contents	6-2
6.3	Tokens in Layout Templates	6-3
6.4	Sample Layout Templates	6-3
6.4.1	default_layout.txt.....	6-3
6.4.2	snippet_layout.txt	6-4
6.5	Creating a Layout Template for Your Content Items	6-5
6.6	Associating a Layout Template With a Template Rule.....	6-6
6.7	Specifying a Default Layout Template	6-6
6.8	Including Scripts, Images, and CSS in a Layout Template.....	6-7

7 Script Templates

7.1	About Script Templates.....	7-1
7.2	Elements	7-2
7.2.1	Element Tree	7-2
7.2.2	Leaf Elements	7-4
7.2.3	Repeatable Elements	7-5
7.2.4	Element Definitions	7-5
7.3	Indexes.....	7-9
7.3.1	Index Variable Keywords.....	7-9
7.3.1.1	Whole Number	7-10
7.3.1.2	Current, Next, Previous, First, and Last.....	7-10
7.3.1.3	Up, Down, Left, and Right	7-11
7.3.2	Creating a Set of HTML Files for Each Slide in a Presentation.....	7-12
7.4	Macros	7-12
7.4.1	About Macros	7-12
7.4.2	Units: {## UNIT}, {## HEADER}, and {## FOOTER}	7-13
7.4.3	Insert Element: {## INSERT}	7-14
7.4.4	Conditional: {## IF...}, {## ELSEIF...}, and {## ELSE}	7-18
7.4.5	Loop: {## REPEAT}.....	7-19
7.4.6	Linking With Structured Breaking: {## LINK}	7-20
7.4.6.1	{## LINK} Usage Scenarios.....	7-21

7.4.6.2	{## LINK} Archive File Example	7-22
7.4.6.3	{## LINK} Presentation File Example	7-22
7.4.7	Linking With Content Size Breaking: {## ANCHOR}	7-23
7.4.8	Comment Put in the Output File: {## IGNORE}	7-24
7.4.9	Comment Not Put in the Output File: {## COMMENT}.....	7-24
7.4.10	Including Other Templates: {## INCLUDE}	7-25
7.4.11	Setting Options Within the Template: {## OPTION}.....	7-25
7.4.12	Copying Files: {## COPY}	7-28
7.4.13	Deprecated Template Macros	7-29
7.5	Pragmas	7-30
7.5.1	Pragma.Charset	7-30
7.5.2	Pragma.CSSFile	7-30
7.5.3	Pragma.EmbeddedCSS	7-31
7.5.4	Pragma.JsFile	7-31
7.5.5	Pragma.SourceFileName	7-31
7.6	Sample Script Templates.....	7-32
7.6.1	Basic	7-32
7.6.2	Elements	7-33
7.6.3	Plain	7-34
7.6.4	SimpleToc.....	7-35
7.6.5	Slideshow, Slideshowb, and Slideshowc.....	7-37
7.6.6	Textout.....	7-40
7.7	Setting Script Template Formatting Options	7-42
7.7.1	Changing the Format Used for Converted Graphics	7-43
7.7.2	Generating Bullets and Numbers for Lists.....	7-43
7.8	Breaking Documents by Structure.....	7-43
7.9	Breaking Documents by Content Size.....	7-46
7.9.1	A Sample Size Breaking Template	7-47
7.9.2	Templates Without {## UNIT} Macros	7-48
7.9.3	Indexes and Size-Based Breaking	7-48
7.10	Using Grids to Navigate Spreadsheet and Database Files.....	7-48

8 HTML Snippets

8.1	About HTML Snippets	8-1
8.2	Portal-Style Website Sample	8-1
8.3	Combining HTML Snippets Into a Web Page	8-2
8.3.1	Generating a Snippet of HTML	8-2
8.3.2	Include HTML Snippet Using Idoc Script Function	8-3
8.4	Inline Dynamic Conversion.....	8-4
8.5	Displaying Content Server Metadata on a Web Page.....	8-4

9 Working With Converted Content

9.1	Viewing Content Information	9-1
9.2	Viewing a Converted File	9-3
9.2.1	Search Results Page	9-4
9.2.2	Content Information Page	9-4
9.3	Previewing a Document Before Check-In	9-5

10 Implementation Considerations

10.1	Metadata Fields With Multi-Byte Characters	10-1
10.2	Conversion of PDF Files in UNIX.....	10-2
10.3	Embedded Graphics on UNIX	10-2
10.4	Use of Vector Versus Raster Graphics Formats.....	10-2
10.5	Converting Vector Graphics and Spreadsheet Text in UNIX.....	10-3
10.6	URL Rewriting.....	10-3
10.7	Relative URLs in Templates and Layout Files.....	10-4
10.8	Browser Caching	10-4
10.9	Image Sizing Rules.....	10-5
10.10	CSS Considerations.....	10-5
10.11	Style Names Used by Dynamic Converter.....	10-5
10.12	Overriding Dynamic Converter Styles	10-6
10.13	Pragma.CSSFile and {## LINK}.....	10-6
10.14	Well-Formed HTML	10-7
10.15	Positional Frames Support.....	10-7
10.16	Template Writing Tips	10-7

A User Interface

A.1	Dynamic Converter Admin Page	A-1
A.2	Dynamic Converter Configuration Page.....	A-2
A.2.1	General Conversion Settings.....	A-3
A.2.2	UNIX Configuration Settings.....	A-5
A.2.3	Classic HTML Template Conversion Configuration Settings.....	A-6
A.2.4	Script Template Conversion Configuration Settings.....	A-7
A.2.5	Conversion and Caching Optimizations	A-7
A.3	Template Selection Rules Page.....	A-9
A.4	Template Check-In Form	A-12
A.5	Edit Templates Page	A-14

B Conversion Filters

B.1	Application Filters	B-1
B.2	Graphics Filters	B-5

C Input File Formats

C.1	Word Processing Formats.....	C-1
C.2	Desktop Publishing Formats	C-3
C.3	Database Formats.....	C-3
C.4	Spreadsheet Formats	C-4
C.5	Presentation Formats.....	C-4
C.6	Graphic Formats.....	C-5
C.7	Compressed Formats.....	C-6
C.8	E-Mail Formats	C-7
C.9	Other Formats.....	C-8

D Office 2007 Considerations

D.1	All Office Applications.....	D-1
D.2	Word 2007	D-1
D.3	Excel 2007	D-2
D.4	PowerPoint 2007.....	D-2
D.5	Examples of Unsupported Objects	D-3

E Elements Script Template

E.1	About the Elements Script Template	E-1
E.2	Elements Script Template Code.....	E-1

Index

Preface

Dynamic Converter is a Content Server component that performs on-demand document conversion using customizable templates.

Audience

This document is intended for experienced Content Server users.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at <http://www.oracle.com/accessibility/>.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit <http://www.oracle.com/support/contact.html> or visit <http://www.oracle.com/accessibility/support.html> if you are hearing impaired.

Related Documents

For more information, see the *Oracle® Fusion Middleware Dynamic Converter Template Editor Guide*.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

This section covers the following topics:

- ["About Dynamic Converter"](#) on page 1-1
- ["What's New"](#) on page 1-2
- ["Basic Dynamic Converter Concepts"](#) on page 1-2
- ["Dynamic Converter Process"](#) on page 1-2
- ["Upfront Conversions"](#) on page 1-3
- ["Forced Conversions"](#) on page 1-4
- ["Fragment-Only Conversions"](#) on page 1-4
- ["Caching and Querying"](#) on page 1-5
- ["Special Conversions"](#) on page 1-7
- ["Dynamic Converter Interface in Content Server"](#) on page 1-9

1.1 About Dynamic Converter

Dynamic Converter provides an industry-proven transformation technology and on-demand publishing solution for critical business documents. With Dynamic Converter, you can easily convert any business document into a web page for everyone to see without use of the application used to create that document. The benefits are immediate; information can be exchanged freely without the bottleneck of proprietary applications.

When a web browser first requests a document, a set of rules are applied to determine how that document should appear as a web page. These rules are defined in a template, a core component of Dynamic Converter.

Dynamic Converter offers a number of important benefits to users:

- Business documents can be easily viewed in a web browser.
- Native applications (such as Adobe Acrobat, Microsoft Word, etc.) are not required.
- Multiple renditions of a document are available for different web browsers.
- Templates are interchangeable with Content Publisher.
- Numerous business document types, including legacy formats, are supported.

The HTML renditions of source documents in the Content Server are made available to users via an HTML link on the search results page and the content information page in the Content Server.

1.2 What's New

Dynamic Converter 11gR1 provides the following new and enhanced features (compared to version 10gR3):

- **Compatibility with Content Server 11gR1:** This version of Dynamic Converter is designed to work seamlessly with Content Server 11gR1.
- **New template editor option:** Dynamic Converter now offers the HTML Conversion Editor to use for customizing your templates. The former GUI Template Editor is now named the Classic HTML Conversion Editor. Both editors have extensive Help systems that can be launched from within the interface.

1.3 Basic Dynamic Converter Concepts

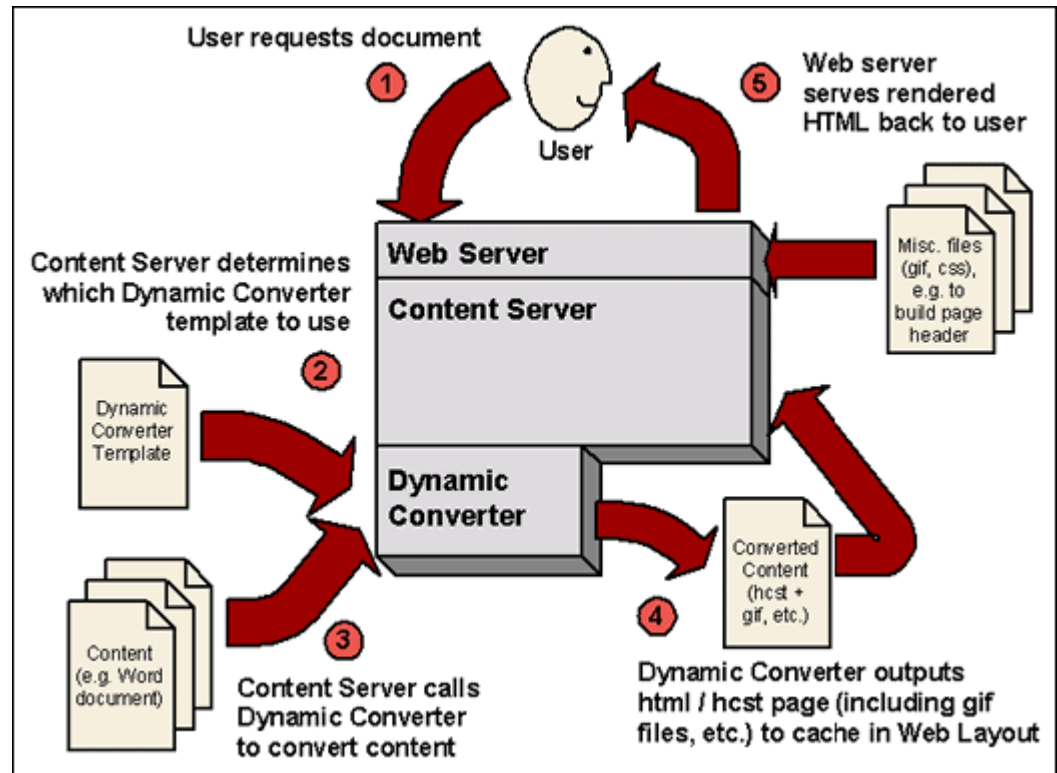
The following concepts are important in the context of Dynamic Converter:

- **Developer:** The individual who integrates Dynamic Converter into another technology or application.
- **Source file:** The document, spreadsheet, presentation or other information that the developer wishes to convert to a web page (also referred to as source document and content item).
- **Output file:** The file being created from the source file (also referred to as the web-viewable format).
- **Output files:** The complete set of files that together make up the rendered output (web page) from a source file.
- **Template:** A template tells the conversion engine how to convert the source document into the output document.
- **Template rules:** Documents matching certain criteria are converted using the specified template, layout, and options.

1.4 Dynamic Converter Process

[Figure 1-1](#) shows the basic Dynamic Converter process.

Figure 1-1 Basic Dynamic Converter Process



The process consists of five steps:

1. A user requests a content item through a web browser.
2. The web server passes this request to Dynamic Converter, which determines the template to be used for the HTML conversion (based on metadata matching criteria).
3. Dynamic Converter converts the native file (for example, a Word document or Excel spreadsheet).
4. The conversion produces one or more HTML pages with supporting files (GIF, JPEG, and so on), which Dynamic Converter outputs to a special caching area in Content Server's web-viewable file repository ("Web Layout").
5. The web server retrieves any additional files (for example, CSS files or images used for the page header and footer), and serves these, together with all files produced by Dynamic Converter, to the user.

Note: Dynamic Converter uses caching to reduce the load on the server and ensure that documents are not unnecessarily re-translated.

1.5 Upfront Conversions

In earlier versions of Dynamic Converter, a content item was converted to a web-viewable format (HTML, XML, etc.) when the content item was first requested by the user; more specifically, when the user clicked the (HTML) link beside the content item on the search results or content information page. Once the content item was converted, it was cached in the Content Server so that each subsequent request for the converted file would be immediate.

Since version 6.0 (circa 2004), Dynamic Converter converts content items that match a conversion rule when the content item is checked in, rather than when the user requests it. As a result, users will be able to immediately view the dynamically converted rendition of the content item.

This upfront conversion applies only to content items that match a conversion rule in Dynamic Converter. Rules are specified on the Template Selection Rules page (see "[Template Selection Rules Page](#)" on page A-9).

If no rule exists for the content item, then an upfront conversion will not take place, even if a default template and layout file are available for the content item. The default templates and layout files are specified on the Dynamic Converter Configuration page (see "[Dynamic Converter Configuration Page](#)" on page A-2).

Please note that upfront conversions must be enabled in the Conversion and Caching Optimizations section of the Dynamic Converter Configuration page (see "[Conversion and Caching Optimizations](#)" on page A-7).

1.6 Forced Conversions

You can designate multiple conversions of the same content item so that it can be used for different purposes on your web site. You might, for example, include it as a snippet of HTML code in one location and as a complete article in another location. This is done using a forced conversion in Dynamic Converter.

Forced conversions allow you to specify a list of rules where every rule is evaluated. If the first rule matches, it will be applied. If the next rule matches, it will also be applied, and so on. In this way, Dynamic Converter may create multiple renditions of the same content, if necessary. As a result, content can be converted multiple times using different templates and layout files.

You can enable forced conversion for a template rule on the Template Selection Rules page (see "[Template Selection Rules Page](#)" on page A-9).

A forced conversion takes place at the same time as an upfront conversion; that is, when the content item is checked into the Content Server. The end users will not be able to tell the difference between an upfront conversion and a forced conversion. Regardless of the method, the goal is the same: to have a content item converted and stored in cache by the time the user clicks the "(HTML)" link.

Please note that forced conversions must be enabled in the Conversion and Caching Optimizations section of the Dynamic Converter Configuration page, along with upfront conversions (see "[Conversion and Caching Optimizations](#)" on page A-7).

1.7 Fragment-Only Conversions

One type of forced conversion (see "[Forced Conversions](#)" on page 1-4) is the fragment-only conversion. A fragment is a piece of content that will be included in another content item. Individual fragments can then be combined to form a content-rich web page. A fragment generally contains no `<html>` or `<body>` tags, so that it can be easily included in another web page. The fragment is not intended to be viewed by itself and as such should not be displayed to users who click the "(HTML)" dynamic conversion link. Rules designed for fragments should be excluded from Dynamic Converter's rule evaluation during a user request.

When forced conversions are selected, you can enable fragment-only conversion for a template rule on the Template Selection Rules page (see "[Template Selection Rules Page](#)" on page A-9).

Like other forced conversions, fragment-only conversions take place upfront, when the content item is checked into the Content Server.

1.8 Caching and Querying

Dynamic Converter includes a conversion and caching strategy that significantly improves the overall performance of your intranet or external web site. This Element allows Content Server to serve up dynamically created web pages much more quickly than was possible in earlier versions.

While the conversion and caching enhancements are built into the application, there are several configuration options that you can set in order to fine-tune Dynamic Converter:

- [Caching of Timestamps](#)
- [Metadata Changes](#)
- [Timestamp Checking Frequency](#)
- [Cache Interval](#)
- [Cache Size](#)
- [Cache Expiration Period](#)

All these configuration options can be set in the Conversion and Caching Optimizations section of the Dynamic Converter Configuration page (see "[Conversion and Caching Optimizations](#)" on page A-7).

1.8.1 Caching of Timestamps

Every time a user clicks the "(HTML)" dynamic conversion link on the search results page or content information page, three files are queried in the Content Server database: the source document, the conversion template, and the layout file (if applicable). The database queries confirm that the dynamically converted file is the most recent, but these queries are done even when an up-to-date conversion is available.

Dynamic Converter version 6.2 and higher use a new method of verifying the revision of content items and conversion templates without querying the database each time. Instead, the time stamps of the converted content items are stored in the server's memory-based cache. Future conversion requests can then compare these cached time stamps with the time stamps of the content item to be converted without querying the database. When combined with the upfront conversion Element (see "[Upfront Conversions](#)" on page 1-3), Dynamic Converter becomes much more efficient in its revision and conversion queries. Using time stamps, the caching and querying mechanism detects the new revisions of content items in the Content Server, because with each new revision a new file is created with a new time stamp.

1.8.2 Metadata Changes

If you or your users make metadata-only changes to a content item, neither a new file nor a new time stamp is created, and the changes will go undetected. To address this problem, you must make sure that all metadata changes are identified by Dynamic Converter. You can do this by enabling the "Reconvert when metadata is updated" option on the Dynamic Converter Configuration page (see "[Conversion and Caching Optimizations](#)" on page A-7). This option forces the Content Server to update the time stamp of the source content items after a metadata update. With this option enabled, the time stamps of all web-viewable formats are updated to reflect the metadata

change that occurred for the corresponding source content item. The updated time stamp, as a result, will be recognized by Dynamic Converter, and the content item, with metadata updates, will be reconverted.

Database Method of Checking

You can choose to use the database method of checking whether the content item's metadata has been updated. You set this option on the Dynamic Converter Configuration page (see "[Conversion and Caching Optimizations](#)" on page A-7). With this configuration option enabled, content item updates continue to signal timestamp changes in the converted files, but the new caching and querying method is not used to determine if the content items are up to date. Instead, the Content Server database is queried for this information. You might use this method, for example, if you are experiencing problems with the optimized query Element or you are troubleshooting a related issue.

1.8.3 Timestamp Checking Frequency

By default, Dynamic Converter checks the time stamp of the converted content items every 1,500 milliseconds, or 1.5 seconds. You can increase or decrease this value if you would like to balance the number of queries performed with the number of visitors to your site. You can change the timestamp checking frequency on the Dynamic Converter Configuration page (see "[Conversion and Caching Optimizations](#)" on page A-7).

If you increase this setting to, say, one minute (60,000 milliseconds) and a new content item is checked into the Content Server, then the new version will not be available to users until one minute has passed.

1.8.4 Cache Interval

The cache interval is the frequency with which the conversion cache is evaluated and cached items may be considered for deletion, depending on how long they have been in the cache and their conversion status. You can set the cache interval (in days) on the Dynamic Converter Configuration page (see "[Conversion and Caching Optimizations](#)" on page A-7). The default is seven days (i.e., once every week).

1.8.5 Cache Size

Dynamically converted files are kept in a cache to avoid unnecessary re-conversion. You can set the maximum cache size on the Dynamic Converter Configuration page (see "[Conversion and Caching Optimizations](#)" on page A-7). The default is 10,000 MB (about 10 GB). If the cache exceeds this maximum size, then during the next clean-up cycle (which, by default, is seven days) the cached items that have not been accessed for the longest period of time are deleted first. (The list for deleting is sorted by the "last accessed" date in ascending order.) If the cache size limit is not exceeded, then the cached items are examined for potential deletion in the same order, but items that are forced conversions of existing documents are not deleted.

1.8.6 Cache Expiration Period

Dynamic Converter keeps converted content items in the Web Layout conversion cache to prevent items from being reconverted unnecessarily. You can control the number of days that must pass before converted items in the cache may be considered for deletion. By default, cache clean-up is evaluated every seven days. Date expiration only applies to cached items for documents that are no longer present and to cached

items that were not generated by forced conversion (see "[Forced Conversions](#)" on page 1-4). The default cache expiration period is seven days.

The cache expiration setting works in conjunction with the cache interval (see "[Cache Interval](#)" on page 1-6), which controls the frequency with which the cache is evaluated. For example, if the cache interval is set to 14 days and the cache expiration period is set to 8 days, then the cache will be evaluated every 14 days and all cached items older than 8 days will be deleted (unless they were the result of forced conversion).

1.9 Special Conversions

Dynamic Converter supports the following special conversions:

- [Conversion of HTML Forms to HTML](#)
- [Conversion of XML to HTML](#)
- [Rendering Paragraphs as Graphics](#)

1.9.1 Conversion of HTML Forms to HTML

Dynamic Converter supports the conversion of HTML forms into HTML. This allows information supplied through HTML forms to be presented in flexible ways.

For example, the HTML form used to enter data might look something like this:

Figure 1–2 Data Entry Form

The screenshot shows a web form with the following elements:

- ID:** A text input field containing the text "Test ID".
- Color:** A dropdown menu with "Red" selected.
- Shape:** A dropdown menu with "Square" selected.
- Description:** A large text area containing the text "Description".
- Buttons:** Two buttons labeled "Submit" and "Reset" are positioned at the bottom right of the form.

This HTML form, together with the values entered, is automatically checked into the Content Server as an HCSF file when it is submitted by clicking the **Submit** button. If a user then wants to view the form data, a template could be used to present the data from the HTML form.

Figure 1–3 Form Data in Table

Field	Value
ID	Test ID
Color	Red
Shape	Square
Description	Description

Note: Both the HTML form and HTML template shown above are included as samples.

1.9.2 Conversion of XML to HTML

Dynamic Converter supports the conversion of XML to HTML by means of an XSL file. The XSL file (with the extension .xsl) is a template that defines how XML files are presented as HTML in a web browser.

In order for Dynamic Converter to properly identify and convert XML files, you must:

- Check the XSL file into the Content Server.
- Configure Dynamic Converter to recognize XML files. See "[Adding File Formats For Dynamic Conversion](#)" on page 2-2 for an explanation of how to add a file format for dynamic conversion. (In this case, you would add "application/xml" in the Formats text box.)
- Create a Dynamic Converter rule that matches the XML files you want to convert and specify the XSL file as the conversion template for that rule. See [Chapter 3, "Template Rules"](#) for more information.

Note: A sample XML file and XSL file are included in the directory */ucm/Distribution/DynamicConverterSamples*.

To use the XML-to-HTML Element, the following line must be in the *intradoc.cfg* file (which is located in the *DomainHome/ucm/cs/bin/* subdirectory of the Content Server installation directory):

- **For Microsoft Windows:**

```
CLASSPATH=$COMPUTEDCLASSPATH; [CS-Dir]/shared/classes/xalan.jar; [CS-Dir]/shared/classes/xerces.jar
```

- **For UNIX:**

```
CLASSPATH=$COMPUTEDCLASSPATH: [CS-Dir]/shared/classes/xalan.jar: [CS-Dir]/shared/classes/xerces.jar
```

(where *[CS-Dir]* is the installation directory of your Content Server instance).

Note: The Classpath for xalan.jar and xerces.jar on Windows and UNIX is set during the installation of Content Server. You should check the *intradoc.cfg* file to verify this.

1.9.3 Rendering Paragraphs as Graphics

Dynamic Converter lets you render paragraphs as graphics. You can use this feature to add custom and protected fonts to documents without allowing public access to the fonts.

This setting is in the Classic HTML Conversion Template Editor (select **Formatting** and then **Paragraph**). If you are running Dynamic Converter on Windows, your selection of the font to be rendered is the same font that is used in conversion.

If Dynamic Converter is installed on a UNIX platform, the conversion process draws from a different group of fonts. In that event, the font selected in the Template Editor must also be available on the UNIX system. Both fonts must have exactly the same name for the rendering to take effect. The `GD_Font_Path` variable must point to a font directory, and that directory must contain at least one TrueType font with the .tff file extension. If these requirements are not fulfilled, rendering paragraphs as graphics will fail.

When rendering paragraphs as graphics, Dynamic Converter does not support embedded graphics. Any images in the paragraph will be replaced by the string []. Templates should avoid using rendering paragraphs as graphics in sections that contain graphics.

1.10 Dynamic Converter Interface in Content Server

This section covers the changes to the Content Server interface after the Dynamic Converter software is installed:

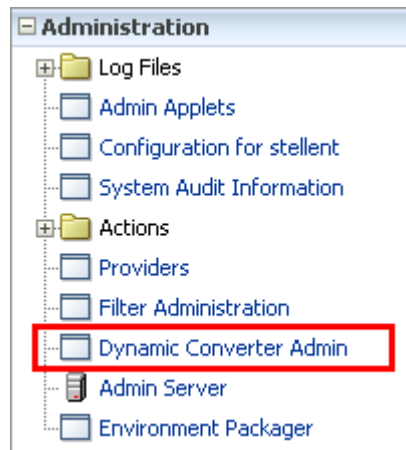
- ["Dynamic Converter Admin Link"](#) on page 1-9
- ["Dynamic Converter Admin Page"](#) on page A-1

See the *Dynamic Converter Installation Guide* for more information on installing the Dynamic Converter software.

1.10.1 Dynamic Converter Admin Link

If Dynamic Converter was added to Content Server successfully, the Administration page and menu includes a link called **Dynamic Converter Admin**.

Figure 1–4 *Dynamic Converter Admin Link in Administration Tray*



If the Dynamic Converter Admin links are missing, the Dynamic Converter component was not correctly installed or enabled. For details on how to install the Dynamic Converter component, refer to the *Dynamic Converter Installation Guide* in the Dynamic Converter distribution package (in the documentation directory).

Configuring Dynamic Converter

This section covers the following topics:

- ["Before Using Dynamic Converter"](#) on page 2-1
- ["Setting the Default Template"](#) on page 2-1
- ["Setting Up Conversion Formats"](#) on page 2-2
- ["Configuring Slideshow Template Files for PowerPoint Presentations"](#) on page 2-3
- ["Removing Wireless Templates"](#) on page 2-5

2.1 Before Using Dynamic Converter

Before you begin using Dynamic Converter to design templates for your content items, the following must be in place:

- Content items checked into the Content Server. See the *Content Server User Guide* for information on how to check in content.
- Dynamic Converter templates checked into the Content Server with the correct template type. See [Chapter 4, "Conversion Templates"](#) for more information.
- Template selection rules added, with associated metadata and templates. See [Chapter 3, "Template Rules"](#) for more information.
- Conversion formats and other pertinent information specified on the Dynamic Converter Configuration page. See ["Dynamic Converter Configuration Page"](#) on page A-2 for more information.

2.2 Setting the Default Template

A default template is applied to content items that do not match your defined template criteria. To change the default template associated with your content items, complete the following steps:

1. Open the Dynamic Converter Admin page (see ["Dynamic Converter Admin Page"](#) on page A-1).
2. Click **Configuration Settings**.

The Dynamic Converter Configuration Page is displayed (see ["Dynamic Converter Admin Page"](#) on page A-1).

3. In the **Template** text box, under the Default Template heading, enter the content ID for a template. You can select the type of template from the **Template Types**

menu and then choose your desired template from the **Available Templates** menu.

4. In the **Layout** text box, under the Default Layout heading, enter the content ID for a layout template. You can also choose your desired layout template from the **Available Layouts** menu. (Layouts only apply for the Classic HTML Conversion templates.)
5. Click **Update** at the bottom of the Dynamic Converter Configuration page to enable your default templates.

In earlier versions of Dynamic Converter, the following error message appeared when a content item did not match any template criteria and a default template (such as "plain.hcst") was not specified:

```
Content Server Request Failed. Could not convert the content to html. The default conversion template has not been set.
```

This is no longer the case. A blank template is automatically assigned to content items that do not match any of your template criteria. You can override this template with your own default template (by following the above steps).

2.3 Setting Up Conversion Formats

The file format (MS Word, RTF, plain text, etc.) of your content items must be included in the conversion formats list in order for Dynamic Converter to recognize and convert the content item. Only file formats included in this list will have an **(HTML)** link beside them on the search results page, content information page, and so on.

This section covers the following topics:

- ["Adding File Formats For Dynamic Conversion"](#) on page 2-2
- ["Removing File Formats From Dynamic Conversion"](#) on page 2-3

2.3.1 Adding File Formats For Dynamic Conversion

You can add one or more file formats on the Dynamic Converter Configuration page at any time. To add a new file format, complete the following steps:

1. Open the Dynamic Converter Admin page (see ["Dynamic Converter Admin Page"](#) on page A-1).
2. Click **Configuration Settings**.

The Dynamic Converter Configuration Page is displayed (see ["Dynamic Converter Configuration Page"](#) on page A-2).

3. Under the Conversion Formats heading, in the formats text box, type the file format that you would like converted into a web page (or select it from the menu to the right). Formats in the text box must be separated by a comma and a space, for example:

```
application/msword, application/vnd.ms-excel
```

File formats must follow the naming convention in Content Server's Configuration Manager. For example, Microsoft Word documents are entered as *application/doc* or *application/msword*.

Note that with Office 2007 files (e.g., docx, xlsx, and pptx), you must enable the following formats on the DC Configuration page to see the **(HTML)** link:

```

application/vnd.openxmlformats-officedocument.presentationml.presentation
application/vnd.openxmlformats-officedocument.presentationml.slide
application/vnd.openxmlformats-officedocument.presentationml.slideshow
application/vnd.openxmlformats-officedocument.presentationml.template
application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
application/vnd.openxmlformats-officedocument.spreadsheetml.template
application/vnd.openxmlformats-officedocument.wordprocessingml.document
application/vnd.openxmlformats-officedocument.wordprocessingml.template

```

For more information on file format naming conventions, see the Content Server administration documentation.

4. Click **Update** to add your file formats to Dynamic Converter.

2.3.2 Removing File Formats From Dynamic Conversion

You can remove file formats on the Dynamic Converter Configuration page at any time. To remove a file format, complete the following steps (you cannot undo this operation):

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Configuration Settings**.

The Dynamic Converter Configuration Page is displayed (see "[Dynamic Converter Configuration Page](#)" on page A-2).

3. Under the Conversion Formats heading, in the formats text box, select the file format that you would like to remove.
4. Press the Delete key on your keyboard to remove the file format from the text box. Be sure that you only remove the format that you wish to eliminate, and not all the formats listed in the box.
5. Click **Update** to remove the file format from Dynamic Converter.

Note: If you accidentally delete the wrong file format, add it again, and then click **Update**.

2.4 Configuring Slideshow Template Files for PowerPoint Presentations

If you intend to use Dynamic Converter templates to convert PowerPoint presentations, it is recommended that you use an HTML Conversion template. You will be able to customize your template by navigating to Document Formatting > Presentations Tab in the template editor.

If you use a Classic HTML Conversion template, you should use the "slideshow" hcst files provided. You will need to check in all three of the slideshow files (*slideshow.hcst*, *slideshowb.hcst*, and *slideshowc.hcst*). A sample PowerPoint file (*dc_powerpoint.ppt*) is also included in the */ucm/Distribution/DynamicConverterSamples* directory.

If you have Dynamic Converter configured in such a way that it automatically assigns content IDs upon file check-in, you need to edit each slideshow template file to reflect this. Each file must then be checked in again before you can begin using the templates.

To configure the slideshow template files for conversion of PowerPoint presentations, complete the following steps:

1. Check all three slideshow files (*slideshow.hcst*, *slideshowb.hcst*, and *slideshowc.hcst*) into the Content Server. Make sure that you check them in as script templates. See ["Checking In a Template"](#) on page 4-2 for more information.

To ensure the files are checked into the correct Web Layout directory, make sure that you use the same content type, security group, and account (if applicable) for all three files.

If the content IDs are generated automatically, locate and note the automatically generated content ID of each slideshow file.

If the content IDs are not generated automatically, it is recommended that you use something like "DC-Slideshow," "DC-SlideshowB," and "DC-SlideshowC."

2. Access the slideshow files.
3. Open each of the slideshow hcst files in a text editor, such as WordPad or vi, and then search for and replace the following slideshow references with the appropriate content IDs:

Important: Be sure to save your changes before closing each file.

- **slideshow.hcst:**

Search for: "slideshowbtemplate"

Replace with: the content ID of the checked-in *slideshowb.hcst* template. For example, 1002 or DC-SlideshowB.

- **slideshowb.hcst:**

Search for: "slideshowctemplate"

Replace with: the content ID of the checked-in *slideshowc.hcst* template. For example, 1003 or DC-SlideshowC.

- **slideshowc.hcst:**

Search for: "slideshowbtemplate"

Replace with: the content ID of the checked-in *slideshowb.hcst* template. For example, 1002 or DC-SlideshowB.

Make absolutely sure that you retain the file extension, so something like:

```
{## link element=sections.current.bodyorimage  
template=slideshowbtemplate.hcst}.
```

If you do not, the application may throw an exception during the HTML conversion.

On UNIX systems, the content ID is case-sensitive, so "dc-slideshow" is not the same as "DC-Slideshow" or "DC-SLIDESHOW."

4. Search for the slideshow files in the Content Server and click the **Info** link on the search results page.
The content information page is displayed.
5. Click **Check out**.
6. Click **Check In** on the check-out confirmation page.

7. Browse to the modified slideshow files in the `/ucm/Distribution/DynamicConverterSamples` directory and click **Check In** on the content check-in form.
8. Repeat steps 4 to 7 for the each of the slideshow files.

You can now set up the conversion format for PowerPoint presentations (see "[Setting Up Conversion Formats](#)" on page 2-2) and assign the checked-in templates to a template rule (see [Chapter 3, "Template Rules"](#)).

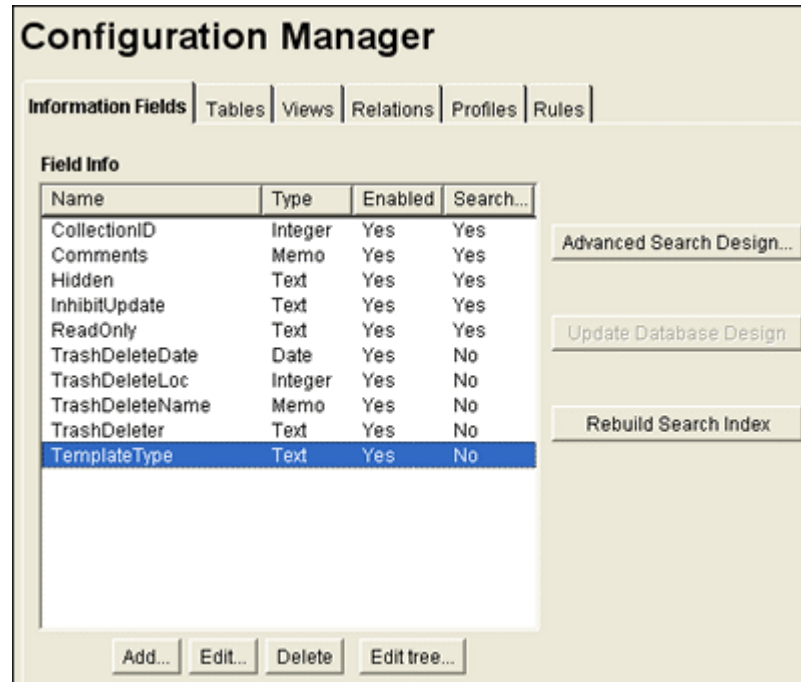
2.5 Removing Wireless Templates

Dynamic Converter 11gR1 does not support the wireless template type, but still provides wireless support based on the Classic HTML Conversion templates. An existing Content Server with an earlier version of Dynamic Converter may still have the wireless template type, but attempting to use it will cause failure.

To remove the wireless template from the list of available templates, complete these steps:

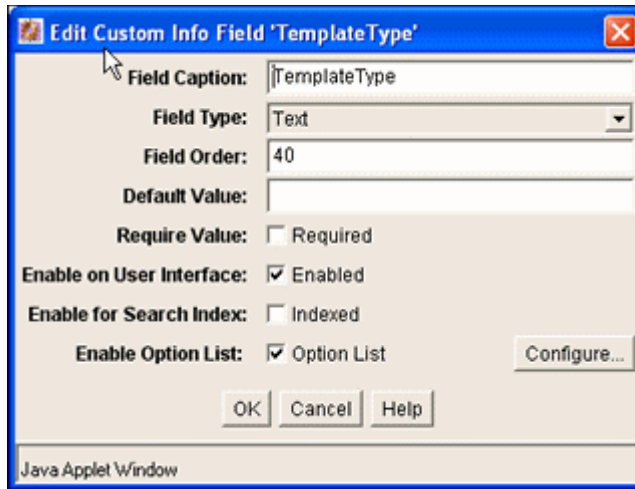
1. Open a new browser window and log into Content Server as a system administrator (with the "sysmanager" role).
2. Open the **Admin Applets** page.
3. Click **Configuration Manager** under the Administration Applets section.
The Configuration Manager window is displayed.

Figure 2–1 Configuration Manager Window



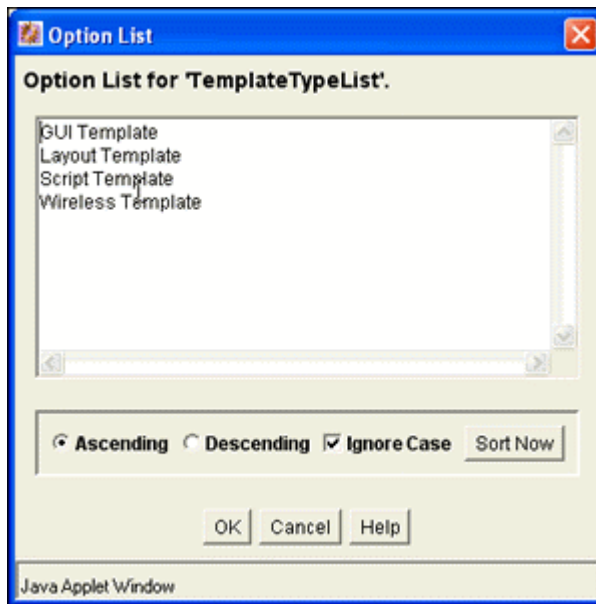
4. On the Information Fields tab, choose the `TemplateType` row and click **Edit**.
The editing dialog for the `TemplateType` field is displayed.

Figure 2–2 Edit TemplateType Dialog



5. Click **Configure**.
The Configure Option List dialog is shown.
6. Click **Edit**.
The Option List dialog is displayed.

Figure 2–3 Option List for TemplateType



7. Highlight **Wireless Template** and press the Delete key to remove the Wireless Template as an option.
8. Click **OK** three times to return to the Configuration Manager window.
9. Choose **Options** from the menu bar.
10. Choose **Publish Schema**. This propagates your changes to the Content Server.
11. Exit the Configuration Manager application and wait a few minutes. Then revisit the template check-in form.

Template Rules

This section covers the following topics:

- ["About Template Rules"](#) on page 3-1
- ["Managing Your Template Rules"](#) on page 3-1
- ["Assigning Metadata Criteria to a Rule"](#) on page 3-3
- ["Choosing a Template for a Rule"](#) on page 3-3

3.1 About Template Rules

A rule is a set of instructions that drive the conversion process in Dynamic Converter. These instructions identify source documents in the Content Server and then determine whether or not these documents should be converted based on their metadata (content ID, type, author, etc.) and file type. The rule then requests that the document be converted using the template associated with the rule (for more on templates, see [Chapter 4, "Conversion Templates"](#)). You can have more than one rule in Dynamic Converter. If this is the case, the first rule to match the source document's metadata is used for dynamic conversion. Depending on the system configuration, other matching rules may also be applied (see ["Forced Conversions"](#) on page 1-4).

The Template Selection Rules page (see ["Template Selection Rules Page"](#) on page A-9) allows you to add, remove, and reorganize rules; specify the criteria (metadata) to base a rule on; and assign a template (or templates) to the rule.

A number of features have come together to form the Template Selection Rules page. You can add multiple rules and then change the order in which those rules will apply to source documents. You can select a number of metadata fields to base a rule on (and add even more fields using the configuration page). Lastly, you can assign a template (or templates) to the rule and then edit those templates using the Edit Template button.

3.2 Managing Your Template Rules

The top section of the Template Selection Rules page (see ["Template Selection Rules Page"](#) on page A-9) enables you to manage the template rules.

- [Adding a Rule](#)
- [Deleting a Rule](#)
- [Reordering the Rules](#)

3.2.1 Adding a Rule

To add a new template rule, complete these steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Template Selection Rules**.
The Template Selection Rules page is displayed (see "[Template Selection Rules Page](#)" on page A-9).
3. Type a name for your rule in the **New rule name** text box (under the Template Selection Rules heading).
4. Click **Add New Rule**.
When your rule is highlighted, you will notice that the criteria and template fields for the rule are blank. You can start entering the desired metadata criteria and template for this rule right away.
5. Click **Update** at the bottom of the Template Selection Rules page.

3.2.2 Deleting a Rule

To delete a template rule from the Template Selection Rules list, complete these steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Template Selection Rules**.
The Template Selection Rules page is displayed (see "[Template Selection Rules Page](#)" on page A-9).
3. Highlight the rule to be deleted and click **Delete Rule**.
4. Click **Update** at the bottom of the Template Selection Rules page.

Important: Deleting a rule will remove all of the settings (metadata criteria and template) for that rule. You cannot undo this operation.

3.2.3 Reordering the Rules

To change the order in which your template rules are processed, complete these steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Template Selection Rules**.
The Template Selection Rules page is displayed (see "[Template Selection Rules Page](#)" on page A-9).
3. Do either of the following:
 - To move a rule up the list, where it is prioritized over other rules, highlight the rule and click **Move Up**. Then click **Update**.
 - To move a rule down the list, where it will receive a lower priority, highlight the rule and click **Move Down**. Then click **Update**.

3.3 Assigning Metadata Criteria to a Rule

When assigning conversion templates to content items, you need to make sure that the metadata specified here matches the metadata assigned to your source documents. You can verify this by opening the content information page for your source documents in the Content Server.

To assign metadata to a template selection rule, complete these steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).

2. Click **Template Selection Rules**.

The Template Selection Rules page is displayed (see "[Template Selection Rules Page](#)" on page A-9).

3. Choose a metadata field from the first **Field** list (under the "Criteria for selected rule" heading). You may choose Type, Author, Title, Content ID, Title, or a number of other fields.

4. In the **Value** text box, enter the metadata that you would like your rule to target.

You can select the metadata value from the menu to the right of the Value text box. You can also use wildcards to specify a metadata value.

5. If desired, choose a second and third metadata field for your rule.

There will always be an "AND" relationship between the metadata fields, which means that only those content items that meet *all* criteria are converted by this rule.

The maximum number of criteria that you can specify for each rule is controlled by a setting on the Dynamic Converter Configuration Page (see "[Dynamic Converter Configuration Page](#)" on page A-2).

6. Click **Update** on the bottom of the Template Selection Rules page to update your rule.

3.4 Choosing a Template for a Rule

Your template selection rule is not complete until you choose a template for the rule. The template will ultimately drive the appearance of your converted documents.

To assign a template to a rule, complete these steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).

2. Click **Template Selection Rules**.

The Template Selection Rules page is displayed (see "[Template Selection Rules Page](#)" on page A-9).

3. Enter the content ID for the template in the **Template** text box (under the "Template and layout for selected rule" heading).

You can select a type of template (HTML Conversion, Classic HTML Conversion, or Script) from the **Template Types** menu, and then you can select your desired template from the **Available Templates** menu.

4. If you chose a Classic HTML Conversion template in the previous step, you may want to complement it with a layout template. If so, enter the content ID for the

layout template in the **Layout** text box (again, you may select the layout template from the **Available Layouts** menu).

5. Click **Update** to add the template to your rule.

Once you have created a template selection rule, assigned the appropriate metadata criteria to it, and selected a template (or templates) for the rule, you should verify your configuration settings on the Dynamic Converter Configuration page (see "[Dynamic Converter Configuration Page](#)" on page A-2). In particular, make sure that you have added the necessary file types to the Conversion Formats list.

See [Chapter 4, "Conversion Templates"](#) for more information about templates.

Conversion Templates

This section covers the following topics:

- ["About Templates"](#) on page 4-1
- ["Template Types"](#) on page 4-1
- ["Template Strategy"](#) on page 4-2
- ["Checking In a Template"](#) on page 4-2

See also:

- [Chapter 5, "HTML Conversion Templates"](#)
- [Chapter 6, "Classic HTML Conversion Layout Templates"](#)
- [Chapter 7, "Script Templates"](#)

4.1 About Templates

Much of the power, flexibility, and complexity of Dynamic Converter is bound up in its use of templates to drive the conversion process. Templates give you immense control over the visual and navigational properties of the converted web page.

A template is a plain-text HTML or XML file that may include special tags which allow template writers to insert, repeat through, condition on, and link to various elements in the source document. You can associate these sets of formatting instructions with one or multiple content items that are stored in the Content Server. When you assign a template to your content items (on the [Template Selection Rules](#) page (see ["Template Selection Rules Page"](#) on page A-9), you are controlling the way your content items will appear as web pages.

When users click the **(HTML)** link (generated by Dynamic Converter) for a content item, a dynamic conversion takes place using the template associated with that content item (see ["Dynamic Converter Process"](#) on page 1-2).

4.2 Template Types

There are four types of templates available in Dynamic Converter:

- **Classic HTML Conversion templates:** Classic HTML Conversion templates (formerly known as GUI templates) are written in XML (Extensible Markup Language) and are designed for use with the Dynamic Converter Classic HTML Conversion Editor. You can use the Classic HTML Conversion Editor to make changes in these templates and view them in real time. Classic HTML Conversion

templates have the *.ttp* file extension. See [Chapter 5, "HTML Conversion Templates"](#) for more information.

- **HTML Conversion templates:** The HTML Conversion Editor's primary goal is producing faithful representations of source files using the HTML, GIF, JPEG, and PNG formats. Using a C API and a powerful, customizable XML file, you can use the HTML Conversion Editor to set various options that affect the content and structure of the output. The HTML Conversion Editor is Java-based and can run in any browser instance where a JRE is present. See [Chapter 5, "HTML Conversion Templates"](#) for more information.
- **Classic HTML Conversion Layout templates:** Layout templates are designed to complement GUI templates in that they control the overall page layout for converted content items. A layout template can be used to create a common set of borders, site navigation, or a company logo on each converted web page. It can also be used to maintain the Content Server look and feel with links to Home, Search, etc. Layout templates typically contain HTML code (especially HTML tables), tokens (which represent GUI template settings), and Idoc Script or a different scripting language. See [Chapter 6, "Classic HTML Conversion Layout Templates"](#) for more information.
- **Script templates:** Script templates are text-based conversion templates that apply a set of scripted rules to your converted documents. They are plain-text files that must be hand-coded with elements, indexes, macros, pragmas, and Idoc Script. Changing script templates requires a knowledge of the language that they were written in. Script templates have the *.hcst* file extension. See [Chapter 7, "Script Templates"](#) for more information.

For more information on the differences between HTML Conversion templates and script templates, as well as suggestions for migrating, see ["Migrating From Script Templates to Classic HTML Conversion Templates"](#) on page 5-19.

4.3 Template Strategy

Through the use of templates, Dynamic Converter users have infinite flexibility in the way they can present converted documents. Users typically use one of the following three strategies to select a template:

1. Dynamic Converter is shipped with a number of sample templates, which are designed to meet different needs for Dynamic Converter users (polished navigation, simple HTML for document indexing engines, etc.). You can find the sample templates in the */ucm/Distribution/DynamicConverterSamples* directory.
2. With a bit more effort, you can modify one of the sample templates shipped with Dynamic Converter. Simple changes, such as adding graphics or static text, should be easily accomplished by someone with a willingness to experiment with these templates.
3. Advanced users may choose to write a template of their own design, customized specifically to their needs. Such templates can incorporate elements from a wide range of Web standards, such as Java. Needless to say, users who go this route should have strong technical skills at the outset.

4.4 Checking In a Template

You need to check a template into the Content Server before it can be assigned to a template selection rule (see [Chapter 3, "Template Rules"](#)) and used by Dynamic Converter in the conversion process.

To check in a template, complete the following steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Check In Existing Template**.
The Template Check-In Form is displayed (see "[Template Check-In Form](#)" on page A-12).
3. Specify all required metadata for the template.
Make sure that you select the correct template type. If you do not, a template may not be included in the list of available templates of a particular type. If that is the case, you need to open the content information page of the checked-in template and update its template type.
4. When you are done, click **Check In** to check the template file into the Content Server.

See the *Content Server User Guide* for more information about checking content into the Content Server.

HTML Conversion Templates

This section covers the following main topics:

- ["About Templates"](#) on page 5-1
- ["HTML Conversion Template Editor"](#) on page 5-3
- ["Classic HTML Conversion Template Editor"](#) on page 5-9

5.1 About Templates

A template is a set of formatting instructions you can associate with a source document. When you check a document into Content Server, you either associate it with a default conversion template, or you can create a new customized template.

The following template options are available:

- **HTML Conversion templates:** These are the newest template types, which can be configured in a cross-platform editor.
- **Classic HTML Conversion templates:** These were previously known as GUI templates. There is no direct migration path from the GUI templates to the HTML Conversion templates. If you select a Classic HTML Conversion template, you may also select a Classic HTML Conversion layout.
- **Script templates:** These run with default settings, and can be edited with a text editor.

After you have chosen a template type to associate with your document, and named the template, you can edit the template. There are two template editing utilities for customizing the appearance of native documents converted to an HTML format. These template editors are used to control the look and feel of the web pages you create.

- The HTML Conversion Editor is used to edit the HTML Conversion Templates.
- The Classic HTML Conversion Editor is used to edit the Classic HTML Conversion Templates and Classic HTML Conversion Layouts.

To turn a source document into a web page, you can use the default settings to perform a conversion. Alternatively, you can create a template, associate it with the document, and then edit the template, using one of the two template editor options.

The following sections describe tasks common to both template editors:

- ["Creating a New HTML Conversion Template"](#) on page 5-2
- ["Editing an Existing HTML Conversion Template"](#) on page 5-3

5.1.1 Creating a New HTML Conversion Template

Use the Dynamic Converter New HTML Conversion Template Form to create a new HTML Conversion Template. To access this page, click **Create New Template** on the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).

Figure 5–1 New HTML Conversion Template Form

Dynamic Converter New HTML Conversion Template Form
 Administration --> Dynamic Converter Admin --> Dynamic Converter New HTML Conversion Template Form

* Content ID

* Type

* Title

* Filer

* Security Group

Template Format HTML Conversion Template Classic HTML Conversion Template

* Revision

Folder

Hidden

Comments

User Access List

Trash Delete Old Name

Trash Delete Location

Trash Delete Date

Trash Deleter

Note: See the *Content Server User Guide* for more information about checking content into the Content Server.

To create a new HTML Conversion template, complete the following steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Create New Template**.
 The New HTML Conversion Template form is displayed (see [Figure 5–1, "New HTML Conversion Template Form"](#)).
3. Select the template format: **HTML Conversion Template** or **Classic HTML Conversion Template**.
4. Specify all other required metadata for the template.

Note: The template type is set by default to HTML Conversion Template. The Classic HTML Conversion Template is the former GUI Template.

5. When you have completed the form, click **Check In** to check the HTML Conversion template file into the Content Server.

After checking a new HTML Conversion template into the Content Server, you can edit it using the Template Editor (see "[Editing an Existing HTML Conversion Template](#)" on page 5-3).

5.1.2 Editing an Existing HTML Conversion Template

The HTML Conversion Template Editor requires Internet Explorer on a Windows XP or greater system. To edit an existing HTML Conversion Template or a Classic HTML Conversion template (that is, one that is already checked into the Content Server), complete the following steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).

2. Click **Edit Existing Template**.

The Edit Templates Page is displayed (see "[Edit Templates Page](#)" on page A-14).

3. Select a template from the list of HTML Conversion templates in the Content Server.

If a known HTML Conversion template is not included in the list of available templates, then it was most likely not assigned the correct HTML Conversion Template type when it was checked into the Content Server (see "[Checking In a Template](#)" on page 4-2). You then need to open the content information page of the checked-in template and update its template type.

The Edit Template button does not become available until you specify the name of an existing template.

4. Click the **Edit Template** button. The HTML Conversion Template Editor is downloaded to your machine. With some browsers, such as Firefox, you may be prompted for how to handle the file `dc_hcmapedit.jnlp`. The correct way to open this file is with Java(TM) Web Start Launcher (default).

The Template Editor is started. If you have not run the editor before, it is installed first and you may need to confirm a few prompts.

You can now edit the HTML Conversion template in the Template Editor.

You can also edit an existing HTML Conversion template from the Template Selection Rules page (see "[Template Selection Rules Page](#)" on page A-9).

Note: The Template Editor comes with its own extensive help system, which can be called from the application's user interface.

5.2 HTML Conversion Template Editor

This section provides a description of the HTML Conversion Template Editor. More detail can be found in the *Dynamic Converter Template Editor Guide*.

The HTML Conversion Editor allows you to set various options that affect the content and structure of the output. The HTML Conversion Editor is Java-based and can run in any browser instance where a JRE is present.

The following topics are covered in this section:

- "[Formatting Different File Types](#)" on page 5-4

- ["Adding Document Properties"](#) on page 5-4
- ["Adding Text Elements"](#) on page 5-5
- ["Adding Navigation Elements"](#) on page 5-5
- ["Configuring HTML Settings"](#) on page 5-5
- ["Adding Output Markup Items"](#) on page 5-6
- ["Adding Output Text Formats"](#) on page 5-6
- ["Adding Format Mapping Rules"](#) on page 5-7
- ["Adding Output Page Layouts"](#) on page 5-8
- ["Previewing Your Content"](#) on page 5-8
- ["Saving Your Template"](#) on page 5-9

5.2.1 Formatting Different File Types

The top item in the left-hand navigation pane of the HTML Conversion Editor allows you to set up custom formatting for different file types. Each file type uses a layout, either the default layout, or one created under Output Page Layouts. The exported files will use the same template as the root conversion.

These file types have slightly different options for formatting:

- **Text/Word Processing** - Allows you to set options for bullets, footnotes and endnotes, handling character styles and embedded graphics, and setting pagination.
- **Spreadsheets** - Allows you to set up section formatting and labeling, display gridlines, and size embedded graphics.
- **Presentations** - Allows you to set up section formatting and labeling, and size slides.
- **Images** - Allows you to set up section formatting and labeling, and size images.
- **Archives** - Allows you to display either Filenames (the names of files and folders in the archive will be output) or Decompressed files (the filenames will be output as links to the exported files). The exported files use the same template as the root conversion.
- **Database** - Allows you to set up section formatting and labeling, and set the number of records per page.

5.2.2 Adding Document Properties

This item allows you to add predefined and custom properties. You can assign default values, metatag names, and output format to these properties

By default, no document properties are defined. In order to include them in the output from the conversion, each desired document property must first be defined here. They must then be added to the output from the conversion by inserting them into page layouts defined in the Output Page Layouts item.

The most common predefined properties are as follows:

- Primary author
- Title
- Subject

- Keywords
- Content

Several more less common predefined properties are also available.

For a custom property, you can create a descriptive name, and then assign default values, metatag names, and output formats.

5.2.3 Adding Text Elements

Text elements allow the user to insert strings into the output. Each text element is defined as a name-value pair with an optional output format that will be used to format the text.

If an output format is not specified, the text will be inserted into the output as-is, with no additional markup.

5.2.4 Adding Navigation Elements

Navigation elements allow you to have navigation links generated in the output. There are three kinds of navigation elements:

- **Document Navigation** - This allows you to link to various items in the source document based on the document's structure. A common example of how one would use this type of navigation is to create links to all the paragraphs marked with outline level 1 (such as "Heading 1" paragraphs) in the document. Before using this form of navigation, link mapping rules must first be added. Link mapping rules establish which parts of the input document will be used to create links.
- **Page Navigation** - This provides a way to link to certain key pages in the output (first page, next page, etc). It also provides a way to link to external pages.
- **Section Navigation** - This provides navigation for multi-section documents, such as spreadsheets and presentations.

Once you have added one of these, the left hand side of the editor displays expanded levels. You can specify information about the link, link set markup and formatting, and create link mapping rules. Link mapping rules allow you to match on a paragraph outline level or paragraph style name in order to generate navigation based on these two aspects of the source document. Once you define rules, click back on the Link Mapping Rules page to determine the sequence of the rules. The mapping rules are ordered so that the first rule that matches is the one that is applied.

5.2.5 Configuring HTML Settings

There are six major categories that you can configure:

- **HTML Settings** - Allows you to set the HTML DOCTYPE and Language string.
- **CSS options** - Allows you to specify whether Cascading Style Sheet ("CSS") formatting will be used, and if so, the method of CSS presentation. By default, the CSS is embedded in the HTML of each output file. You may also choose to output CSS styles in a separate file. The external stylesheet option allows you to specify a stylesheet with user-generated styles that will be referenced by the conversion.
- **Character set** - Allows you to specify which character set should be used in the output file. Source documents that contain characters from many character sets will look best only when this option is set to Unicode or UTF-8. You may also

select a character to be used when a character cannot be found in the output character set (unmappable).

- **Graphics output** - Allows you to specify the format of the graphics produced by the technology: GIF, JPG, PNG, or none. Other options in this section allow you to specify quality and sizing of the graphic output.
- **Link options** - This option allows you to specify how the browser should select which frame or window in which to open source document links. This value is used for the target attribute of the links the technology generates. This target value will be applied to all such links encountered in the source document.
- **Output formatting** - This option causes the technology to write new blank lines to the output strictly to make the generated HTML more readable and visually appealing. While setting this option will make it easier to read the generated markup in a text editor, it does not affect the browser's rendering of the document. You can also include information about source document style names and how they are mapped, and the user can see what format has been mapped to a particular paragraph or text sequence by mousing over it.

5.2.6 Adding Output Markup Items

Markup items are HTML fragments that may be inserted directly into the output HTML as part of a page layout (see "[Adding Output Page Layouts](#)" on page 5-8). Each markup item is a name/value pair. The name is what will appear in the screens for editing page layouts. The value is a block of HTML that will be inserted into the output HTML wherever the markup item appears in a page layout.

Click the **Add** button and specify a **Name** to use for referencing this piece of markup. Then enter the HTML into the **Markup** text box.

5.2.7 Adding Output Text Formats

Output text formats define text and formatting attributes of output document text. This allows you to standardize the look of the output despite differing formatting styles used by the various authors of the source documents. Text formats are only applied to text from word processing files. They cannot be used to change the formatting of text that is rendered as part of any graphics generated by the conversion. They are also not applied to text inside spreadsheets.

1. Click the **Add** button to display the Markup tab. Then specify a **Name** to use for referencing this format.
2. Under **Tag name**, enter the HTML paragraph level tag to put around paragraphs using this format. Note that any tag name may be entered here, whether it is legal or not. Only the tag name should be entered, not the surrounding angle brackets ("**<**" and "**>**"). The paragraph tag ("**p**") is the default.
3. Under Custom Attributes, you can enter attributes that apply to the tag whose name was specified by the **Tag name** option above. To set the name and value of the new attribute, just click on them in the Custom Attributes table.
4. Custom Markup allows you to enter HTML and/or regular text that will be inserted before and/or after every paragraph using this format.
5. Other formatting options on the Markup tab include inserting new lines into the HTML before the paragraph to make it easier to view the HTML of the output of the conversion (only written if the **Format HTML source for readability** option is set on the Output Pages screen); specifying that a new output page is created every time this format is applied to a paragraph; and whether or not the first

instance of this format should start a new page (to help avoid empty or mostly empty pages at the beginning of the output).

6. On the Formatting tab, you can choose how to specify the formatting for paragraphs. If you click on the **Use external CSS class** option, a text field becomes available in which you must enter the name of a class from an external CSS file. The URL of the external CSS file is specified with the **External user stylesheet** option set on the Output Pages page.

If you do not specify an external stylesheet, you can choose to format the document by observing the original source document formatting or forcing other formatting options. Character, Paragraph and Border formatting for an array of options can be set to one of four values:

- Always off - Forces the attribute to always be off when formatting the text.
- Always on - Forces the attribute to always be on when formatting the text.
- Inherit (default) - Takes the state of the attribute from the source document. In other words, if the source document had the text rendered with bold, then the technology will create bold text.
- Do not specify - Leave the formatting unspecified.

5.2.8 Adding Format Mapping Rules

Once you have defined text formats, you must define rules to map output text formats to output text.

1. Select Format Mapping Rules and click **Add Format Mapping Rule**.
2. In the **Format** drop-down box, select one of your defined text formats.
3. In the **Match on** drop down box, you may select one of the following paragraph formatting options for the rule to check:
 - Outline level - Match the outline level specified in the source document. Application-predefined "heading" styles typically have corresponding outline levels applied as part of the style definition.
 - Style name - Match the paragraph or character style name.
 - Is footnote - Match any footnote.
 - Is endnote - Match any endnote.
 - Is header - Match any document header text.
 - Is footer - Match any document footer text.
4. For the Paragraph outline level, if Match on above is set to Outline level, then this defines which outline level to match. This option cannot be set/is ignored for all other matching rules.
5. If Match on above is set to Style name, then this defines which source document paragraph or character style name to match. When matching on style names, you must supply a style name here, and no default value is provided. The name must exactly match the style name from the source document. Style name matching is done in a case-sensitive manner. This option cannot be set/is ignored for all other matching rules.
6. When you have finished defining rules, you can go back to the Format Mapping Rules page and click on **Move Up** or **Move Down** to arrange the sequence in

which the rules are checked. The mapping rules should be ordered so that the first rule that matches is the one, and only one, that is applied.

5.2.9 Adding Output Page Layouts

The Output Page Layouts section allows you to define the content of a set of output files. Page layouts are used to organize how the various pieces of the output are arranged.

1. Click **Add** to add a new output page layout. On the next page, enter a name to use to refer to this layout (required). Once this has been done, click on the left side of the editor. The name you have just entered is displayed in the tree view. Click on this to expand the levels underneath.
2. Click the box in front of **Include navigation layout** if you want to generate a single file containing markup and links to the document content specified in the page layout. This allows the user to create a "table of contents" page. You will need to define a navigation element under the Navigation Layout.
3. The first item under the name of your output page layout is **<title> Source**. This lets you select where to get the value to use for the HTML **<title>** tag. Select Section Name, Text Element, Property, or Output Text Format (the last three must be previously defined). Click back on the **<title> Source** page to order the sequence of these sources.
4. The Navigation Layout triggers the creation of a separate file with nothing but links to the actual document content. In order to generate this, you must have previously defined either a Document Navigation, Page Navigation, or Section Navigation element under Generated Content (see "[Adding Navigation Elements](#)" on page 5-5). In the expanded level under Navigation Layout, you can further select Markup Items to be placed in the Head and/or Body of the navigation page.
5. The top level of the Page Layout section lets you set pagination options. The six options under Page Layout let you define how output documents are arranged. These six options are as follows:
 - Head - Items placed in the HTML **<head>** of each output file.
 - Page Top - Items to be placed at the top of each output page. For example, links to the first page, previous page and next page in the output.
 - Before Content - Items to be output before the document content.
 - Before Section - Items inserted before each section of a multi-section document. Note that this is not applicable to word-processing documents.
 - After Content - Items to be output after the document content.
 - Page Bottom - Items placed at the top of each output page; for example, a copyright notice.

After selecting items to display for these six options, click at the top level of each one to set the order in which they will appear .

5.2.10 Previewing Your Content

The HTML Conversion Editor provides two options for previewing your content. They are both located in the Tools menu at the top of the interface.

- **View XML Structure** - Click this option to display the XML structure viewer, which shows a text-based XML version of your chosen template options.

- **Set preview document** - Click this option to enter the Content ID of the source document, and then click **Preview Conversion**. Your browser will open and display how the current template settings would affect the converted output.

5.2.11 Saving Your Template

If you start to exit the template editor, you will be prompted to supply an XML filename and location to store your template.

5.3 Classic HTML Conversion Template Editor

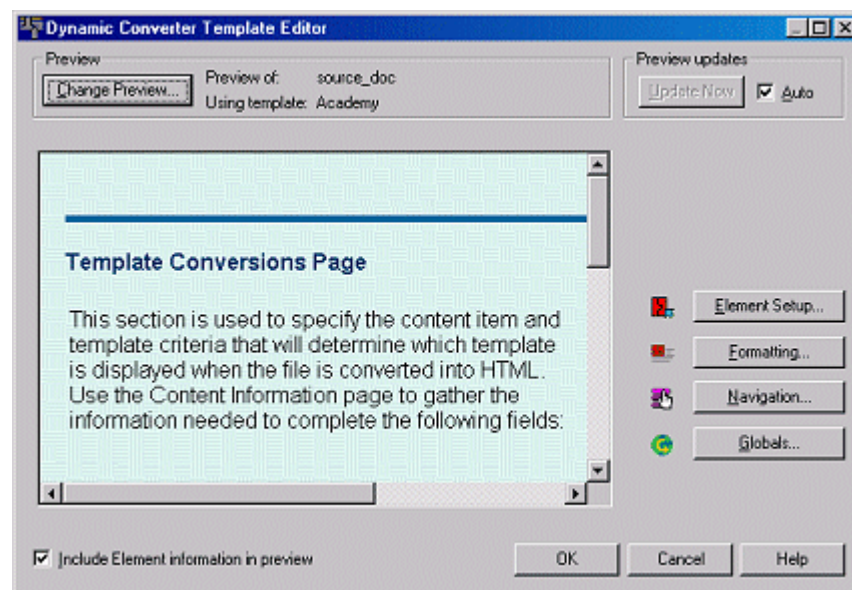
The following topics are covered in this section:

- ["Template Elements"](#) on page 5-10
- ["Sample Classic HTML Conversion Templates"](#) on page 5-11
- ["Migrating From Script Templates to Classic HTML Conversion Templates"](#) on page 5-19

You can select Classic HTML Conversion Editor on the Edit Templates page. The first time the **Edit Template** button is clicked on the Edit Templates page (see ["Edit Templates Page"](#) on page A-14) or Template Selection Rules page (see ["Template Selection Rules Page"](#) on page A-9), the Classic HTML Template Editor is downloaded onto the client machine. The Classic HTML Conversion Template Editor is an ActiveX control that must be run on Microsoft Windows with Internet Explorer 4.0 or higher present.

When you specify the name of a recognized Classic HTML Conversion template on the Edit Templates page or Template Selection Rules page, the **Edit Template** button is activated. Click this button to open the Template Editor. The Template Editor features a template preview area and four editing buttons: **Element Setup**, **Formatting**, **Navigation**, and **Globals**. Each button opens a property sheet that contains numerous settings for your template; all of which can be edited in a graphical user interface.

Figure 5–2 Classic HTML Conversion Template Editor



Each source document that you plan to convert to a web page using Dynamic Converter contains individual formatting attributes. You may have prepared styles in your source documents and assigned those styles to a specific typeface or font. Or, you may have manually formatted the content inside each source document (for example, headings in 14-point bold, sub-headings in 12-point italic, etc.). Classic HTML Conversion templates in Dynamic Converter can recognize both.

Classic HTML Conversion templates and the Template Editor can recognize styles as well as manually formatted documents. Once an element is assigned to these individual parts of your source document, you can then begin modifying the appearance and functionality of those elements using the Template Editor. When source documents are converted into web pages, it is the elements (stored in the template) that ultimately control how the web page will appear.

The Template Editor includes a very useful screentip Element, where you can place your cursor above a piece of text in the preview document and see the element that has been assigned to that text.

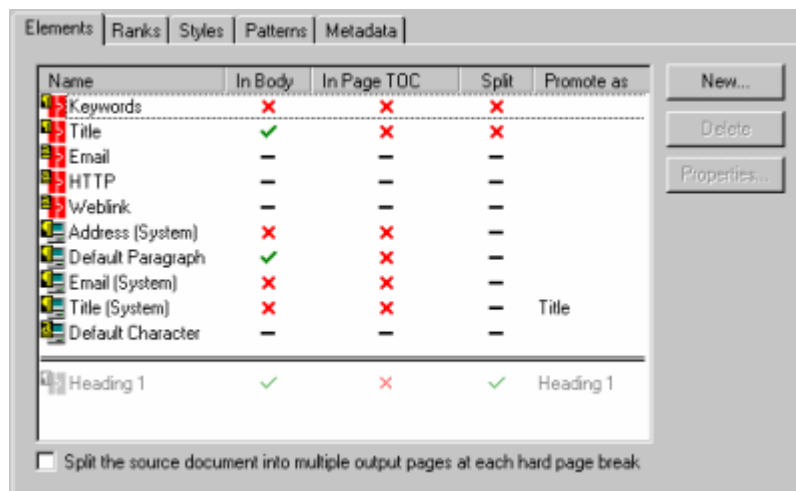
Many of the settings in the Template Editor apply to a single element. The more you define each element, the more control you have exert over the converted web page; all without ever touching the source document.

Note: The Template Editor comes with its own extensive help system, which can be called from the application's user interface.

5.3.1 Template Elements

Nearly every source document has a title, a heading, and body text. Each one will likely have a unique font size and weight. The Template Editor can be used to assign unique elements to each piece of text and save that information in the Classic HTML Conversion template.

Figure 5-3 *Elements in Template Editor*



Elements are created from ranks, styles, or patterns:

- Rank:** Used by the Template Editor to identify the structure of the content of a document based on the hierarchy of that content. Ranks can be used with patterns in Element Setup to prepare a template for editing.

- **Style:** A set of formatting characteristics with an assigned name that defines how text appears in a document. Styles can be assembled together to make up a style sheet or Cascading Style Sheet (CSS).
- **Pattern:** A set of text attributes in a source document that the Template Editor can identify and associate with an element. If a manually-formatted source document has headings in Arial, 18-point, bold, you can base a pattern on these attributes and associate this pattern with an element. You can then use this element to format the content associated with the pattern.

You will find that styles in your source documents are the most useful and manageable for conversion purposes. As such, you should first try to implement styles in your source documents and perhaps distribute a style sheet to your content contributors.

Dynamic Converter templates are designed to be interchangeable with other Content Server related products, such as Content Publisher. The features that apply to reference pages in a web publication do not apply and will not work in Dynamic Converter. (An example of this would be adding a table of contents for multiple source documents.)

Note: The Template Editor includes a separate and comprehensive online Help system (which is downloaded with the Template Editor). Each dialog box and property sheet in the Template Editor includes a Help button that describes that particular Element. To access these topics, click **Help**.

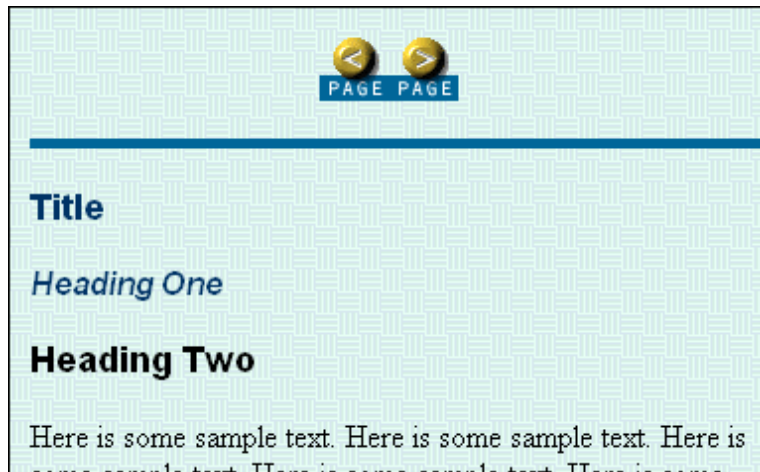
5.3.2 Sample Classic HTML Conversion Templates

Dynamic Converter comes with a number of sample Classic HTML Conversion templates that you can check into Content Server and begin using with the Template Editor right away.

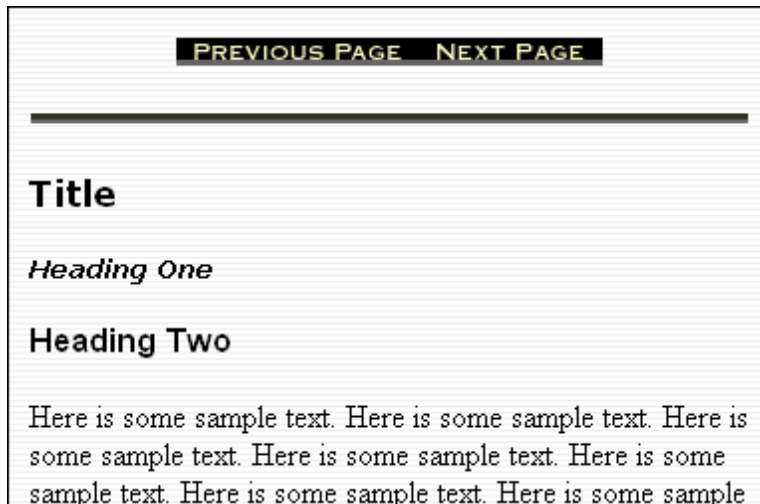
- [Academy](#)
- [Acclaim CSS](#)
- [Account](#)
- [Adagio CSS](#)
- [Administration](#)
- [Analysis](#)
- [Archive CSS](#)
- [Blank](#)
- [Business](#)
- [Ceremonial](#)
- [Courtesy](#)
- [Executive](#)
- [Introduction CSS](#)
- [Lotus 1-2-3](#)
- [Lotus Freelance](#)
- [MS Excel](#)

- [MS PowerPoint](#)
- [Purple Frost](#)
- [Retrofied! CSS](#)

5.3.2.1 Academy

A screenshot of a web page template with a light blue background and a white grid pattern. At the top, there are two yellow circular arrows pointing left and right, with the text 'PAGE PAGE' in a blue box below them. Below a horizontal blue line, the text reads: **Title**, *Heading One*, **Heading Two**, and a paragraph of sample text: 'Here is some sample text. Here is some sample text. Here is...'

5.3.2.2 Acclaim CSS

A screenshot of a web page template with a white background and a light grey grid pattern. At the top, there are two yellow rectangular buttons with black text: 'PREVIOUS PAGE' and 'NEXT PAGE'. Below a horizontal black line, the text reads: **Title**, *Heading One*, **Heading Two**, and a paragraph of sample text: 'Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text.'

5.3.2.3 Account

previous page | next page

Title

Heading One

Heading Two

Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text.

5.3.2.4 Adagio CSS

Previous Page Next Page

Title

Heading One

Heading Two

Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text.

5.3.2.5 Administration

PREVIOUS PAGE | NEXT PAGE

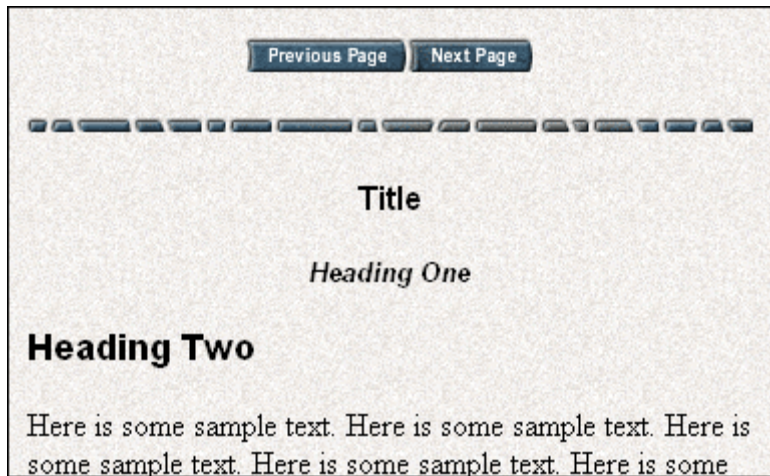
Title

Heading One

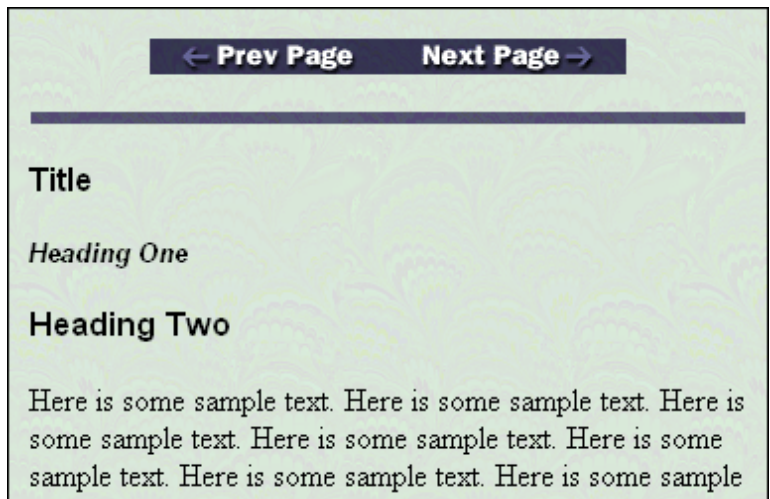
Heading Two

Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text.

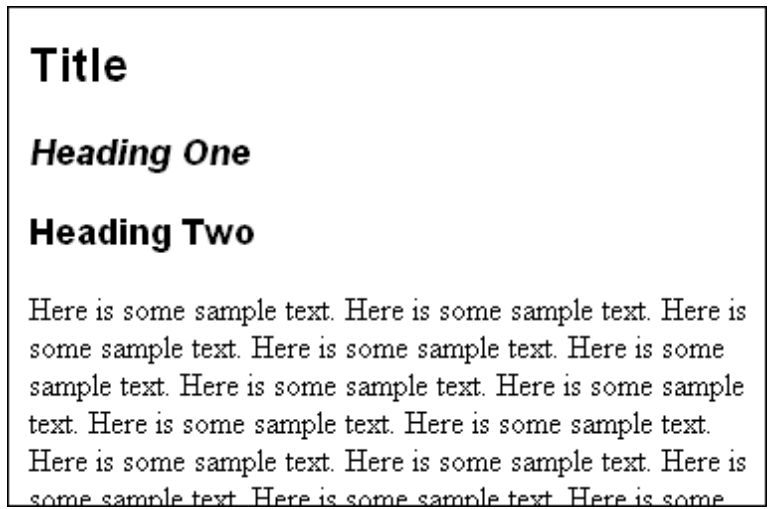
5.3.2.6 Analysis



5.3.2.7 Archive CSS

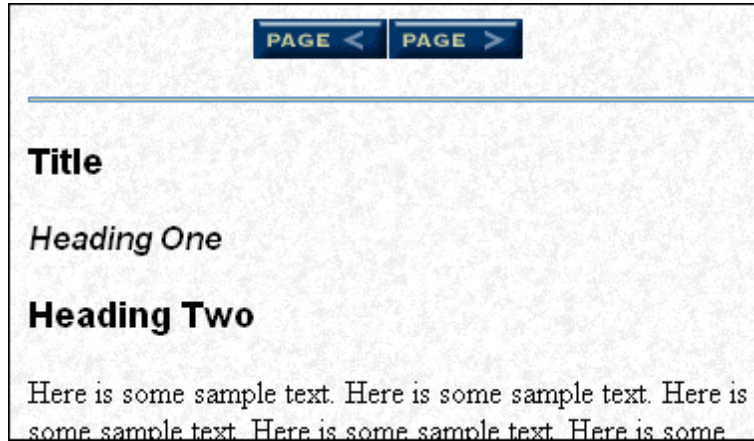


5.3.2.8 Blank



Note: This is the default template.

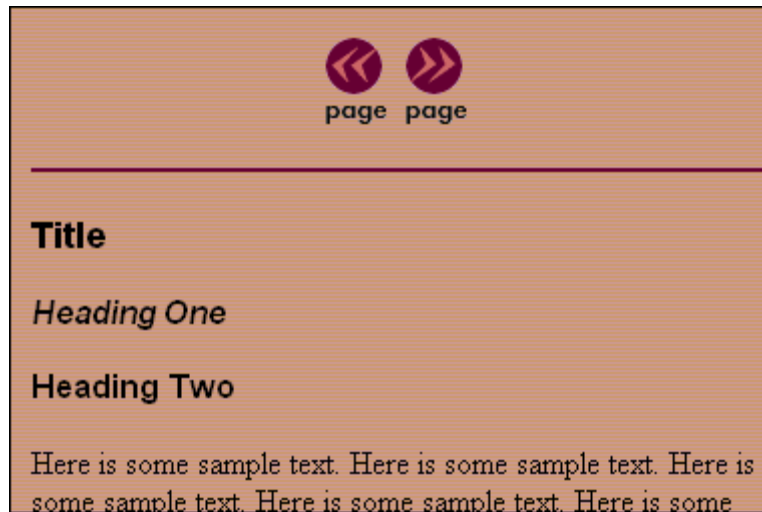
5.3.2.9 Business



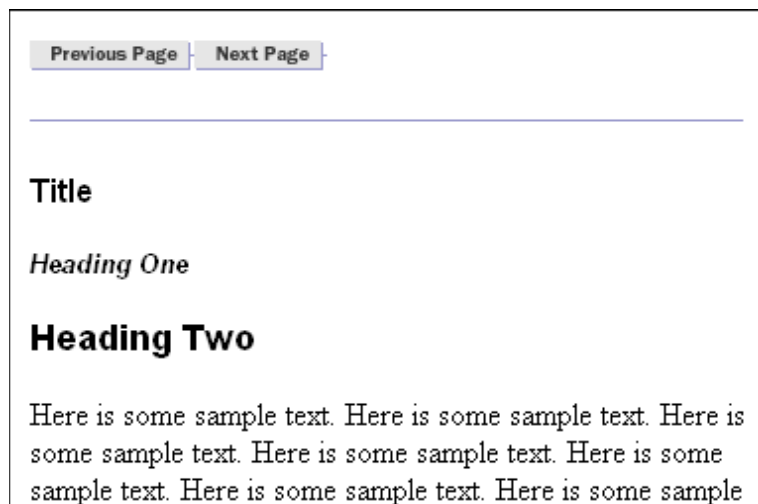
5.3.2.10 Ceremonial



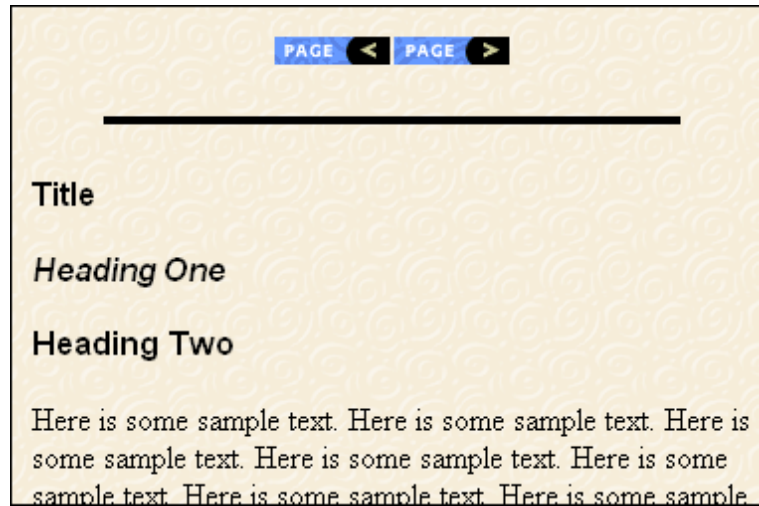
5.3.2.11 Courtesy



5.3.2.12 Executive



5.3.2.13 Introduction CSS



5.3.2.14 Lotus 1-2-3

PAGE < PAGE >

Sheet1

70.55475	77.47401							
53.3424	1.401764							
57.95186	76.07236							

5.3.2.15 Lotus Freelance

< PAGE PAGE >

70.55475	77.47401						
53.3424	1.401764						
57.95186	76.07236						
28.95625	81.44901						
22.4040	72.00070						

5.3.2.16 MS Excel

[previous page](#) | [next page](#)

Title

Heading One

Heading Two

Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text.

5.3.2.17 MS PowerPoint

< PAGE > PAGE

Click to add title

5.3.2.18 Purple Frost

[PAGE](#) [PAGE](#)

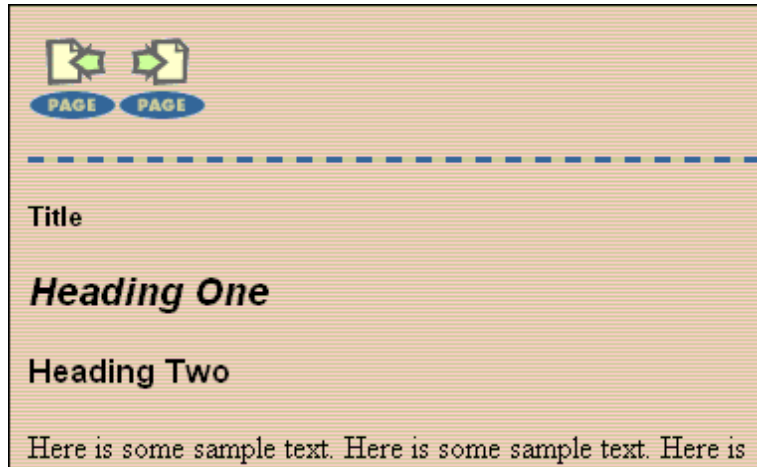
Title

Heading One

Heading Two

Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text. Here is some sample text.

5.3.2.19 Retrofied! CSS



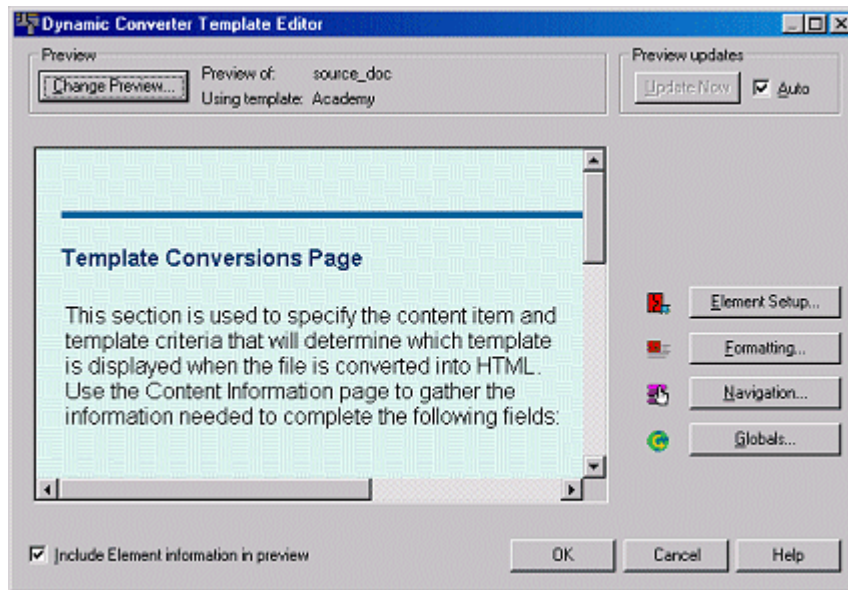
5.3.3 Migrating From Script Templates to Classic HTML Conversion Templates

The script templates (see [Chapter 7, "Script Templates"](#)) in earlier versions of Dynamic Converter were hand-coded text files that contain elements, macros, pragmas, indexes, and Idoc Script. A basic script template might look something like this:

```
<HTML>
<BODY>
<P>Here is the document you requested.
{## INSERT ELEMENT=Property.Title} by
{## INSERT ELEMENT=Property.Author}
<P>Below is the document itself
{## INSERT ELEMENT=Body}
</BODY>
</HTML>
```

Dynamic Converter now also supports XML-based Classic HTML Conversion templates designed for use with the GUI-driven Template Editor (see "[Classic HTML Conversion Template Editor](#)" on page 5-9).

A basic Classic HTML Conversion template might look something like this in the Template Editor:

Figure 5–4 Classic HTML Conversion Template in Template Editor

As a result of these differences, there is no automated upgrade process from the previous script templates to the current version Classic HTML Conversion templates. We can, however, recommend a migration path so that you can begin using the powerful Classic HTML Conversion templates in Dynamic Converter.

5.3.3.1 Updating an Old Template

To update a script template from an earlier version of Dynamic Converter to the Classic HTML Conversion template format, complete the following steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Create a new Classic HTML Conversion template.
3. Click **Template Selection Rules** on the Dynamic Converter Admin page.
The Template Selection Rules page is displayed (see "[Template Selection Rules Page](#)" on page A-9).
4. Highlight the rule associated with your previous template (the script template) and then scroll down to the "Template and layout for selected rule" area.

Note: Rules that were created in an earlier version of Dynamic Converter (prior to version 6.1) will appear as a numbered rule in this version of Dynamic Converter. You can continue using that rule or delete it and re-create the rule in Dynamic Converter 11gR2 (you cannot rename a rule).

You may want to modify the criteria assigned to your previous rule using the additional metadata fields available in Dynamic Converter. See "[Assigning Metadata Criteria to a Rule](#)" on page 3-3.

5. From the **Available Templates** menu, select the Classic HTML Conversion template that you created in Step 2 (templates are listed by content ID).
6. Click **Edit Template** to open the Classic HTML Conversion Template Editor.

7. In the Template Editor, click **Change Preview** to select a source document (by content ID) to preview your template with.
8. Re-create the settings from your previous script template using the Template Editor (see "[Classic HTML Conversion Template Editor](#)" on page 5-9). This will likely be the most time-consuming part of the migration process. You may want to open another web browser and preview a dynamically converted document that used the previous template, so that you can compare the templates as you work. Click **OK** to close the Template Editor when you are finished making changes.
9. Enter the content ID of your previous script template in the **Layout** field (so that you can turn a former script template into a layout template that is used with the Classic HTML Conversion template).
10. Click **Update** to associate your new Classic HTML Conversion template and layout template with your template selection rule.
11. Search for the layout template (former script template) in the Content Server, check it out, and open it in a text editor.

Make the following changes:

- Insert the following token at the top of your file, before the first <HTML> tag:

```
<!--TRANSIT - CUSTOMLAYOUT(TOP) -->
```

- Insert the following token between the HTML <HEAD> tags:

```
<!--TRANSIT - CUSTOMLAYOUT(HEAD) -->
```

- Insert the following token in the HTML <BODY> tag::

```
%%TRANSIT-BODYATTRIBUTES%%
```

- Replace your existing Insert Body Element tag with the following token (this token will replace most of your previous element settings):

```
<!-- TRANSIT - CUSTOMLAYOUT(BODY) -->
```

- Remove all references to elements, macros, pragmas, and indexes.
 - Leave Idoc Script tags in place (those that call outside files or services).
12. Save your new layout template and check it into the Content Server.

Note: Unlike script templates, layout templates in the Content Server do not require the HCST file extension.

5.3.3.2 Sample of Newly Converted Template (From a Pre-6.0 Version)

To update an earlier Dynamic Converter script template (prior to version 6.0) to the current version Classic HTML Conversion template, you will need to recreate your original template settings in the Template Editor (see "[Classic HTML Conversion Template Editor](#)" on page 5-9) and then turn your previous script template into a layout template. While all Idoc Script tags can remain, you will need to remove the syntax for elements, macros, pragmas, and indexes. These values are replaced with template tokens, in particular the CUSTOMLAYOUT(BODY) token, which represents nearly all of the settings made in the Template Editor.

The following example illustrates a very simple script template created in an earlier version of Dynamic Converter (prior to version 6.0) that is turned into a layout

template in the current version. All element formatting, of course, must be recreated in the new Template Editor. (**Bold** text indicates a tag that is replaced.)

Example 5-1 Original Script Template (Example)

```
<html>
<head>
<title>
{## insert element=property.title suppress=tags}
</title>
<${defaultPageTitle="Converted Content"$}>
<${include std_html_head_declarations$}>
</head>
<body>
<${include body_def$}>
<${include std_page_begin$}>
<${include std_header$}>
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr><td>
{## INSERT ELEMENT=Property.Title}
</td></tr>
<tr><td>
{## INSERT ELEMENT=Body}
</td></tr>

/table>
<${include std_page_end$}>
</body>
</html>
```

Example 5-2 Migrated Layout Template (Example)

```
<!-- TRANSIT - CUSTOMLAYOUT(TOP) -->
<html>
<head>
<!-- TRANSIT - CUSTOMLAYOUT(HEAD) -->
<${defaultPageTitle="Converted Content"$}>
<${include std_html_head_declarations$}>
</head>
<body %%TRANSIT-BODYATTRIBUTES%%>
<${include body_def$}>
<${include std_page_begin$}>
<${include std_header$}>
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr><td>
<!-- TRANSIT - CUSTOMLAYOUT(BODY) -->
</td></tr>
</table>
<${include std_page_end$}>
</body>
</html>
```

Classic HTML Conversion Layout Templates

This section covers the following topics:

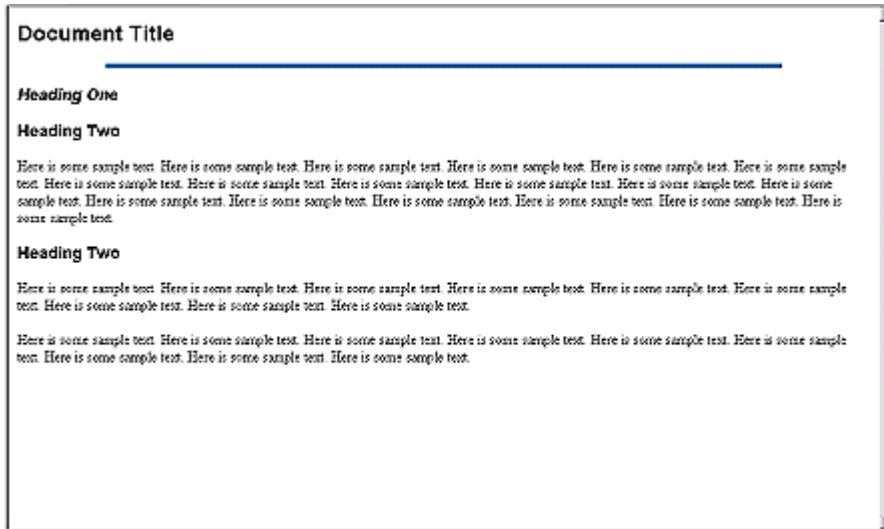
- ["About Classic HTML Conversion Layout Templates"](#) on page 6-1
- ["Layout Template Contents"](#) on page 6-2
- ["Tokens in Layout Templates"](#) on page 6-3
- ["Sample Layout Templates"](#) on page 6-3
- ["Creating a Layout Template for Your Content Items"](#) on page 6-5
- ["Associating a Layout Template With a Template Rule"](#) on page 6-6
- ["Specifying a Default Layout Template"](#) on page 6-6
- ["Including Scripts, Images, and CSS in a Layout Template"](#) on page 6-7

6.1 About Classic HTML Conversion Layout Templates

Layout templates can be used to complement Classic HTML Conversion templates (see [Chapter 5, "HTML Conversion Templates"](#)). They can be used to control the placement of items on a web page, in particular, the areas outside of the converted document. You can add shared borders, navigation, custom scripting, and much more in your layout template. You might use the layout template to create a common set of hyperlinks around your converted documents (such as "additional resources"), or you might prefer to maintain the Content Server look and feel around your documents using Idoc Script header and footer tags.

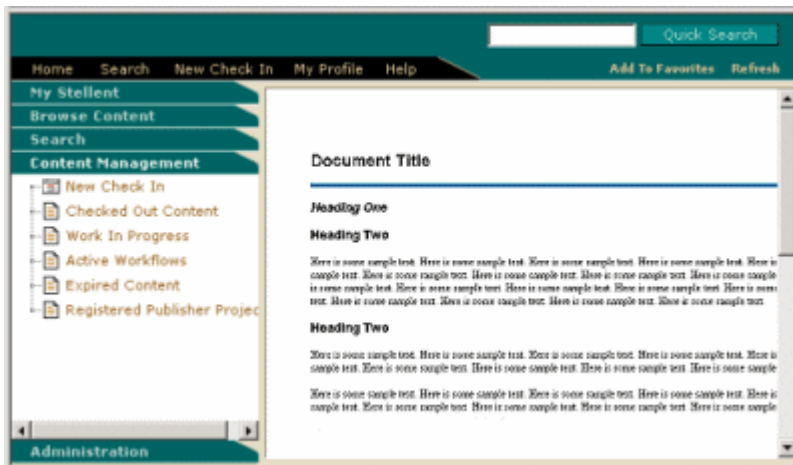
If you do not specify a layout template on the Template Selection Rules page (see ["Template Selection Rules Page"](#) on page A-9), your converted document will take up the entire web browser screen area when a user clicks the **(HTML)** link in the Content Server interface (see ["Viewing a Converted File"](#) on page 9-3).

Figure 6–1 *Converted Document Without Layout Template*



If you specify a layout template, such as the *default_layout.txt* sample, you can add a consistent look and feel around your content items.

Figure 6–2 *Converted Document With Layout Template*



6.2 Layout Template Contents

A typical layout template contains the following parts:

- HTML top and head information
- HTML tables (used to control page layout)
- Tokens for your template settings (see "[Tokens in Layout Templates](#)" on page 6-3)
- Idoc Script code (for various purposes)

When used together, you will find that you can fine tune the appearance of your converted documents on a global level, giving your online information a professional and consistent look and feel.

6.3 Tokens in Layout Templates

Tokens are placeholders or variables for the Classic HTML Conversion template settings that you create in the Template Editor. A layout template is used to control the placement of items around your converted content. If you wanted to include a particular TOP or HEAD setting from your Classic HTML Conversion template (keep in mind that layout templates are frequently used with Classic HTML Conversion templates), this would normally require you to copy and paste the information into your layout template (in the TOP or HEAD HTML tag). With a token, you can reserve that space for a Classic HTML Conversion template setting.

There are four tokens available:

- `<!--TRANSIT - CUSTOMLAYOUT(TOP)-->`

Place this token at the top of the layout template before the `<HTML>` tag. Your template could replace this value with an HTML declaration, such as the W3C document type identifier.

- `<!--TRANSIT - CUSTOMLAYOUT(HEAD)-->`

Place this token between the `<HEAD>` tags. Your template could replace this value with a web page title, meta tag keyword, and much more.

- `%%TRANSIT-BODYATTRIBUTES%%`

Place this token in the `<BODY>` tag. Your template could replace this value with a background color, text color, link behavior, and much more.

- `<!-- TRANSIT - CUSTOMLAYOUT(BODY) -->`

Place this token at the location where you would like your actual content items to appear on the web page. You will likely place this somewhere in the middle of your layout template. Your template will replace this value with each content item. This token can be used by itself to generate minimum HTML output so that the content item can be included in another web page. See [Chapter 8, "HTML Snippets"](#).

6.4 Sample Layout Templates

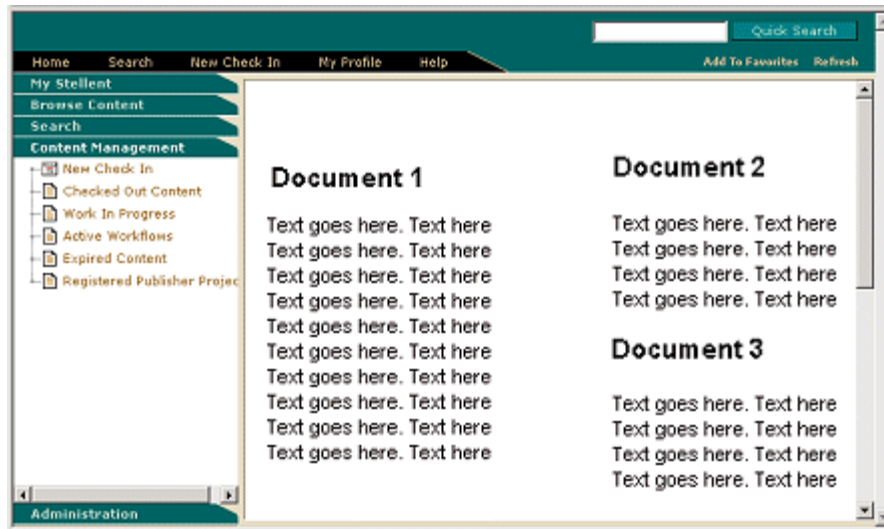
Dynamic Converter comes with a number of sample layout templates that you can check into Content Server and begin using right away.

The following sample layout templates are available:

- [default_layout.txt](#)
- [snippet_layout.txt](#)

6.4.1 default_layout.txt

The *default_layout.txt* template wraps Content Server borders and navigation around your converted documents using Idoc Script and HTML tables.

Figure 6–3 Default Layout

The *default_layout.txt* layout template contains the following code:

```
<html>
<head>
<!-- TRANSIT - CUSTOMLAYOUT (HEAD) -->
<$defaultPageTitle="Converted Content"$>
<$include std_html_head_declarations$>
</head>

<$include body_def$>
<$include std_page_begin$>
<$include std_header$>

<table border="0" cellpadding="0" cellspacing="0" width="550">
<tr><td>

<!-- TRANSIT - CUSTOMLAYOUT (BODY) -->

</td></tr>
</table>

<$include std_page_end$>

</body>
</html>
```

6.4.2 snippet_layout.txt

The *snippet_layout.txt* template places the converted document on a web page, by itself, without the top, head, or body HTML markup. The result is very similar to what happens when there is no layout template associated, but the advantage here is that you can easily pull this content into another web page, possibly a portal site, as an HTML snippet.

The *snippet_layout.txt* layout template consists of a single line of code:

```
<!-- TRANSIT - CUSTOMLAYOUT (BODY) -->
```

This is a token that displays the actual content item on the web page. Since it used by itself here, minimum HTML output is generated which can be included in another web page or HTML snippets (see [Chapter 8, "HTML Snippets"](#)).

Snippet Demo

The *snippet_demo.hcst* sample includes the basic ingredients for a portal-style web page that draws information (HTML snippets) from other content items stored in the Content Server, while preserving the borders and navigation.

The *snippet_demo.hcst* sample contains the following code:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<title>This is my incDynConv script test page</title>
<meta name="GENERATOR" content="Dynamic Converter">
<$defaultPageTitle="Converted Content"$>
<$include std_html_head_declarations$>
</head>

<$include body_def$>
<$include std_page_begin$>
<$include std_header$>
```

This is a sample page that shows how to include multiple snippets of dynamically
converted content on a single page using the new Idoc function `incDynamicConversion`.

```
<table border="1" cellpadding="0" cellspacing="0" width="550">
  <tr>
    <td>
      <$incDynamicConversion("<source_contentID_1>", "latest", "<template_
contentID_1>", "snippet_layout")$>
    </td>
    <td>
      <$incDynamicConversion("<source_contentID_2>", "latest", "<template_
contentID_2>", "snippet_layout")$>
    </td>
  </tr>
  <tr>
    <td colspan=2>
      <$incDynamicConversion("<source_contentID_3>", "latest", "<template_
contentID_3>", "snippet_layout")$>
    </td>
  </tr>
</table>

<$include std_page_end$>

</body>
</html>
```

6.5 Creating a Layout Template for Your Content Items

To create and edit a layout template, complete the following steps:

1. Create a new layout template in a text editor or WYSIWYG tool. For information on the contents of a layout template, see ["Layout Template Contents"](#) on page 6-2.

Tip: Dynamic Converter is shipped with some sample layout templates that you can use as a starting point (see ["Sample Layout Templates"](#) on page 6-3).

2. Open the Dynamic Converter Admin page (see ["Dynamic Converter Admin Page"](#) on page A-1).
3. Click **Check In Existing Template** and follow the steps to check in an existing template (see ["Checking In a Template"](#) on page 4-2). Make sure that you choose **Layout Template** as the template type.
4. Return to the Dynamic Converter Admin page.
5. Associate your layout template with a template rule (see ["Associating a Layout Template With a Template Rule"](#) on page 6-6).

6.6 Associating a Layout Template With a Template Rule

You can associate a particular layout template with a template rule on the Template Selection Rules page (see ["Template Selection Rules Page"](#) on page A-9). In the example below, the template sample titled "default_layout" has been selected.

Figure 6-4 Selection of Layout Template on Template Selection Rules Page

To specify a layout template for a template rule, complete these steps:

1. Open the Dynamic Converter Admin page (see ["Dynamic Converter Admin Page"](#) on page A-1).
2. Click **Template Selection Rules**.
The Template Selection Rules page (see ["Template Selection Rules Page"](#) on page A-9).
3. Highlight the rule that you would like to specify a layout template for.
4. Enter the content ID for the layout in the **Layout** text box (under the "Template and layout for selected rule" heading). You can also select the layout template from the **Available Layouts** menu.
5. Click **Update** at the bottom of the page.

6.7 Specifying a Default Layout Template

In addition to associating a layout template with a specific template rule, you can also specify a default layout that is applied to all content items that do not match your defined template criteria. You specify the default layout on the Dynamic Converter

Configuration page (see "[Dynamic Converter Configuration Page](#)" on page A-2). In the example below, the template sample titled "default_layout" has been selected.

Figure 6–5 Default Layout Template on Dynamic Converter Configuration Page

The screenshot shows the "Dynamic Converter Configuration" page. It is divided into two main sections: "Default Template" and "Default Layout".

Default Template
 Template that will be used if none of the other selection rules match.
 This section contains three fields: "Template" (an empty text box), "Available Templates" (a dropdown menu), and "Template Types" (a dropdown menu currently showing "GUI Template").

Default Layout
 Layout that will be used if none of the other selection rules match.
 This section contains two fields: "Layout" (an empty text box) and "Available Layouts" (a dropdown menu).

To set the default layout template associated with your content items, complete the following steps:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Configuration Settings**.
 The Dynamic Converter Configuration page is displayed (see "[Dynamic Converter Configuration Page](#)" on page A-2).
3. In the **Layout** text box, under the Default Layout heading, enter the content ID for a layout template. You can also choose your desired layout template from the **Available Layouts** menu.
4. Click **Update** at the bottom of the page to enable your default templates.

6.8 Including Scripts, Images, and CSS in a Layout Template

The layout template that you associate with your content items may include references to other files, such as custom scripts, images, Cascading Styles Sheets (CSS), and more. In fact, if you have a number of script templates that were created in an earlier version of Dynamic Converter, you can copy the Idoc Script tags from those templates and paste them into the new layout template. See [Chapter 7, "Script Templates"](#) for more information on script templates.

Identifying the appropriate path to use for an included file can be a challenge because the location of each content item checked into the Content Server may change if its metadata changes (metadata ultimately determines the URL of a content item). As such, you will not know the address of a new content item until it is checked into the Content Server with assigned metadata.

In this type of environment, relative paths create immediate problems. You must use a path that will work from anywhere in the Content Server. See "[Relative URLs in Templates and Layout Files](#)" on page 10-4 for a list of solutions.

Note: To assign a default layout template to your content items, see ["Setting the Default Template"](#) on page 2-1. In earlier versions of Dynamic Converter (prior to version 6), default layouts were assigned as configuration variables in Content Server. You can now make this setting on the Dynamic Converter Configuration page (see ["Dynamic Converter Admin Page"](#) on page A-1).

Script Templates

This section covers the following topics:

- ["About Script Templates"](#) on page 7-1
- ["Elements"](#) on page 7-2
- ["Indexes"](#) on page 7-9
- ["Macros"](#) on page 7-12
- ["Pragmas"](#) on page 7-30
- ["Sample Script Templates"](#) on page 7-32
- ["Setting Script Template Formatting Options"](#) on page 7-42
- ["Breaking Documents by Structure"](#) on page 7-43
- ["Breaking Documents by Content Size"](#) on page 7-46
- ["Using Grids to Navigate Spreadsheet and Database Files"](#) on page 7-48

7.1 About Script Templates

Script templates are the text-based conversion templates that were primarily used in earlier versions of Dynamic Converter. They are plain-text files that must be hand-coded with elements, indexes, macros, pragmas, and Idoc Script. You can still use this template format in Dynamic Converter, but Classic HTML Conversion templates (see [Chapter 5, "HTML Conversion Templates"](#)) have, for the most part, replaced script templates.

Note: See the Content Server developer documentation for more information on Idoc Script.

The following is the code for a very simple script template:

```
{## unit}{## header}
<html>
<body>
{## /header}
<p>Here is the document you requested.
{## insert element=property.title} by
{## insert element=property.author}</p>

<p>Below is the document itself</p>
{## insert element=body}
```

```
{## footer}  
</body>  
</html>  
{## /footer}{## /unit}
```

The {## unit}, {## /unit}, {## header}, {## /header}, {## footer} and {## /footer} macros can be ignored for the moment. Their purpose is described in [Macros](#).

The remainder of the file is regular HTML code with the exception of three macros in the form {## insert element=xxx}. Dynamic Converter uses this template plus the source file to create its output. To accomplish this, Dynamic Converter reads through the template file, writing it byte for byte to the output file unless character mapping is performed on the template. This continues until the template contains a properly formatted macro. Dynamic Converter reads the macro and executes the macro's command. Usually this means inserting an HTML version of some element from the source file into the output file. Dynamic Converter then continues reading the template and executing macros until the end of the template file is reached.

In the example above, the first {## insert} macro uses the element syntax (described in [Insert Element: {## INSERT}](#)) to insert the title of the document. The second macro inserts the author of the document and the third macro inserts the entire body of the document. The resulting HTML might look like this (HTML that is the result of a macro is in **bold**):

```
<html>  
<body>  
<p>Here is the document you requested.  
A Poem by  
Phil Boutros</p>  
  
<p>Below is the document itself</p>  
<p>Roses are red</p>  
<p>Violets are blue</p>  
<p>I'm a programmer</p>  
<p>and so are you</p>  
  
</body>  
</html>
```

7.2 Elements

This section covers the following topics:

- ["Element Tree"](#) on page 7-2
- ["Leaf Elements"](#) on page 7-4
- ["Repeatable Elements"](#) on page 7-5
- ["Element Definitions"](#) on page 7-5

7.2.1 Element Tree

Dynamic Converter uses the concept of an element tree to make various pieces and attributes of the source file individually addressable from within a script template.

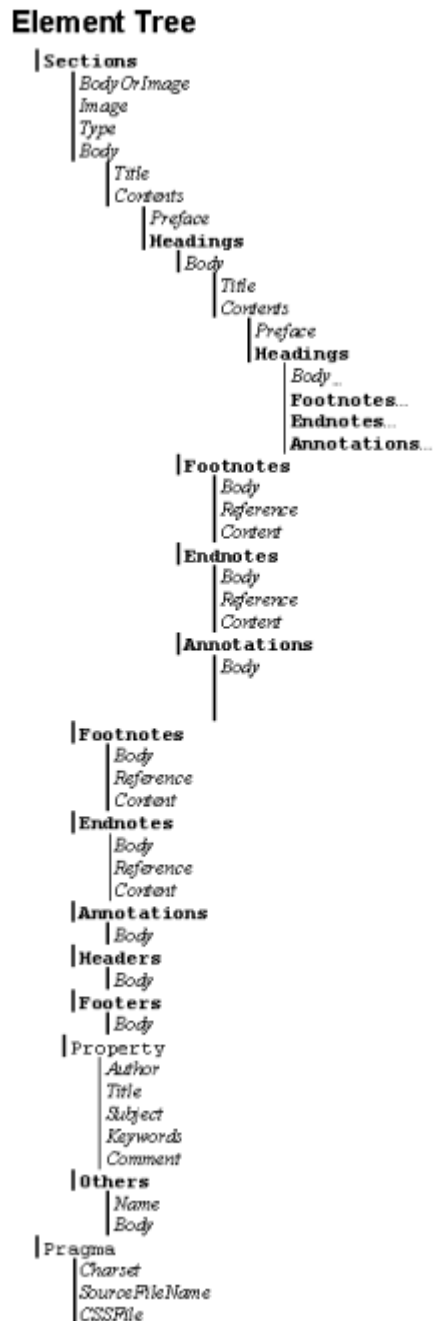
The nodes of the element tree are used to generate a path to a specific element, and a period is used to separate the nodes in this path. For example, the path of the author property of a document is Property.Author.

For convenience, certain nodes in an element path may be skipped because they represent the obvious default behavior. These nodes include the Sections node (Sections.Current.Body.Title is equivalent to Body.Title), and the Body and Contents nodes (Body.Contents.Headings.1.Body is equivalent to Headings.1.Body).

Important: These nodes may not be skipped if they are the last node in the path (Heading.1.Body is *not* equivalent to Headings.1).

There are two types of elements in the element tree: leaf elements and repeatable elements (see "[Leaf Elements](#)" on page 7-4 and "[Repeatable Elements](#)" on page 7-5, respectively).

Figure 7-1 Example of an Element Tree



7.2.2 Leaf Elements

Leaf elements are single identifiable pieces of the source file like the author property (Property.Author) or the preface of the document (Body.Contents.Preface). This type of element is a valid target for inserting, testing and linking using the {## INSERT}, {## F} and {## LINK...} macros. The last node in this type of path must be a valid leaf node in the document tree. Valid leaf nodes are shown in *italics* in the element tree example in [Element Tree](#).

7.2.3 Repeatable Elements

Repeatable elements have multiple instances associated with them, like the footnotes in a document (Sections.1.Footnotes). This type of element may not be directly inserted, tested or linked to but its instances may be looped through using the `{## REPEAT}` macro. The last node in this type of path must be a valid repeatable node in the document tree. Valid repeatable nodes are shown in **bold** in the element tree example in [Element Tree](#).

Some templates use `{## REPEAT}` loops to generate one output file per repeatable element. For example, a template may render a presentation file as a group of output files, with one output file for each slide. When an input file contains an exceptionally large number of sections, it is possible for an operating system to run out of file handles. See your operating system's documentation or system administrator to find out how many open file handles are allowed. To avoid this extremely rare problem, set a value for the `maxreps` attribute of the `{## REPEAT}` macro or configure the operating system to allow more file handles.

7.2.4 Element Definitions

The following table contains a list of all supported elements and a brief description of each. (See "[Indexes](#)" on page 7-9 for a description of valid values for *x*.)

Element	Type	Description
Property.Author	Leaf	Author property of the source file.
Property.Title	Leaf	Title property of the source file.
Property.Subject	Leaf	Subject property of the source file
Property.Keywords	Leaf	Keywords property of the source file.
Property.Comments	Leaf	Comments property of the source file.
Property.Others	Repeatable	This permits access to all properties not specifically accessible through property elements described above, and includes both the "Name" and the "Body" of the property. Which "Other" properties are supported is file format dependant. Some file formats also allow for additional user definable properties. Only text properties are accessible. Properties such as Yes/No, numeric values, and dates are not supported.
Property.Others.x.Name	Leaf	Descriptive name for the property.
Property.Others.x.Body	Leaf	Text of the property.
Sheets	Repeatable	See 'Sections' below.
Slides	Repeatable	See 'Sections' below.
Sections	Repeatable	Sections are used to represent the highest level of abstraction within the source file. In general, word processor documents will have only one section, the document itself. Spreadsheets have one section for each sheet or chart. Presentations have one section for each slide. Graphics generally have one section but may have more, as in a multi-page TIFF. For convenience and readability, Sheets and Slides are synonymous with Sections.

Element	Type	Description
Sections.x.Body	Leaf	<p>This element represents the main textual area of the source file.</p> <p>For word processing documents, it includes the entire document excluding footnotes, endnotes, headers, footers, and annotations. (Footnote/endnote references are always included automatically in the body. If the template includes footnotes/endnotes, then these references provide a link to the note. Annotation references are not placed in the body unless the template includes annotations, in which case they provide links to the annotations.)</p> <p>For spreadsheets, it includes the entire sheet.</p> <p>For graphics, it includes any text that actually appears as text in the file format.</p>
Sections.x.Body.Title	Leaf	<p>For word processing documents, this element is the text marked with the title style. This may be different than the Property.Title. For all other types, this element will be the "name" of the section. For example, if the source file is a spreadsheet, this element will be the name of the sheet as it appears on the spreadsheet application's navigation tabs.</p>
Sections.x.Body.Contents	Leaf	<p>For word processing documents, this is the same as Sections.x.Body.</p> <p>For all other document types, this is the same as the body minus the title, if a title exists.</p>
Sections.x.Body.Contents.Preface	Leaf	Text between the top of the body and the first heading.
Sections.x.Body.Contents.Headings	Repeatable	Headings are labels in a word processor document inserted by the author to give a document structure. See "Breaking Documents by Structure" on page 7-43 for more information on headings. Dynamic Converter reads this structure and, through the use of the Headings element, allows you to access it.
Sections.x.Body.Contents.Headings.x.Body	Leaf with Leaves and Repeatables below	Under each heading, the structure of a complete document from Body down is repeated. See "Breaking Documents by Structure" on page 7-43 for a clearer picture of how these elements map to parts of a document.
Sections.x.Body.Contents.Headings.x.Footnotes	Repeatable with Leaves below	Only footnotes contained in this heading.
Sections.x.Body.Contents.Headings.x.Endnotes	Repeatable with Leaves below	Only endnotes contained in this heading.
Sections.x.Body.Contents.Headings.x.Annotations	Repeatable with Leaves below	Only annotations contained in this heading.
Sections.x.Grids	Repeatable	Only valid for spreadsheet and database formats. This permits access to the "grids" inside a section or sheet of a spreadsheet or database file.
Sections.x.Grids.x.Body	Repeatable	Only valid for spreadsheet and database formats. This permits access to the "grids" inside a section or sheet of a spreadsheet or database file.
Sections.x.Image	Leaf	This element represents a graphic image of the content of the section. It is valid only for bitmap, drawing, chart and presentation sections.

Element	Type	Description
Sections.x.BodyOrImage	Leaf	This element exists to make it easy to build templates that handle a range of section types. In word processing documents, spreadsheets and database sections, BodyOrImage is synonymous with Body. In bitmap, drawing, chart and presentation sections, BodyOrImage is synonymous with Image.
Sections.x.Title	Leaf	Same as Sections.x.Body.Title. For word processing documents, this element is the text marked with the title style. This may be different than the Property.Title. For all other types, this element will be the "name" of the section. For example, if the source file is a spreadsheet, this element will be the name of the sheet as it appears on the spreadsheet application's navigation tabs.
Sections.x.Type	Leaf	This element exists only for query purposes. It is valid only at the ELEMENT of a <code>{## IF...}</code> macro. This element is normally used only for query purposes, but it may be inserted as well. See " Conditional: {## IF...}, {## ELSEIF...}, and {## ELSE} " on page 7-18 for further details on how to use this in an <code>{## IF}</code> macro.
Sections.x.Footnotes	Repeatable	All footnotes.
Sections.x.Footnotes.x.Body	Leaf	The complete footnote reference and content text.
Sections.x.Footnotes.x.Reference	Leaf	The reference number for the footnote.
Sections.x.Footnotes.x.Content	Leaf	The content text for the footnote.
Sections.x.Footnotes	Repeatable	All footnotes.
Sections.x.Endnotes.x.Body	Repeatable with Leaves below	The complete endnote reference and content text.
Sections.x.Endnotes.x.Reference	Repeatable with Leaves below	The reference number for the endnote.
Sections.x.Endnotes.x.Content	Repeatable with Leaves below	The content text for the endnote.
Sections.x.Annotations	Repeatable	All annotations.
Sections.x.Annotations.x.Body	Leaf	The complete annotation reference and content text.
Sections.x.Annotations.x.Reference	Leaf	The reference text for the annotation.
Sections.x.Annotations.x.Content	Leaf	The content text for the annotation.
Sections.x.Slidenotes	Repeatable	All slide notes. Please note that converting the slide notes will slow down the conversion process for PowerPoint files.
Sections.x.Slidenotes.x.Body	Leaf	The notes for the current slide. It is recommended that you write slide notes at the end of the output file for performance reasons (PowerPoint files keep slide notes at the end of the file, not next to each slide). Not doing so will slow conversion, as the technology will be forced to perform excessive seeking in the input file.
Sections.x.Headers	Repeatable	All headers.

Element	Type	Description
Sections.x.Headers.x.Body	Leaf	Text of the header.
Sections.x.Footer	Repeatable	All footers.
Sections.x.Footer.x.Body	Leaf	Text of the footer.
Pragma.Charset	Leaf	<p>The HTML text string associated with the character set of the characters that Dynamic Converter is generating. In order for Dynamic Converter to correctly code the character set into the HTML it generates, all templates should include a META tag that uses the <code>{## INSERT}</code> macro as follows.</p> <pre><META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset={## INSERT ELEMENT=pragma.charset}"></pre> <p>If the template does not include this line, the user will have to manually select the correct character set in their browser.</p>
Pragma.SourceFileName	Leaf	The name of the source document being converted. Note that this does NOT include the path name.
Pragma.CSSFile	Leaf	<p>This element is used to insert the name of the Cascading Style Sheet (CSS) file into HTML documents. This name is typically used in conjunction with an HTML <code><LINK></code> tag to reference styles contained in the CSS file generated by Dynamic Converter.</p> <p>When used with the <code>{## INSERT}</code> macro, this pragma will generate the URL of the CSS file that is created. This macro must be used with <code>{## INSERT}</code> inside every template file that inserts contents of the source file and when the selected HTML flavor supports CSS. The CSS file will only be created if the selected HTML flavor supports CSS.</p> <p>When used with the <code>{## IF}</code> macro, the conditional will be true if the selected HTML flavor supports Cascading Style Sheets or not.</p> <p>If CSS is required for the output, <code>{## IF element=pragma.embeddedcss}</code> or <code>{## IF element=pragma.cssfile}</code> must be used. However, Dynamic Converter does not differentiate between the two, as the choice of using embedded CSS vs. external CSS is your decision and you may even wish to mix the two in the output.</p> <p>An example of how to use this pragma that works when exporting either CSS or non-CSS flavors of HTML would be as follows:</p> <pre>{## IF ELEMENT=Pragma.CSSFile} <LINK REL=STYLESHEET HREF="{## INSERT ELEMENT=Pragma.CSSFile}"> </LINK> {## /IF}</pre>

Element	Type	Description
Pragma.EmbeddedCSS	Leaf	<p>This element is used to insert CSS style definitions in a single block in the <HEAD> of the document.</p> <p>When used with the <code>{## INSERT}</code> macro, this pragma will insert the block of CSS style definitions needed for use later in the file. This macro must be used inside every output HTML file where <code>{## INSERT}</code> is used to insert document content.</p> <p>When used with the <code>{## IF}</code> macro, the conditional will be true if the selected HTML flavor supports CSS.</p> <p>If CSS is required for the output, <code>{## IF element=pragma.embeddedcss}</code> or <code>{## IF element=pragma.cssfile}</code> must be used. However, Dynamic Converter does not differentiate between the two, as the choice of using embedded CSS vs. external CSS is your decision and you may even wish to mix the two in the output.</p> <p>If a style is used anywhere in the input document, that style will show up in the embedded CSS generated for all the output HTML files generated for the input file. Consider a template that splits its output into multiple HTML files. In this example, the input file contains the "MyStyle" style. It does not matter if during the conversion only one output HTML file actually references the "MyStyle" style. The "MyStyle" style definition will still show up in the embedded CSS for all the output files, including those files that never reference this style.</p>
Pragma.JsFile	Leaf	<p>This element is used to insert the name of the JavaScript file into HTML documents. This name is typically used in conjunction with an HTML <SCRIPT> tag to reference JavaScript contained in the .js file generated by HTML Export.</p> <p>When used with the <code>{## INSERT}</code> macro, this pragma will generate the URL of the JavaScript file that is created. This macro must be used with <code>{## NSERT}</code> inside every template file that inserts contents of the source file when:</p> <ul style="list-style-type: none"> ■ The selected HTML flavor supports JavaScript. ■ The <code>javascriptTabs</code> option has been set to true. <p>The JavaScript file will only be created if the selected HTML flavor supports JavaScript.</p> <p>When used with the <code>{## IF}</code> macro, the conditional will depend upon whether the selected HTML flavor supports JavaScript or not.</p>

7.3 Indexes

Repeatable nodes have an associated index variable that has a current value at any given time in the export process. For elements that contain repeatable nodes as part of their paths, the instance of the repeatable element must be specified by using a number or one of the index variable keywords.

This section covers the following topics:

- ["Index Variable Keywords"](#) on page 7-9
- ["Creating a Set of HTML Files for Each Slide in a Presentation"](#) on page 7-12

7.3.1 Index Variable Keywords

The possible values for this index (referred to as 'x' in element definitions (see ["Element Definitions"](#) on page 7-5) are as follows:

- [Whole Number](#)
- [Current, Next, Previous, First, and Last](#)
- [Up, Down, Left, and Right](#)

7.3.1.1 Whole Number

For numeric values, the number is simply inserted as another node in the path.

Note: Dynamic Converter indexes begin counting with 1 (not 0).

For example, `Slides.1.Image` references the first slide in a presentation and `Footnotes.2.Body` references the second footnote in a document.

If it cannot be guaranteed that elements are within the document which the template is applied on, they should not be explicitly referenced. For example, referencing `Sections.4.Body` may result in unexpected behavior in documents that have fewer than four sections.

Requesting a non-existent element will not cause an error in Dynamic Converter. The insertion will just be ignored. However, if other HTML surrounding the insertion depends on the results of the insert, the output may be invalid HTML.

7.3.1.2 Current, Next, Previous, First, and Last

The 'current', 'next', 'previous', 'first', and 'last' keywords are fairly self-explanatory. When the script template is processed, these variables are replaced with the appropriate index value. For example, `Slides.Current.Image` references the current slide and `Slides.Next.Image` refers to the next slide.

'Next' and 'previous' do not change the value of the index, as was the case in earlier versions of Dynamic Converter. As a result, the only places where the index is changed are inside of a `{## REPEAT}` loop and as the result of a `{## LINK}` statement.

{## REPEAT...}

The initial value of the index variable for any given repeatable element typically is 1. For `{## REPEAT}` loops, the index is incremented with each iteration. Termination of a `{## REPEAT}` loop resets the counter to its initial value. Actually, it is more accurate to say that the scope of the index is the repeat loop.

The following template fragment uses `current` in a repeat loop, which outputs all the footnotes in the source file:

```
{## REPEAT element=footnotes}
{## INSERT element=footnotes.current.body}
{## /REPEAT}
```

When a template containing a repeat statement is the target of a `{## link}` statement that specifies the element to be used as the repeat element, the initial value of the index will be determined by the `{## LINK}` processing.

{## LINK...}

The `{## LINK}` statement does not affect the index variable in the context of the current template. The `{## LINK}` statement can only affect index variables when both an element and a template are specified. In this case only the index variables in the target for the specified element are affected.

If the element specified in the `{## LINK}` contains a next or previous keyword, the value of current in the target file will be affected. The initial value of current in the target will be the value of (current in the source)+1 for next. Similarly, previous has the effect of decrementing the value of current.

The following example uses a single template file and the `{## link}` macro to create a set of HTML files, one for each slide in a presentation. The `{## link}` does the dual job of driving the generation of the HTML files and providing a "next" link for navigation. Notice the use of the `next` keyword in the `{## if}` macro that checks to see if there is a next slide:

```
{## unit}
<html>
<body>
<!-- insert the current slide -->
{## insert element=slides.current.image width=300}
<hr />
<!-- Is there a next slide? -->
{## if element=slides.next.image}
  <!-- If yes, generate a URL to an HTML file containing
  the next slide. The HTML file is generated using
  the current template (because there is no template
  attribute). While generating the new HTML file, the
  value of the index on slides will be its current
  value plus 1 once control returns to this template,
  the value of the index on slides is unchanged. -->
  <p><a href="{## link element=
  slides.next.image}">Next</a></p>
{## else}
  <!-- If no, create a link to the HTML containing the
  first slide. -->
  <p><a href="{## link element=
  slides.1.image}">First</a></p>
{## /if}
</body>
</html>
{## /unit}
```

7.3.1.3 Up, Down, Left, and Right

In addition to the [Current](#), [Next](#), [Previous](#), [First](#), and [Last](#) index variable keywords, repeatable grid elements have four additional keywords:

- Up
- Down
- Left
- Right

These keywords may only appear immediately after the Grids node in the document tree. For example, `Grids.Up.Body` is legal, but `Sections.Left.Grids.1.Body` is not. Use of these keywords is otherwise self-explanatory.

Note, too, that individual grids are only addressable relative to each other. In other words, while it is possible to specify the "up" grid, it is not possible to arbitrarily specify a grid directly (i.e., "5, 7").

7.3.2 Creating a Set of HTML Files for Each Slide in a Presentation

The following example uses a single script template file and the `{## LINK...}` macro to create a set of HTML files, one for each slide in a presentation. The `{## LINK...}` does the dual job of driving the generation of the HTML files and providing a "next" link for navigation. Notice the use of the `Next` keyword in the `{## IF...}` macro that checks to see if there is a next slide.

```
<html>
<body>
<!-- Insert the current slide -->
{## INSERT ELEMENT=Slides.Current.Image WIDTH=300}
<hr />
<!-- Is there a next slide? -->
{## IF ELEMENT=Slides.Next.Image}
<!-- If yes, generate a URL to an HTML file containing the next slide. The HTML
file is generated using the current template (because there is no TEMPLATE
attribute). While generating the new HTML file, the value of the index on Slides
is its current value plus 1 once control returns to this template, the value of
the index on Slides is unchanged. -->
<p><a href="{## LINK ELEMENT=Slides.Next.Image}">Next</a></p>
{## ELSE}
<!-- If no, create a link to the HTML containing the first slide. -->
<p><a href="{## LINK ELEMENT=Slides.1.Image}">First</a></p>
{## /IF}
</body>
</html>
```

7.4 Macros

This section covers the following topics:

- ["About Macros"](#) on page 7-12
- ["Units: {## UNIT}, {## HEADER}, and {## FOOTER}"](#) on page 7-13
- ["Insert Element: {## INSERT}"](#) on page 7-14
- ["Conditional: {## IF...}, {## ELSEIF...}, and {## ELSE}"](#) on page 7-18
- ["Loop: {## REPEAT}"](#) on page 7-19
- ["Linking With Structured Breaking: {## LINK}"](#) on page 7-20
- ["Linking With Content Size Breaking: {## ANCHOR}"](#) on page 7-23
- ["Comment Put in the Output File: {## IGNORE}"](#) on page 7-24
- ["Comment Not Put in the Output File: {## COMMENT}"](#) on page 7-24
- ["Including Other Templates: {## INCLUDE}"](#) on page 7-25
- ["Setting Options Within the Template: {## OPTION}"](#) on page 7-25
- ["Copying Files: {## COPY}"](#) on page 7-28
- ["Deprecated Template Macros"](#) on page 7-29

7.4.1 About Macros

Macros are commands to Dynamic Converter within script templates. Despite their casual similarity to HTML tags, they are not bound by any of the rules that tags would

usually follow inside an HTML file. Macros may appear anywhere in the script template file, except inside another macro.

In the documentation and examples, the pieces of a macro are always shown delimited by spaces. However, semicolons may also delimit them. This option was added to accommodate certain HTML editors. In certain editors, URLs entered into dialog boxes may not have non-quoted spaces. This made it difficult or impossible to use the `{## LINK}` macro in these situations.

For example, `{## INSERT ELEMENT=Sections.1.Body}` may also be written as `{##;INSERT;ELEMENT=Sections.1.Body}`.

Note that template macro string parameters and options support printf style escaped characters. This means that characters such as `\x22`, `\r` and `%%` are supported. Also note that most template attribute values may be quoted. The exception is template element strings, which may not be quoted at this time.

For example:

```
{## ANCHOR aref="next" format="<a href=\"%url\">Next</a><br/>\r\n"}
```

7.4.2 Units: `{## UNIT}`, `{## HEADER}`, and `{## FOOTER}`

If a template file is going to make use of the `{## UNIT}` macro at all, this macro must be the first macro in the template file. It delimits the beginning and end of each unit. Unit boundaries are used when determining where to break the document when breaking based on content size (see ["Breaking Documents by Content Size"](#) on page 7-46).

A unit consists of a header, a footer (both of which are optional), and a body (which may be empty). To ensure that the header is the first item in the template and the footer is the last item, text between the `{## UNIT}` tag and the `{## HEADER}` tag will be ignored, as will text between the `{## /FOOTER}` tag and the `{## /UNIT}` tag, including whitespace. The header and footer of a unit will be output in every page containing that unit, enclosing that portion of the unit's body that is able to fit in a particular page. The entire template is a unit that may contain additional units.

Syntax

```
{## UNIT [BREAK]}
  [{## HEADER}
   any HTML
   {## /HEADER}]

   any HTML

  [{## FOOTER}
   any HTML
   {## /FOOTER}]
{## /UNIT}
Attributes
BREAK
```

Attribute	Description
BREAK	<p>This optional attribute forces a page break before inserting the unit contents unless doing so would cause the body of the first page to be empty. One situation where this attribute would be useful would be to force a page break between each section of a document, perhaps to get one presentation slide per page.</p> <p>The <code>{## UNIT}</code> macro and its <code>BREAK</code> attribute are ignored when <code>SCCOPT_EX_PAGESIZE</code> <code>pagesize</code> is set to zero.</p> <p>It is sometimes important to make sure that a break does not occur in the midst of text that is intended to be on the same page. To prevent breaks like this from occurring, enclose the text that should be kept on the same page inside a nested <code>{## UNIT}{## HEADER}</code> pair. For example, to prevent a page break from occurring while a link is being created, the template author might write something like the following:</p> <pre>{## unit}{## header} Link {## /header}{## /unit}</pre>

7.4.3 Insert Element: `{## INSERT}`

This macro inserts an element of the source file into the output file at the current location.

Syntax

```
{## INSERT [ELEMENT=element [WIDTH=width] [HEIGHT=height] [SUPPRESS=suppress]
[TRUNCATE=truncate]] | [NUMBER=number] [URLENCODE]}
```

Attribute	Description
ELEMENT	<p>This attribute describes which part of the source file should be placed in the output file at the location of the macro. See "Element Definitions" on page 7-5 for the possible values for this attribute. If the value of this attribute is not in the element tree, Dynamic Converter considers it to be a custom element and the <code>EX_CALLBACK_ID_PROCESSELEMENTSTR</code> callback is called.</p> <p>Example: <code>{## INSERT ELEMENT=Sections.1.Body}</code></p>
WIDTH	<p>This optional attribute defines the width in pixels of the element being inserted. It is currently only valid for the Image element. If the <code>WIDTH</code> attribute is not present but the <code>HEIGHT</code> attribute is, the width of the image is calculated automatically based on the shape of the element. If neither the <code>WIDTH</code> and <code>HEIGHT</code> attributes are present, the image's original dimensions are used. If the image's original dimensions are unknown, the defaults assume a <code>HEIGHT</code> and <code>WIDTH</code> of 200.</p> <p>Example: <code>{## INSERT ELEMENT=Slides.1.Image WIDTH=400}</code></p>
HEIGHT	<p>This optional attribute defines the height in pixels of the element being inserted. It is currently only valid for the Image element. If the <code>HEIGHT</code> attribute is not present, but the <code>WIDTH</code> attribute is, the height of the image is calculated automatically based on the shape of the element.</p> <p>Example: <code>{## INSERT ELEMENT=Slides.1.Image HEIGHT=400}</code></p>

Attribute	Description
SUPPRESS	<p>This optional attribute allows certain things to be suppressed from the output. This is very useful if elements need to be inserted in contexts where HTML is not appropriate, such as passing information to Java applets, ActiveX controls, or populating parts of a form. Possible values are as follows:</p> <p>TAGS: All HTML tags are suppressed from the output of the element, however the text may still contain HTML character codes like &quot; or &#123;</p> <p>For non-embedded graphics such as presentations and graphic files, the URL of the converted graphic will not be suppressed. The tag that would normally surround the URL is suppressed, however.</p> <p>For embedded graphics such as those found in word processing sections and spread sheets, both the URL and the tag are suppressed. Since there would be no way to access the resulting converted embedded graphic, conversion of the graphic is not done.</p> <p>Example:</p> <pre data-bbox="662 724 1373 835"><form method="POST"> <input type="text" size="20" name="Author" value="{## INSERT ELEMENT=Property.Author SUPPRESS=TAGS}"> </form></pre> <p>BOOKMARKS: Turns off all bookmarks in the inserted section. Bookmarks automatically precede many inserted elements so that other template elements may link to them. SUPPRESS=BOOKMARKS is provided to prevent problems with nested <a> tags. Note that this represents a subset of the suppression behavior provided by SUPPRESS=TAGS.</p> <p>INVALIDXMLTAGCHARS: Drops from the output all characters that are not allowed in XML tag names. This is designed to allow template authors to {## INSERT} custom document property names inside angle brackets ("<" and ">") to create XML tags. Most characters in Unicode and its subset character sets may be used as part of XML tag names. Illegal tag characters include "control" characters such as line feed and carriage return. In addition there are special rules for what characters can be the first character in a tag name.</p> <p>Example:</p> <pre data-bbox="662 1285 1263 1514">{## REPEAT Sections.Property.Others} <{## INSERT ELEMENT=Property.Others.Current.Name SUPPRESS=InvalidXMLTagChars}> <{## INSERT ELEMENT=Property.Others.Current.Body SUPPRESS=InvalidXMLTagChars}> </{## INSERT ELEMENT=Property.Others.Current.Name SUPPRESS=InvalidXMLTagChars}> {/## REPEAT}</pre> <p>produces something similar to the following:</p> <pre data-bbox="662 1562 1141 1593"><MyProperty>PropertyValue</MyProperty></pre>

Attribute	Description
TRUNCATE	<p data-bbox="581 233 1354 470">When set, this attribute forces a maximum length in characters for the inserted element. This allows elements to be truncated rather than broken across pages when the page size option is in use. Truncated elements will end with the truncation identifier which is "..." (three periods). All elements that have a truncate value will be no more than the specified number of characters in length including the length of the truncation identifier. In Dynamic Converter, elements are inserted in their entirety if no truncation size is specified. The value of this attribute must be greater than or equal to five characters.</p> <p data-bbox="581 485 1328 537">An example of a situation where element truncation is useful is to limit the size of entries when building a table of contents.</p> <p data-bbox="581 552 1360 604">The TRUNCATE attribute implies suppression of tags for the insert. It also auto applies the no source formatting option for the insert.</p> <p data-bbox="581 619 1360 693">Note that the TRUNCATE attribute cannot be used with custom elements, because the custom element definition precludes the existence of any other attributes to <code>{## INSERT}</code>.</p> <p data-bbox="581 707 1338 760">The TRUNCATE attribute has three special aspects to its behavior when grids are being inserted:</p> <p data-bbox="581 774 1354 848">When truncation is in effect, the truncation size refers to the number of characters of content in each cell, not the number of characters in the grid as a whole.</p> <p data-bbox="581 863 1360 936">While truncation normally causes all markup tags to be suppressed, when grids are in use, the table tags are retained (assuming that the output flavor supports tables).</p> <p data-bbox="581 951 1360 1148">Users are reminded that only one grid size may be selected for each spreadsheet sheet or database inserted. The size of the grid will be based in part on the TRUNCATE value if one or both the grid dimensions are not specified and the SCCOPT_EX_PAGESIZE option is in use. In this situation, if a grid from a single sheet is inserted in more than one place in the template, and there are differing TRUNCATE values, then the grid dimensions will be based on the largest TRUNCATE value specified.</p>

Attribute	Description
NUMBER	<p>This attribute allows the developer to retrieve the total instance count or the current index value of any repeatable element. This can be very useful for writing JavaScript, BasicScript, etc. Two special keywords do not appear in the element tree but can be used as nodes in the following special case.</p> <p>Count and CountB0: When appended to a repeating element and used with the NUMBER attribute, these nodes allow the developer to insert a text representation of the number of instances of the given repeatable element. Count gives the count assuming the first index is 1 and CountB0 gives it assuming the first index is 0.</p> <p>Example: If a presentation has three slides, the template fragment,</p> <pre data-bbox="659 562 1097 615"><P>{## INSERT NUMBER=Slides.Count} <P>{## INSERT NUMBER=Slides.CountB0}</pre> <p>produces the following text:</p> <pre data-bbox="659 667 708 720"><P>3 <P>2</pre> <p>Value and ValueB0: When appended to a repeating element and used with the NUMBER attribute, these nodes allow the developer to insert a text representation of the current value of the index of the given repeatable element. Value gives the count assuming the first index is 1 and ValueB0 gives it assuming the first index is 0.</p> <p>Example: If the current value of the index on Slides is 2, the template fragment,</p> <pre data-bbox="659 947 1195 999"><P>{## INSERT NUMBER=Slides.Current.Value} <P>{## INSERT NUMBER=Slides.Current.ValueB0}</pre> <p>Produces the following text:</p> <pre data-bbox="659 1052 708 1104"><P>2 <P>1</pre>
URLENCODE	<p>This optional attribute causes the inserted element to be URL encoded. As such, it is ignored unless it is specified as part of an insert that contains a file name. The following elements may be URL encoded:</p> <ul data-bbox="659 1220 954 1367" style="list-style-type: none"> ■ pragma.sourcefilename ■ pragma.cssfile ■ pragma.embeddedcss ■ pragma.jsfile <p>In addition, the following elements will be URL encoded when the section type is "Archive" or "AR":</p> <ul data-bbox="659 1440 922 1629" style="list-style-type: none"> ■ sections.x.fullname ■ sections.x.basename ■ sections.x.body ■ sections.x.title ■ sections.x.reflink <p>For all other {## INSERT} tags, this attribute is ignored. As such, you should note that Dynamic Converter does not modify any URLs coming out of the input documents being converted. These URLs continue to be passed through as is. This attribute is also ignored if the URL was created using the EX_CALLBACK_ID_CREATENEWFILE callback. Such URLs are assumed to already be URL-encoded.</p>

Inserting Properties

Because of the special ways that properties are used in documents, property strings are inserted into the output HTML a little differently than the way other `{## INSERT}` macros work.

The property is always inserted as if the `SCCOPT_NO_SOURCEFORMATTING` option were set. This prevents formatting characters such as new lines from interfering with the property strings.

The property is always inserted as if the script template specified `Suppress=Tags`. This provides you with maximum control over how property strings are presented.

7.4.4 Conditional: `{## IF...}`, `{## ELSEIF...}`, and `{## ELSE}`

This macro allows an area of the script template to be used based on information about an element of the source file.

Syntax

```
{## IF ELEMENT=element [CONDITION=Exists|NotExists]
[VALUE=value]}
    any HTML
{## /IF}
```

or

```
{## IF ELEMENT=element [[CONDITION=Exists|NotExists] |
[VALUE=value]]}
    any HTML
{## ELSE}
    any HTML
{## /IF}
```

or

```
{## IF ELEMENT=element [[CONDITION=Exists|NotExists] |
[VALUE=value]]}
    any HTML
{## ELSEIF ELEMENT=element [[CONDITION=Exists|NotExists] |
[VALUE=value]]}
    any HTML
{## ELSE}
    any HTML
{## /IF}
```

Note: Multiple `{## ELSEIF}` statements may be used after `{## IF}`. In addition, `{## ELSE}` is not required when using `{## ELSEIF}`.

Attribute	Description
ELEMENT	This attribute describes which part of the source file should be tested. See "Element Definitions" on page 7-5 for the possible values for this attribute. If neither the <code>CONDITION</code> nor <code>VALUE</code> attribute exists, the element is tested for existence.
CONDITION	Defines the condition the element is tested for, possible values are <code>EXISTS</code> and <code>NOTEXISTS</code> .

Attribute	Description
VALUE	<p>Defines the values the element should be tested against. The VALUE attribute is currently valid only for the Sections.x.Type element for testing of the type of a section of the source file.</p> <p>Possible values include:</p> <ul style="list-style-type: none"> ■ ar = archive ■ bm = bitmap ■ ch = chart ■ db = database ■ dr = drawing ■ mm = multimedia ■ pr = presentation ■ ss = spreadsheet ■ wp = word processing document <p>Examples:</p> <pre>{## if element=property.comment} <p>Comment property exists</p> {## else} <p><i>Comment property does not exist</i></p> {## /if} {## if element=sections.1.type value=wp} <p>The source file is a word processor file</p> {## /if} {## if element=sections.1.type value=ss} <p>Spreadsheet</p> {## elseif element=sections.1.type value=ar} <p>Archive</p> {## elseif element=sections.1.type value=ch} <p>Chart</p> {## else} <p>Not ss, ar, or ch</p> {## /if} {## if element=sections.current.type value=pr condition=notexists} <p>We can do something here for all document types other than presentations.</p> {## else} <p>This is used only for presentations.</p> {## /if}</pre>

7.4.5 Loop: {## REPEAT}

This command allows an area of the script template to be repeated, once for each occurrence of an element.

Syntax

```
{## REPEAT ELEMENT=element [MAXREPS=maxreps] [SORT=sort]}
  any HTML
{## /REPEAT}
```

Attribute	Description
ELEMENT	<p>This attribute describes what part of the source file should be repeated on. It must be a repeatable element. See "Element Definitions" on page 7-5 for the possible values for this attribute.</p> <p>Any HTML may be defined between the <code>{## REPEAT... }</code> macro and its closing <code>{## /REPEAT}</code> macro. This HTML is repeated once for each instance for the element specified. In addition, the word <code>Current</code> may be used in any other <code>{##}</code> tag as the element-index of the element being repeated. For instance, the following HTML in the template will produce a list of the footnotes in the document.</p> <p>Example:</p> <pre data-bbox="553 548 1101 800"><HTML> <BODY> <P>Here are the footnotes {## REPEAT ELEMENT=Footnotes} <P>{## INSERT ELEMENT=Footnotes.Current.Body} {## /REP} <P>No more footnotes </BODY> </HTML></pre> <p>Similarly, the following HTML in the template will insert the names of all the items in an archive:</p> <pre data-bbox="553 884 1127 961">{## repeat element=sections} {## insert element=sections.current.fullname} {## /repeat}</pre>
MAXREPS	<p>This attribute limits the total number of loops the repeat statement may make to the value specified. It is useful for preventing exceptionally large documents from producing an unwieldy amount of output.</p>
SORT	<p>This optional attribute defines whether to sort the output or not. This attribute is ignored if the input file is not an archive file of arctype file. All sorts are done based on the character encoding of the values in the input file. The sorts are also case insensitive at this time. Valid values of the sort attribute are:</p> <ul style="list-style-type: none"> ■ fullname: sort by <code>Sections.Current.FullName</code> ■ basename: sort by <code>Sections.Current.BaseName</code> ■ none: no sorting is done. This is the default.

7.4.6 Linking With Structured Breaking: `{## LINK}`

This macro generates a relative URL to a piece of the document produced by Dynamic Converter. Normally this URL would then be encapsulated by the template with HTML anchor tags to create a link. `{## LINK}` is particularly powerful when used within a `{## REPEAT}` loop.

Syntax

```
{## LINK ELEMENT=element [TOP]}
```

or

```
{## LINK TEMPLATE=template}
```

or

```
{## LINK ELEMENT=element TEMPLATE=template [TOP]}
```

Attribute	Description
ELEMENT	<p>Defines the element that is the target for the link. The URL that the <code>{## LINK...}</code> macro generates will point to the first instance of this element in the output file. If this attribute is not present, the resulting URL will link to any output file that was produced with the specified script template. If such a file does not exist, the specified script template is used to generate a file.</p> <p>Remember that each element has one or more index values, some of which may be variables. An example of this type of index variable is the "current" in <code>Sections.Current.Body</code>. Use of <code>{## LINK}</code> affects the value of those index variables, which may cause subtle side effects in the behavior of the linked template file.</p> <p>For a description of how <code>{## LINK}</code> affects the index of inserted elements more information, see "Indexes" on page 7-9.</p>
TEMPLATE	<p>The name of a template file which must exist in the same directory as the original template file. If this attribute is not present, the current template will be used. If an element was specified in the <code>{## LINK}</code>, then the template must contain a <code>{## INSERT}</code> statement using that element.</p> <p>It is important to note that while the template language is normally case-insensitive, the case of the template file names specified here is important. The file name specified for the template is passed as is to the operating system. On operating systems such as UNIX, if the wrong case is given for the template file name, the template file will not be found and an error will be returned.</p>
TOP	<p>This attribute is only meaningful if an element is specified in the <code>{## LINK}</code> command. When this attribute exists, the generated URL will not contain a bookmark, and therefore the resulting link will always jump to the top of the HTML file containing the specified element. This is useful if the top of the script template has navigation or other information that the developer would like the user to see.</p>

7.4.6.1 `{## LINK}` Usage Scenarios

Using the first syntax shown at the beginning of this section, a URL for the element bookmark is inserted in the document. Normally this syntax is used to create intradocument links to aid navigation. An example would be creating a link to the next section of the document.

In the second syntax, a URL is created to an output file generated by the specified template. This template is run on the same source document, but may extract different parts of the document. Normally, in this syntax, the "main" template contains a link to a second HTML file. This second file is generated using the template specified by the `{## LINK}` command and contains other document elements. As an example, the "main" template could produce a file containing the body of the document and a link to the second HTML file, which contains the footnotes and endnotes.

The third and most powerful syntax also produces the URL of a file generated by the specified template. This template is then expected to contain an insertion of the specified element. Normally this syntax is used with repeatable elements. It allows the author to generate multiple output files with sequential pieces of the document. As such it provides a way to break large documents up into smaller, more readable pieces. An example of where this syntax would be used is a template that generates a "table of contents" in one HTML file (perhaps a separate HTML frame). The entries in the table are then links to other HTML files generated by different templates.

Note that a `{## LINK}` statement which specifies a template does not always result in a new file being created. New files are only created if the target of the link does not exist yet. So if for example two `{## LINK}` statements specify the same element and

template, only one HTML file is produced and the same URL will be used by both `{## LINK}` statements.

7.4.6.2 `{## LINK}` Archive File Example

The following template generates a list of links to all the extracted and converted files from the source archive file (represented by `decompressedFile` in the following example):

```
{## repeat element=sections}
  <p><a href="{## link element=sections.current.decompressedFile}">
    {## insert Element=sections.current.fullname}</a></p>
{## /repeat}
```

7.4.6.3 `{## LINK}` Presentation File Example

The following example (`template.htm`) uses the first syntax to generate a set of HTML files, one for each slide in a presentation. Each slide will include links to the previous and next slides and the first slide. Note the use of `{## IF}` macros so the first and last slides do not have Previous and Next links respectively:

```
template.htm
<html>
<body>
  {## insert element=slides.current.image width=300}
<hr />
  {## if element=slides.previous.image}
    <p><a href="{## link element=slides.previous.image}">
previous</a></p>
  {## /if}
  {## if element=slides.next.image}
    <p><a href="{## link element=
  slides.next.image}">Next</a></p>
  {## /if}
</body>
</html>
```

Due to the side effects of `{## LINK}` using the `element` attribute, there can be some confusion over what values "current," "previous" and "next" have when each `{## LINK}` is processed. To better illustrate how this template works, consider running it on a presentation that contains three slides:

First Output File

Since no template is specified in the `{## LINK}` statements, `template.htm` is (re)used as the template for all `{## LINK}` statements. For the first slide, nothing interesting happens until `slides.next` is encountered. Since `slides.current` is 1 in this case, `slides.next` refers to `slides.2` and the `{## LINK}` is performed on `slides.2.image`. This `{## LINK}` fills in the anchor tag with the URL for the output file containing the second slide. Since no file containing `slides.2` exists, `{## LINK}` opens a new file.

Second Output File

For the second slide the template is rerun. `slides.current` now refers to `slides.2`, `slides.previous` refers to `slides.1` and `slides.next` refers to `slides.3`. The `{## INSERT}` statement will insert the second slide.

The `{## IF}` statement referring to `slides.previous` succeeds. Since the file containing `slides.1` already exists, no additional file is created. The anchor tag will be filled in with the URL for the first output file.

The `{## IF}` statement referring to `slides.next` also succeeds and the anchor tag will be filled in with the URL for the output file containing the third slide. Since no file containing `slides.3` exists, `{## LINK}` opens a new file.

Third Output File

For the third slide the template is rerun. `slides.current` now refers to `slides.3` and `slides.previous` refers to `slides.2`. `slides.next` refers to `slides.4`, which does not exist. The `{## INSERT}` statement will insert the third slide.

The `{## IF}` statement referring to `slides.previous` succeeds. Since the file containing `slides.2` already exists, no additional file is created. The anchor tag will be filled in with the URL for the second output file.

The `{## IF}` statement referring to `slides.next` fails. At this point processing is essentially complete.

7.4.7 Linking With Content Size Breaking: `{## ANCHOR}`

This macro generates a relative URL to a piece of the document produced by Dynamic Converter when doing document breaking based on content size.

Syntax

```
{## ANCHOR AREF=type [STEP=stepval] FORMAT="anchorfmt" [ALT/LINK="element"]
[ALT/TEXT="text"]}
```

Attribute	Description
AREF	<p>Indicates the relation of the target of the link to the current file. Allowable values for this attribute are:</p> <ul style="list-style-type: none"> ▪ insertStart: links to first page of the inserted element ▪ InsertEnd: links to last page of the inserted element ▪ Next: links to next page in the inserted element ▪ Prev: links to previous page in the inserted element ▪ FirstFile: links to first page created for the entire document ▪ LastFile: links to last page created for the entire document
STEP	<p>This attribute is used to insert a link to "fast forward/rewind" through the output pages. This attribute may only be used if AREF is "next" or "prev." It is specified as a non-zero positive integer. For example, to insert a link to skip ahead five pages in a document, the following statement could be used:</p> <pre>{## unit aref="next" step="5" format="<p>Next</p>"}</pre> <p>If not specified, the default value of the STEP attribute is one (1), which corresponds to the next/previous page. This attribute has no meaning when aref equals "insertstart," "insertend," "firstfile," or "lastfile."</p>
FORMAT	<p>This is an sprintf style format string specifying the text to output as the link. Dynamic Converter replaces the <code>%url</code> format specifier with the target URL into the format string. For example:</p> <pre>{## anchor aref="next" format="Next
\r\n"}</pre>

Attribute	Description
ALTLINK	<p>An attribute used to specify the target of the anchor if it cannot be resolved based on the anchor type. For example, the final file of a breakable element has no "next" file, and thus would resolve to nothing. However, if the altlink attribute is specified, the anchor will be generated using a URL to the first file found containing the specified element.</p> <p>Note that no EX_CALLBACK_ID_ALTLINK callback will be made if an EX_CALLBACK_ID_ALTLINK attribute is specified in the {## ANCHOR} statement.</p> <p>For example:</p> <pre>{## anchor aref=next format="Next" altlink=headings.next.body}</pre>
ALTTEXT	<p>Text to be output if the anchor cannot be resolved. If this attribute is not specified, no text will be output if the anchor target does not exist. For example:</p> <pre>{## anchor aref=next format="Next" alttext="Next"}</pre>

7.4.8 Comment Put in the Output File: {## IGNORE}

This macro causes {##} statements in an area of the template file to be ignored by the template parser. Any text between the {## IGNORE} and {## /IGNORE} tags will be written to the output file as-is. This macro allows {##} statements in an area of the template to be commented out for debugging purposes, or to actually write out the text of another {##} macro. However, the browser will parse any HTML tags inside the ignored block and the text will be formatted accordingly. This macro can ignore all {##} macros except for an {## /IGNORE} macro. No escape sequence has been implemented for this purpose. As a result, {## IGNORE} statements cannot be nested. If they are nested, a run time template parser error will occur.

Syntax

```
{## IGNORE}
    any HTML or other {##} macros
{## /IGNORE}
```

Note: To fully comment out a section of the script template, surround the ## IGNORE statements with HTML comments, for example:

```
<!--{## Ignore} (everything between here and the end HTML comment
is commented out) {## /Ignore}-->
```

7.4.9 Comment Not Put in the Output File: {## COMMENT}

The {## COMMENT} macro allows the template writer to include comments in the template without including them in the final output files. {## COMMENT} provides the functionality of {## ignore}, but the text inside the {## COMMENT} block is not rendered to the output files and is not included in page size calculations. Like {## IGNORE}, {## COMMENT} macros may not be nested.

Syntax

```
{## COMMENT}
    any HTML or other {##} macros
```

```
{## /COMMENT}
```

7.4.10 Including Other Templates: **{## INCLUDE}**

This command allows other templates to be inserted into the current template. It works in a manner similar to the C/C++ `# include` directive.

Syntax

```
{## INCLUDE TEMPLATE=template}
```

Attribute	Description
TEMPLATE	This attribute gives the name of the template to insert.

7.4.11 Setting Options Within the Template: **{## OPTION}**

This macro sets an option to a given value. All `{## OPTION}` statements are executed in the order in which they are encountered. Remember when using this template macro that the `{## UNIT}` tag must be the first template macro in any template.

Options set in the template have template scope. This means that, for example, if a `{## LINK}` macro references another template, options in the referenced template are not affected by the option settings from the parent template. Similarly, when the files contained in an archive file are converted, Export recursively calls itself to perform the exports of the child documents in the archive. Each child document is converted using a copy of the parent template, and that copy does not inherit the option values from the parent template.

Options set using `{## OPTION}` in the template are not inherited by the dynamic conversions performed on files within archives. Each child conversion receives a fresh copy of all option values as originally set with `DASetOption`.

Remember that setting an option in the template overrides any option value set by an application within the scope of the template.

Syntax

```
{## OPTION OPTION=value}
```

Attribute	Description
OPTION	See the table below for the supported options and their values.

Supported Options and Values

Option	Description
SCCOPT_GRAPHIC_TYPE	<p>This option sets the format of the graphics produced by Dynamic Converter when it converts document embeddings.</p> <p>The supported values are:</p> <ul style="list-style-type: none"> ■ FI_GIF: GIF graphics ■ FI_JPEGFIF: JPEG graphics ■ FI_PNG: PNG graphics ■ FI_NONE: no graphic conversion <p>The default is FI_JPEGFIF.</p>
SCCOPT_GIF_INTERLACED	<p>This option specifies whether GIF output should be interlaced or non-interlaced. Interlaced GIFs are useful when graphics are to be downloaded over slow Internet connections. They allow the browser to begin to render a low-resolution view of the graphic quickly and then increase the quality of the image as it is received. There is no real penalty for using interlaced graphics.</p> <p>The supported values are:</p> <ul style="list-style-type: none"> ■ 0 or FALSE (i.e., non-interlaced) ■ 1 or TRUE (i.e., interlaced)
SCCOPT_JPEG_QUALITY	<p>This options sets the lossyness of JPEG compression. This should be a value between 1 and 100 (percent), with 100 being the highest quality but the least compression, and 1 being the lowest quality but the most compression.</p>
SCCOPT_GRAPHIC_SIZEMETHOD	<p>This option determines the method used to size graphics. You can choose among three methods, each of which involves some degree of trade off between the quality of the resulting image and speed of conversion:</p> <ul style="list-style-type: none"> ■ SCCGRAPHIC_QUICKSIZING ■ SCCGRAPHIC_SMOOTHSIZING ■ SCCGRAPHIC_SMOOTHGRAYSCALESIZING <p>Using the quick sizing option results in the fastest conversion of color graphics, though the quality of the converted graphic will be somewhat degraded.</p> <p>The smooth sizing option results in a more accurate representation of the original graphic, as it uses antialiasing. Antialiased images may appear smoother and can be easier to read, but rendering when this option is set will require additional processing time.</p> <p>Please note that the smooth sizing option does not work on images which have a width or height of more than 4,096 pixels.</p> <p>The grayscale-only option also uses antialiasing, but only for grayscale graphics, and the quick sizing option for any color graphics.</p>

Option	Description
SCCOPT_GRAPHIC_OUTPUTDPI	<p>This option specifies the output graphics device's resolution in dots per inch (dpi), and only applies to images whose size is specified in physical units (in/cm). For example, consider a 1-inch square, 100-dpi graphic that is to be rendered on a 50-dpi device (with this option set to '50'). In this case, the size of the resulting WBMP, TIFF, BMP, JPEG, GIF, or PNG will be 50 x 50 pixels.</p> <p>The valid values are any integer between 0 and 2400 (dpi).</p>
SCCOPT_GRAPHIC_SIZELIMIT	<p>This option sets the maximum size of the exported graphic (in pixels). It may be used to prevent inordinately large graphics from being converted to equally cumbersome output files, thus preventing bandwidth waste.</p> <p>This option takes precedence over all other options and settings that affect the size of a converted graphic.</p>
SCCOPT_GRAPHIC_WIDTHLIMIT	<p>This option allows a hard limit to be set for how wide (in pixels) a converted graphic may be. Any images wider than this limit will be resized to match the limit. It should be noted that regardless whether the SCCOPT_GRAPHIC_HEIGHTLIMIT option is set or not, any resized images will preserve their original aspect ratio. Images smaller than this width are not enlarged when using this option.</p>
SCCOPT_GRAPHIC_HEIGHTLIMIT	<p>This option allows a hard limit to be set for how high (in pixels) a converted graphic may be. Any images higher than this limit will be resized to match the limit. It should be noted that regardless whether the SCCOPT_GRAPHIC_WIDTHLIMIT option is set or not, any resized images will preserve their original aspect ratio. Images smaller than this height are not enlarged when using this option.</p>
SCCOPT_EX_FONTFLAGS	<p>This option is used to turn off specified font-related markup in the output. Naturally, if the requested output flavor or other option settings prevent markup of the specified type from being written, this option cannot be used to turn it back on. However, specifying the size, color and font face of characters may all be suppressed by bitwise OR-ing together the appropriate combination of flags in this option.</p> <ul style="list-style-type: none"> ■ SUPPRESS_SIZE ■ SUPPRESS_COLOR ■ SUPPRESS_SIZECOLOR ■ SUPPRESS_FACE ■ SUPPRESS_SIZEFACE ■ SUPPRESS_COLORFACE ■ SUPPRESS_ALL ■ SUPPRESS_NONE
SCCOPT_EX_GRIDROWS	<p>This option specifies the number of rows that each template "grid" (applicable only to spreadsheet or database files) should contain.</p> <p>Setting this option to zero ("0") means that no limit is placed on the number of rows in the grid.</p>

Option	Description									
SCCOPT_EX_GRIDCOLS	<p>This option specifies the number of columns that each template "grid" (applicable only to spreadsheet or database files) should contain.</p> <p>Setting this option to zero ("0") means that no limit is placed on the number of columns in the grid.</p>									
SCCOPT_EX_GRIDADVANCE	<p>This option specifies how the "previous" and "next" relationships will work between grids.</p> <ul style="list-style-type: none"> ■ ACROSS: The input spreadsheet or database is traversed by rows. ■ DOWN: The input spreadsheet or database is traversed by columns. <p>This option has no effect on up/down or left/right navigation.</p>									
SCCOPT_EX_GRIDWRAP	<p>This option specifies how the "previous" and "next" relationships work between grids at the edges of the spreadsheet or database.</p> <p>Consider a spreadsheet that has been broken into 9 grids by HTML Export as follows:</p> <table border="1" data-bbox="797 835 1058 968"> <tr> <td>Grid 1</td> <td>Grid 2</td> <td>Grid 3</td> </tr> <tr> <td>Grid 4</td> <td>Grid 5</td> <td>Grid 6</td> </tr> <tr> <td>Grid 7</td> <td>Grid 8</td> <td>Grid 9</td> </tr> </table> <p>If this option is set to TRUE, then the Grids.Next.Body value after Grid 3 will be Grid 4. Likewise, the Grids.Previous.Body value before Grid 4 will be Grid 3.</p> <p>If this option is set to FALSE, then the Grids.Next.Body after Grid 3 will not exist as far as template navigation is concerned. Likewise, the Grids.Previous.Body before Grid 4 will not exist as far as template navigation is concerned.</p> <p>In other words, this option specifies whether the "previous" and "next" relationships "wrap" at the edges of the spreadsheet or database.</p>	Grid 1	Grid 2	Grid 3	Grid 4	Grid 5	Grid 6	Grid 7	Grid 8	Grid 9
Grid 1	Grid 2	Grid 3								
Grid 4	Grid 5	Grid 6								
Grid 7	Grid 8	Grid 9								

7.4.12 Copying Files: {## COPY}

The {## COPY} macro is used to copy extra, static files into the output directory along with the output from the converted document. For example, if you have added a company logo that was not in the original input document, {## COPY} can be used to make it a part of the converted output document. Other examples include graphics used to mimic "buttons" for navigation, outside CSS files, or a piece of Java code to be run.

Syntax

```
{## COPY FILE=file}
```

Attribute	Description
FILE	This is the name of the file to be copied. If a relative path name is specified as part of the file, then it must be relative to the directory containing the root template file. For example: <code>{## COPY FILE=uparrow.gif}</code>

The `{## COPY}` macro may occur anywhere inside a template. If the `{## COPY}` is inside a `{## IF}`, then the `{## COPY}` will only be executed if the condition is TRUE. In `{## REPEAT}` loops, the `{## COPY}` will only be performed if the loop is executed one or more times. In addition, if the `{## REPEAT}` loops more than once, Dynamic Converter detects this and the `{## COPY}` is executed only once.

As its name suggests, the `{## COPY}` macro is a straight file copy. Therefore, no conversions are performed as part of the copy. For example, graphics formats are not changed and graphics are not resized. Template authors should also remember to use `{## GRAPHIC}` when graphics and other files are copied so that space will be created for the external graphic in the text buffer size calculations.

Since the only action Dynamic Converter takes is to copy the requested file, it is up to the template author to make use of the copied file at another point in the template. For example, a graphic file may be copied and then the template can use an `` tag which references the copied graphic. The following snippet of template code would do this:

```
{## copy FILE=Picture.JPG
{## graphic PATH=Picture.JPG}

```

Note: If the file copy fails, Dynamic Converter will continue and no error will be reported.

7.4.13 Deprecated Template Macros

Earlier releases of Dynamic Converter used different macro syntax where template macros were expected to start with `{Inso}` rather than `{##}`. In addition some words that had been abbreviated must now be spelled out ("insert" instead of "ins"). The old syntax will continue to be supported for the foreseeable future. However, it has been deprecated.

The old Inso macros and their new equivalents are as follows:

- `{insoins}` is now `{## insert}`
- `{insoif} ... {/insoif}` is now `{## if} ... {## /if}`
- `{insoelseif} ... {/insoelseif}` is now `{## elseif} ... {## /elseif}`
- `{insoelse} ... {/insoelse}` is now `{## else} ... {## /else}`
- `{insoignore} ... {/insoignore}` is now `{## ignore} ... {## /ignore}`
- `{insolink}` is now `{## link}`
- `{insorep} ... {/insorep}` is now `{## repeat} ... {## /repeat}`

You cannot mix old-style Inso macros with the new `{##}` macro style in the same template.

No new or future features that Dynamic Converter will include support the old syntax. Thus, for example, the old syntax has not been extended to include support for the new `{## UNIT}` macros.

7.5 Pragmas

Pragmas provide access to certain document elements that are not logically part of the element tree. The following pragmas are supported:

- [Pragma.Charset](#)
- [Pragma.CSSFile](#)
- [Pragma.EmbeddedCSS](#)
- [Pragma.JsFile](#)
- [Pragma.SourceFileName](#)

7.5.1 Pragma.Charset

This pragma represents the HTML text string associated with the character set of the characters that Dynamic Converter is generating. In order for Dynamic Converter to correctly code the character set into the HTML it generates, all templates should include a META tag that uses the `{## INSERT}` macro as follows:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset={## INSERT  
ELEMENT=pragma.charset}">
```

If the template does not include this line, the user will have to manually select the correct character set in their browser.

7.5.2 Pragma.CSSFile

This pragma is used to insert the name of the Cascading Style Sheet (CSS) file into HTML documents. This name is typically used in conjunction with an HTML `<LINK>` tag to reference styles contained in the CSS file generated by Dynamic Converter.

When used with the `{## INSERT}` macro, this pragma will generate the URL of the CSS file that is created. This macro must be used with `{## INSERT}` inside every template file that inserts contents of the source file and when the selected HTML flavor supports CSS. The CSS file will only be created if the selected HTML flavor supports CSS.

When used with the `{## IF}` macro, the conditional will be true if the selected HTML flavor supports Cascading Style Sheets or not.

If CSS is required for the output, `{## IF element=pragma.embeddedcss}` or `{## IF element=pragma.cssfile}` must be used. However, Dynamic Converter does not differentiate between the two, as the choice of using embedded CSS vs. external CSS is your decision and you may even wish to mix the two in the output.

An example of how to use this pragma that works when exporting either CSS or non-CSS flavors of HTML would be as follows:

```
{## IF ELEMENT=Pragma.CSSFile}  
  <LINK REL=STYLESHEET  
    HREF="{## INSERT  
    ELEMENT=Pragma.CSSFile}">
```

```
</LINK>  
{## /IF}
```

7.5.3 Pragma.EmbeddedCSS

This pragma is used to insert CSS style definitions in a single block in the <HEAD> of the document.

When used with the {## INSERT} macro, this pragma will insert the block of CSS style definitions needed for use later in the file. This macro must be used inside every output HTML file where {## INSERT} is used to insert document content.

When used with the {## IF} macro, the conditional will be true if the selected HTML flavor supports CSS.

If CSS is required for the output, {## IF element=pragma.embeddedcss} or {## IF element=pragma.cssfile} must be used. However, Dynamic Converter does not differentiate between the two, as the choice of using embedded CSS vs. external CSS is your decision and you may even wish to mix the two in the output.

If a style is used anywhere in the input document, that style will show up in the embedded CSS generated for all the output HTML files generated for the input file. Consider a template that splits its output into multiple HTML files. In this example, the input file contains the "MyStyle" style. It does not matter if during the conversion only one output HTML file actually references the "MyStyle" style. The "MyStyle" style definition will still show up in the embedded CSS for all the output files, including those files that never reference this style.

7.5.4 Pragma.JsFile

This pragma is used to insert the name of the JavaScript file into HTML documents. This name is typically used in conjunction with an HTML <SCRIPT> tag to reference JavaScript contained in the .js file generated by HTML Export.

When used with the {## INSERT} macro, this pragma will generate the URL of the JavaScript file that is created. This macro must be used with {## INSERT} inside every template file that inserts contents of the source file when:

1. The selected HTML flavor supports JavaScript.
2. The javascriptTabs option has been set to true.

The JavaScript file will only be created if the selected HTML flavor supports JavaScript.

When used with the {## IF} macro, the conditional will depend upon whether the selected HTML flavor supports JavaScript or not.

7.5.5 Pragma.SourceFileName

This pragma represents the name of the source document being converted.

Note: The Pragma.SourceFileName pragma does *not* include the path name.

7.6 Sample Script Templates

Dynamic Converter comes with a number of sample script templates that you can check into Content Server and begin using right away. The sample script templates are available in the `/ucm/Distribution/DynamicConverterSamples` directory.

The following sample layout templates are available:

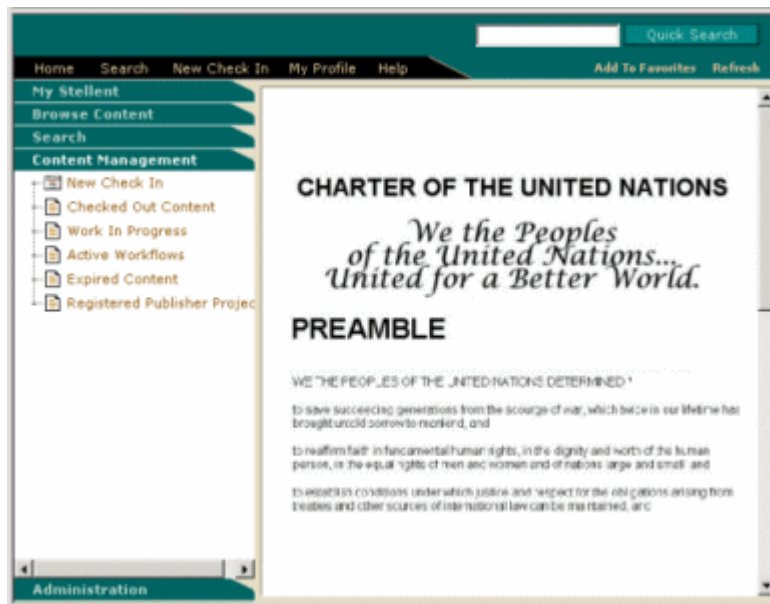
- [Basic](#)
- [Elements](#)
- [Plain](#)
- [SimpleToc](#)
- [Slideshow, Slideshowb, and Slideshowc](#)
- [Textout](#)

These sample script templates are provided as Hypertext Content Server Template (HCST) files.

7.6.1 Basic

Figure 7–2 shows an example of the Basic script template.

Figure 7–2 Example of the Basic Script Template



The Basic sample script template contains the following code:

```
<html>
<head>
{## if element=property.title}
  <title>{## insert element=property.title suppress=tags}</title>
{## else}
  <title>Converted {## insert element=pragma.sourcefilename}</title>
{## /if}

{## if element=pragma.cssfile}
  <link rel="stylesheet" href="{## insert element=pragma.cssfile}"</link>
```



```

{## /if}

<$defaultPageTitle="Converted Content"$>
<$include std_html_head_declarations$>
</head>

<$include body_def$>
<$include std_page_begin$>
<$include std_header$>

<table border="0" cellpadding="0" cellspacing="0" width="550">
<tr><td>

{## repeat element=sections}
  <p align="center">{## insert element=sections.current.bodyorimage
width=500}</p>
  <hr size="3"></hr>
{## /repeat}

</td></tr>
</table>

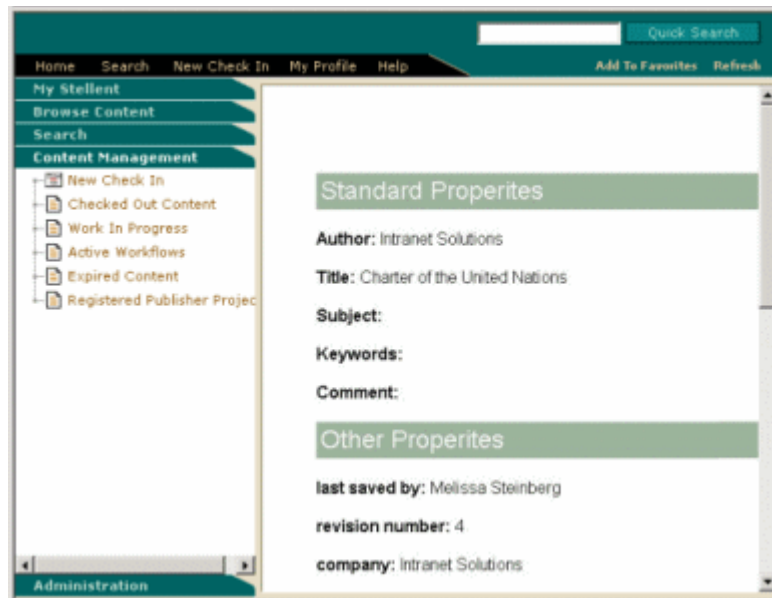
<$include std_page_end$>
</body>
</html>

```

7.6.2 Elements

Figure 7-3 shows an example of the Elements script template.

Figure 7-3 Example of the Elements Script Template



See [Appendix E, "Elements Script Template"](#) for a more elaborate explanation of the Elements template.

7.6.3 Plain

The Plain sample script template contains the following code:

```

<html>
<head>
{## if element=property.title}
  <title>{## insert element=property.title suppress=tags}</title>
{## else}
  <title>Converted {## insert element=pragma.sourcefilename}</title>
{## /if}

{## if element=pragma.cssfile}
  <link rel="stylesheet" href="{## insert element=pragma.cssfile}"</link>
{## /if}

<$defaultPageTitle="Converted Content"$>
<$include std_html_head_declarations$>
</head>

<$include body_def$>
<$include std_page_begin$>
<$include std_header$>

<table border="0" cellpadding="0" cellspacing="0" width="550">
<tr><td>
{## repeat element=sections}
  {## if element=sections.current.type value=wp}
    {## insert element=sections.current.bodyorimage width=500}

    {## if element=sections.current.footnotes.1.body}
    <br></br>
    {## repeat element=sections.current.footnotes}
    {## insert element=sections.current.footnotes.current.body}
    <br></br>
    {## /repeat}
  {## /if}

  {## if element=sections.current.endnotes.1.body}
  <br><br>
  {## repeat element=sections.current.endnotes}
  {## insert element=sections.current.endnotes.current.body}
  <br></br>
  {## /repeat}
  {## /if}
  {## else}
  <h1>{## insert element=sections.current.body.title suppress=tags}</h1>
  {## insert element=sections.current.bodyorimage width=500}
  <br></br><hr></hr><br></br>
  {## /if}
{## /repeat}

</td></tr>
</table>

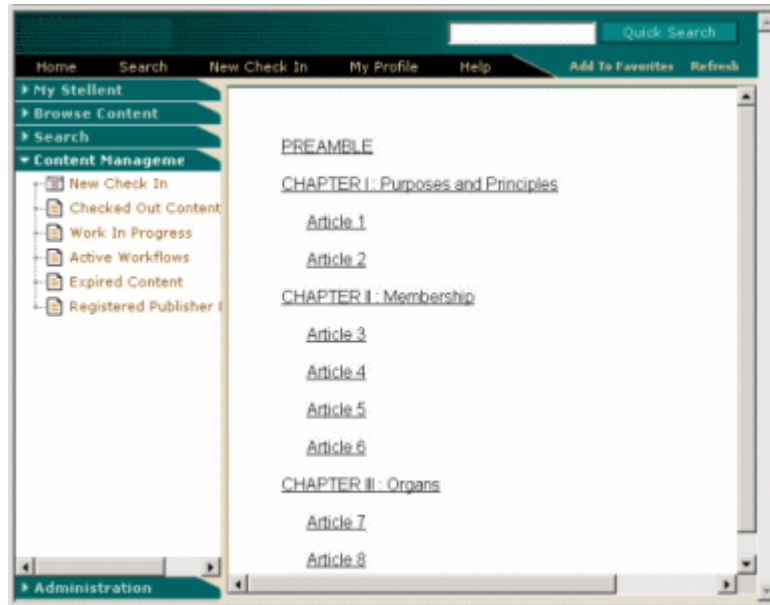
<$include std_page_end$>
</body>
</html>

```

7.6.4 SimpleToc

Figure 7–4 shows an example of the Simple TOC script template.

Figure 7–4 Example of the Simple TOC Script Template



The SimpleToc sample script template contains the following code:

```
<html>
<head>
{## if element=property.title}
  <title>{## insert element=property.title suppress=tags}</title>
{## else}
  <title>Converted {## insert element=pragma.sourcefilename}</title>
{## /if}

{## if element=pragma.cssfile}
  <link rel="stylesheet" href="{## insert element=pragma.cssfile}"</link>
{## /if}

<defaultPageTitle="Converted Content"$>
<include std_html_head_declarations$>
</head>

<include body_def$>
<include std_page_begin$>
<include std_header$>

<table border="0" cellpadding="0" cellspacing="0" width="550">
<tr><td>
  {## repeat element=sections}
  {## if element=sections.current.title}
    <a href="{## link element=sections.current.bodyorimage}">
      {## insert element=sections.current.title suppress=tags}
    </a>
    <br></br>
  {## /if}

  {## if element=sections.current.type value=wp}
```

```

        {## if element=sections.current.headings.1.body}
        {## repeat element=sections.current.headings}
        <a href="{## link element=sections.current.headings.current.body}">
            {## insert element=sections.current.headings.current.title
suppress=tags}
        </a>
        <br></br>

        {## if element=sections.current.headings.current.headings.1.body}
        {## repeat element=sections.current.headings.current.headings}
        &#160;&#160;&#160;&#160;&#160;&#160;&#160;
        <a href="{## link
element=sections.current.headings.current.headings.current.body}">
            {## insert element=sections.current.headings.current.headings.current.title
suppress=tags}
        </a>
        <br></br>
{## /repeat}
{## /if}
{## /repeat}
{## /if}
{## /if}
{## /repeat}

<hr size="3"></hr>
{## repeat element=sections}
{## insert element=sections.current.bodyorimage width=500}
<hr size="3"></hr>
{## /repeat}

{## if element=property.author}
    <p><b>Author:</b><br></br>
        {## insert element=property.author suppress=tags}
    </p>
{## /if}
<br></br>
{## if element=property.title}
    <p><b>Title:</b><br></br>{## insert element=property.title
suppress=tags}</p>
{## /if}
<br></br>
{## if element=property.subject}
    <p><b>Subject:</b><br></br>{## insert element=property.subject
suppress=tags}</p>
{## /if}
<br></br>
{## if element=property.keywords}
    <p><b>Keywords:</b><br></br>{## insert element=property.keywords
suppress=tags}</p>
{## /if}
<br></br>
{## if element=property.comment}
    <p><b>Comment:</b><br></br>{## insert element=property.comment
suppress=tags}</p>
{## /if}
<br></br>
</td></tr>
</table>

<$include std_page_end$>

```

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

7.6.5 Slideshow, Slideshowb, and Slideshowc

The Slideshow templates can be used to convert PowerPoint presentations. See ["Configuring Slideshow Template Files for PowerPoint Presentations"](#) on page 2-3.

Slideshow

The Slideshow sample script template contains the following code:

```
<html>
<head>
  {## if element=property.title}
    <title>{## insert element=property.title suppress=tags}</title>
  {## else}
    <title>Converted {## insert element=pragma.sourcefilename}</title>
  {## /if}

  {## if element=pragma.cssfile}
    <link rel="stylesheet" href="{## insert element=pragma.cssfile}"</link>
  {## /if}

  <$defaultPageTitle="Converted Content"$>
  <$include std_html_head_declarations$>
</head>

<$include body_def$>
<$include std_page_begin$>
<$include std_header$>

<script language="JavaScript"><!--
if (document.images)
{
  {## repeat element=sections}
  thumb{## insert number=sections.current.value} = new Image;
  thumb{## insert number=sections.current.value}.src = "{## insert
element=sections.current.image width=400 suppress=tags}";
  {## /repeat}
}

function swapem(iname,gname)
{
  if (document.images)
  {
    document.images[iname].src = eval(gname + ".src");
  }
}

function openawindow( pageToLoad, winName, width, height, center)
{
  /* Opens a new window on the users desktop.
  Arguments:
  pageToLoad - The URL of a page to load in the browser window.
               This can be a relative URL or fully qualified.
  winName -   Name of the new window.
  width  -   The horizontal size of the new window.
  height -   The vertical size of the new window.
  center -   toggle centering on 4.0 browsers.
```

1=centered window 0=no centering

Values in the "args" section below can all be toggled in the same fashion as the center toggle. Just modify the appropriate value in the args section to be either 0 or 1.

A call to this function might look like this:

```
<script>
<a href="javascript:openAWindow('ice.html','ice',375,250,1)"&gt;Ice</a>
```

Created by Glenn Davis of Project Cool, Inc. for general use. If you use this routine please leave all comments in place so that others may benefit as well.

```
*/
if ((parseInt(navigator.appVersion)) < 4){swidth = 640} else {swidth =
(screen.width-10)}
if ((parseInt(navigator.appVersion)) < 4){sheight = 480} else {sheight =
(screen.height-60)}

args = "width=" + swidth + ","
+ "height=" + sheight + ","
+ "location=0,"
+ "menubar=0,"
+ "resizable=1,"
+ "scrollbars=0,"
+ "status=0,"
+ "titlebar=0,"
+ "toolbar=0,"
+ "hotkeys=0,"
+ "screenx=" + 0 + "," //NN Only
+ "screeny=" + 0 + "," //NN Only
+ "left=" + 0 + "," //IE Only
+ "top=" + 0 ; //IE Only

window.open( pageToLoad,winName,args );
}

// --></script><div align="center"><center>

<table border="0" cellpadding="0" cellspacing="0" width="550">
<tr><td>

{## if element=property.title}
<p><h3>{## insert element=property.title suppress=tags}</h3></p>
{## /if}

{## if element=property.subject}
<p><h3>{## insert element=property.subject suppress=tags}</h3></p>
{## /if}

<p><h3>{## insert number=sections.count} slides by {## insert
element=property.author suppress=tags}</p>

<table border="0" cellpadding="0" cellspacing="0" width="550">
<tr>

<td valign="top">
<table>
```

```

        {## repeat element=sections}
        <tr><td>
            <a href="javascript:openwindow('{## link
element=sections.current.bodyorimage
template=slideshowbtemplate.hcst}','bigslide', 640,480,1)"
onMouseOver="swapem('thumbimg','thumb{## insert number=sections.current.value}')">
            <font size="1">{## insert element=sections.current.body.title
suppress=tags}</font>
            </a>
        </td></tr>
        {## /repeat}
    </td></tr>
</table>
</td>

<td>
</img>
</td>

<td valign="top">
    </img>
</td>
</tr>
</table>

</td></tr>
</table>

<$include std_page_end$>
</body>
</html>

```

Slideshowb

The Slideshowb sample script template contains the following code:

```

<html>

<head>
{## if element=property.title}
    <title>{## insert element=property.title suppress=tags}</title>
{## else}
    <title>converted {## insert element=pragma.sourcefilename}</title>
{## /if}
{## if element=pragma.cssfile}<link rel="stylesheet" href="{## insert
element=pragma.cssfile}"></link>{## /if}
</head>

<body bgcolor="BLACK" topmargin="0" leftmargin="0" scroll="No">

<p><center>
<a href="{## if element=sections.next.bodyorimage}{## link
element=sections.next.image}{## else}{## link template=slideshowctemplate.hcst}{##
/if}">
</img></a></center></p>
</body>
</html>

```

Slideshowc

The Slideshowc sample script template contains the following code:

```
<html>
<head>
{## if element=property.title}
  <title>{## insert element=property.title suppress=tags}</title>
{## else}
  <title>converted {## insert element=pragma.sourcefilename}</title>
{## /if}
{## if element=pragma.cssfile}<link rel="stylesheet" href="{## insert
element=pragma.cssfile}"></link>{## /if}
</head>
<body bgcolor="black">
<font color="white" size="+5">

<center>This is the end of the presentation.
<p><a href="{## link element=sections.1.bodyorimage
template=slideshowbtemplate.hcst}">Return to start of presentation.</a></p>
</center>

</font>
</body></html>
```

7.6.6 Textout

The Textout sample script template contains the following code:

```
<html>
<head>
{## if element=property.title}
  <title>{## insert element=property.title suppress=tags}</title>
{## else}
  <title>Converted {## insert element=pragma.sourcefilename}</title>
{## /if}

{## if element=pragma.cssfile}
  <link rel="stylesheet" href="{## insert element=pragma.cssfile}"></link>
{## /if}

<defaultPageTitle="Converted Content"$>
<include std_html_head_declarations$>
</head>

<$include body_def$>
<$include std_page_begin$>
<$include std_header$>

<table border="0" cellpadding="0" cellspacing="0" width="550">
<tr><td>

<h1><font color="0000FF">Document Body</font></h1>
{## repeat element=sections}
  <h2><font color="0000FF">Section </font>{## insert
number=sections.current.value}</h2>

  {## if element=sections.current.type value=ss}
    <p>
      <font color="0000FF"><b>Section Type</b>=&quot;</font>spreadsheet<font
color="0000FF">&quot;</font>
```



```

</p>
<p>
<font color="0000FF"><b>Title</b>=&quot;;</font>
{## insert element=sections.current.title}<font color="0000FF">&quot;;</font>
</p>
{## insert element=sections.current.bodyorimage width=500}
{## elseif element=sections.current.type value=pr}
<p>
<font color="0000FF"><b>Section Type</b>=&quot;;</font>presentation<font
color="0000FF">&quot;;</font>
</p>
<h3><font color="0000FF">Image</font></h3>
{## if element=sections.current.image}
  {## insert element=sections.current.image width=500}
{## else}
  <p><font color="0000FF">Image is empty</font></p>
{## /if}
<h3><font color="0000FF">Image Text</font></h3>
{## if element=sections.current.body}
  {## insert element=sections.current.body}
{## else}
  <p><font color="0000FF">Image text is empty</font></p>
{## /if}
{## elseif element=sections.current.type value=dr}
<p>
<font color="0000FF"><b>Section Type</b>=&quot;;</font>vector graphic
drawing<font color="0000FF">&quot;;</font>
</p>
<h3><font color="0000FF">Image</font></h3>
{## if element=sections.current.image}
  {## insert element=sections.current.image width=500}
{## else}
  <p><font color="0000FF">Image is empty</font></p>
{## /if}
<h3><font color="0000FF">Image Text</font></h3>
{## if element=sections.current.body}
  {## insert element=sections.current.body}
{## else}
  <p><font color="0000FF">Image text is empty</font></p>
{## /if}
{## elseif element=sections.current.type value=bm}
<p>
<font color="0000FF"><b>Section Type</b>=&quot;;</font>bitmap<font
color="0000FF">&quot;;</font>
</p>
<h3><font color="0000FF">Image</font></h3>
{## if element=sections.current.image}
  {## insert element=sections.current.image width=500}
{## else}
  <p><font color="0000FF">Image is empty</font></p>
{## /if}
<h3><font color="0000FF">Image Text</font></h3>
{## if element=sections.current.body}
  {## insert element=sections.current.body}
{## else}
  <p><font color="0000FF">Image text is empty</font></p>
{## /if}
{## elseif element=sections.current.type value=wp}
<p>
<font color="0000FF"><b>Section Type</b>=&quot;;</font>bitmap<font

```

```

color="0000FF">&quot;;</font>
</p>
<h2><font color="0000FF">Section Body</font></h2>
{## insert element=sections.current.contents width=500}
{## elseif element=sections.current.type value=ar}
<p>
<font color="0000FF"><b>Section Type</b>=&quot;;</font>archive<font
color="0000FF">&quot;;</font>
</p>
{## insert element=sections.current.bodyorimage width=500}
{## elseif element=sections.current.type value=db}
<p>
<font color="0000FF"><b>Section Type</b>=&quot;;</font>database<font
color="0000FF">&quot;;</font>
</p>
{## insert element=sections.current.bodyorimage width=500}
{## elseif element=sections.current.type value=ch}
<p>
<font color="0000FF"><b>Section Type</b>=&quot;;</font>chart<font
color="0000FF">&quot;;</font>
</p>
<h3><font color="0000FF">Image</font></h3>
{## if element=sections.current.image}
  {## insert element=sections.current.image width=500}
{## else}
  <p><font color="0000FF">Image is empty</font></p>
{## /if}
<h3><font color="0000FF">Image Text</font></h3>
{## if element=sections.current.body}
  {## insert element=sections.current.body}
{## else}
  <p><font color="0000FF">Image text is empty</font></p>
{## /if}
{## else}
  <p>
<font color="0000FF"><b>Section Type</b> is not one of wp, ss, pr, dr, bm,
db, ch, or ar<font></p>
</p>
  {## insert element=sections.current.bodyorimage width=500}
  {## /if}
{## /repeat}

</td></tr>
</table>

<$include std_page_end$>
</body>
</html>

```

7.7 Setting Script Template Formatting Options

You can control formatting options for script templates by editing the Script Template Conversion Configuration Settings (see ["Script Template Conversion Configuration Settings"](#) on page A-7) on the Dynamic Converter Configuration page (see ["Dynamic Converter Configuration Page"](#) on page A-2).

The settings that you can change include:

- ["Changing the Format Used for Converted Graphics"](#) on page 7-43

- ["Generating Bullets and Numbers for Lists"](#) on page 7-43

7.7.1 Changing the Format Used for Converted Graphics

If you want to change the format to be used for converted graphics, edit the following option:

```
# SCCOPT_GRAPHIC_TYPE
#
# Determines what graphic format will be used for exported graphics.
# Setting this to "none" disables graphic output.
#
graphicstype    gif
#graphicstype   jpeg
#graphicstype   png
#graphicstype   none
```

Lines that begin with "#" have been commented out. So the above example shows the default setting, with the gif format selected. To use the jpeg format, instead, you would simply comment the first line and uncomment the second line, thus:

```
#graphicstype   gif
graphicstype    jpeg
#graphicstype   png
#graphicstype   none
```

7.7.2 Generating Bullets and Numbers for Lists

If you want to generate bullets and numbers for lists instead of HTML list tags, you would edit the following option:

```
# SCCOPT_GENBULLETSANDNUMS
#
# Generate Bullets and Numbers.  Bullets and numbers will be generated for
# lists instead of using HTML list tags (<ol>, <ul>, <li>, etc.) when
# rendering lists in a document.
#
genbulletsandnums    no
#genbulletsandnums   yes
```

Again, comment one line and uncomment another, thus:

```
#genbulletsandnums    no
genbulletsandnums     yes
```

7.8 Breaking Documents by Structure

One of the most powerful features of the template architecture is the ability to break long word processor documents up into logical pieces and create powerful navigation aids to access them.

To understand how this is done, you must first understand the document tree as it relates to word processing documents. The somewhat complex graphic below attempts to show how the elements in the tree relate to a real-world document (see figure below).

The following are some examples of elements and the data they would produce if run against the document shown in the preceding image. Note the omission of the default nodes **body** and **contents** in the second two examples:

body.contents.headings.2.body.title

would produce "Present Day."

body.contents.headings.2.body.contents.headings.1.body.title

would produce "Commercial."

body.contents.preface

would produce "The History of Flight" and the text below it, up to but not including "Introduction."

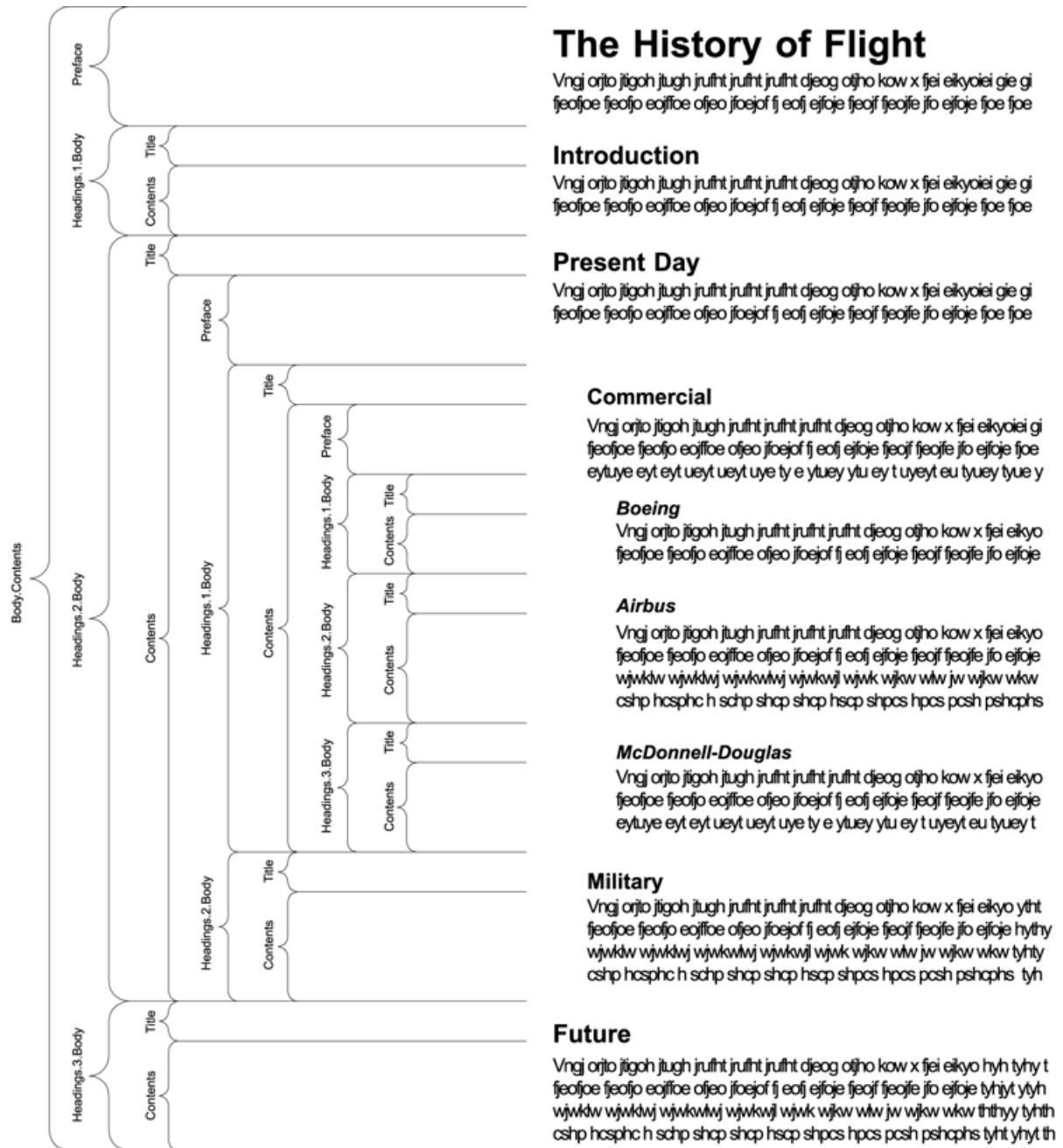
headings.2.headings.1.headings.3.title

would produce "McDonnell-Douglas."

headings.2.headings.1.headings.3.contents

would produce the text below "McDonnell-Douglas" but above "Military."

Figure 7-5 Breaking up documents by structure



Breaking documents requires that Dynamic Converter understands the logical divisions in the structure of a document. Currently the only formats that can give Dynamic Converter this information in an unambiguous manner are Microsoft Word 95 and higher and WordPerfect 6.0 and higher. In these formats, the breaking information is available if the author placed table-of-contents information in the document. Refer to the appropriate software manual for information on the necessary procedure for including this information. That is not to say that the document must have a table of contents, only that the information to build one must be present.

It should be noted that some word processing formats, including Microsoft Word 2002 (XP), allow users to specify TOC entries in multiple ways. Dynamic Converter only supports two of these methods:

TOC specified through...	Supported in Dynamic Converter?
Applied heading styles	Yes
Custom styles with outline levels	Yes
Outline level applied as a paragraph attribute	No
TOC entries	No

Additionally, if a heading style is applied to text inside a table in the original document, Dynamic Converter will not break on that heading. This is because Dynamic Converter will not break within tables.

Indexes and Structure-Based Breaking

All repeatable nodes have an associated index variable that has a current value at any given time in the conversion process. For elements that contain repeatable nodes as part of their path, the instance of the repeatable element must be specified by using a number or one of several index variable keywords. See "[Index Variable Keywords](#)" on page 7-9 for more information on the possible values for the index variables.

7.9 Breaking Documents by Content Size

In addition to breaking documents by structure (see "[Setting Script Template Formatting Options](#)" on page 7-42), Dynamic Converter also supports breaking documents based on the amount of content to be placed in each output file or "page." Documents can even be broken based on both their structure *and* content size.

To break documents by content size, two things must be done. First, the SCCOPT_EX_PAGESIZEpageSize option must be set (see "[Setting Options Within the Template: {## OPTION}](#)" on page 7-25). The second thing that must be done is that the template used must be equipped with the {## UNIT} construct (see "[Units: {## UNIT}, {## HEADER}, and {## FOOTER}](#)" on page 7-13).

The basic idea behind the unit template construct is to tell Dynamic Converter what things should be repeated on every "page" and what pieces should only be shown once. In other words, the unit template construct provides a mechanism for grouping template text and document elements. Unit boundaries are used when determining where to break the document when spanning pages.

Here are some examples of the kinds of things the template author might want to appear on every page:

- The <META> tag inserting the output document character set.
- A company copyright message.
- Navigational elements to link the previous/next pages together.

Typical examples of things that would not go on every page would be:

- The actual content of the document.
- Structural navigational elements like the links for a table of contents.

A unit consists of a header, a footer (both of which are optional), and a body. Items that are to be repeated at the beginning or end of every unit should be placed in the header or footer respectively.

A unit is delimited by the {## UNIT} template macro. Similarly, the {## HEADER} and {## FOOTER} template macros delimit the header and footer respectively. The body is

everything that is left between the header and the footer. The `{## UNIT}` macro must be the first macro in the template. The body frequently contains nested units. The body may be empty.

To ensure that the header is the first item in the template and the footer is the last item, text between the `{## UNIT}` tag and the `{## HEADER}` tag will be ignored, as will text between the `{## /FOOTER}` tag and the `{## /UNIT}` tag, including whitespace. The header and footer of a unit will be output in every page containing that unit, enclosing that portion of the unit's body that is able to fit in a particular page. The entire template is a unit that may contain additional units.

7.9.1 A Sample Size Breaking Template

By way of example, let's take another look at the very simple template from [About Script Templates](#). To make things more interesting, let's insert the character set into the template with a `<meta>` tag. Let's also insert some better navigation to improve movement between the pages. The modified version of the template is as follows:

```
{## unit}{## header}
<html><head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html;
charset={## insert element=pragma.charset}" /></head>
<body>
{## anchor aref="prev" format="<p><a href=\"%url\">Prev</a></p>"}
{## /header}
<p>Here is the document you requested.
{## insert element=property.title} by
{## insert element=property.author}</p>

<p>Below is the document itself</p>
{## insert element=body}
{## footer}
{## anchor aref="next" format="<p><a href=\"%url\">Next</a></p>"}
</body>
</html>
{## /footer}{## /unit}
```

A very small value (about 20 characters) is used for the page size option. The resulting HTML might look like this (HTML that is the result of a macro is in **bold**):

file1.htm

```
<html><head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ASCII" /></head>
<body>
<p>Here is the document you requested.</p>
<p>A Poem by Phil Boutros</p>
<p><a href="file2.htm">Next</a></p>
</body>
</html>
```

file2.htm

```
<html><head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ASCII" /></head>
<body>
<p><a href="file1.htm">Next</a></p>
<p>Below is the document itself</p>
<p>Roses are red</p>
<p>Violets are blue</p>
```

```
<p><a href="file3.htm">Prev</a></p>
</body>
</html>
```

file3.htm

```
<html><head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=us-ASCII" /></head>
<body>
<p><a href="file2.htm">Prev</a></p>
<p>I'm a programmer</p>
<p>and so are you</p>
</body>
</html>
```

There are several things to note here:

- The page size option value does not apply to the text from the template, only the text inserted from the source document. Each page contains roughly 20 characters of visible input document text.
- The {## INSERT} of the character set is part of the {## HEADER} and therefore is inserted into all the output pages.
- Text from the body of the unit is inserted sequentially. Thus "as is" template text such as the line "<p>Below is the document itself</p>" is only inserted once.
- The {## ANCHOR} tags only insert links to the previous/next page if there actually is a previous/next page. Thus the first page does not have a link to the non-existent previous page.

7.9.2 Templates Without {## UNIT} Macros

The {## UNIT} macro is only required in templates that are designed to break pages based on size using the SCCOPT_EX_PAGESIZEpageSize option. An example of a template that would not perform any size-based breaking is one that defines an HTML <FRAME>, but does not include any document content. Another example where size-based breaking might not be desired is a table of contents page, even though a table of contents page does contain document content.

A template that does not conform to the {## UNIT} format is a not a size-based breaking template. Support for this type of template will continue for the indefinite future. The template will be considered to not be a size-based breaking template if the first macro tag encountered is something other than {## UNIT}. This means that there cannot be any {## UNIT}, {## HEADER} or {## FOOTER} macros later in the template. The value of the SCCOPT_EX_PAGESIZEpageSize option will be ignored for this type of template.

7.9.3 Indexes and Size-Based Breaking

As mentioned earlier, all repeatable nodes have an associated index variable. See "[Index Variable Keywords](#)" on page 7-9 for information about using index variable keywords such as "Next" and "Last."

7.10 Using Grids to Navigate Spreadsheet and Database Files

In order to support spreadsheets (and database files, though they are not as common), a template-based navigation concept known as a "grid" is available. Grids offer a way to consistently navigate a spreadsheet or database in an intuitive fashion.

Grids can be used to present the output of large spreadsheets in smaller pieces, so that less scrolling is necessary. It can also be used to help prevent the HTML versions of large spreadsheets from overwhelming browsers, potentially causing them to lock up. Grids can also be used to halt processing of large spreadsheets before they waste too much CPU time.

To use grids, you should use the new grid template element (see "[Element Definitions](#)" on page 7-5). Grids may only be used in templates that have been enabled with the `{## UNIT}` template macro. It is also important to set the grid-related options (see "[Setting Options Within the Template: {## OPTION}](#)" on page 7-25).

The grid support has some important limitations:

1. The output file format and flavor are expected to supports tables, although this is not required.
2. Grids are only used when converting spreadsheets and database input files. Grids are not available for word processing files at this time.
3. Due to size constraints, grid support works best if the contents of the cells in the input file do not make use of a lot of formatting (bold, special fonts, text color, etc.).

To further explain the grid system, consider a multi-sheet spreadsheet workbook as an example. Each sheet in the spreadsheet workbook is broken into a collection of grids. Each grid has a fixed maximum size and is a rectangular portion of the spreadsheet. The size of the grid is specified as a number of spreadsheet cells. For example, consider the 7 x 10 spreadsheet in [Figure 7-6](#).

Figure 7-6 Example 7 X 10 Spreadsheet

A1	B1	C1	D1	E1	F1	G1
A2	B2	C2	D2	E2	F2	G2
A3	B3	C3	D3	E3	F3	G3
A4	B4	C4	D4	E4	F4	G4
A5	B5	C5	D5	E5	F5	G5
A6	B6	C6	D6	E6	F6	G6
A7	B7	C7	D7	E7	F7	G7
A8	B8	C8	D8	E8	F8	G8
A9	B9	C9	D9	E9	F9	G9
A10	B10	C10	D10	E10	F10	G10

If you wanted to break it up into 3 x 4 grids, nine grids would be produced as shown in [Figure 7-7](#).

Figure 7-7 Example 7 x 10 Spreadsheet Split Up in 3 X 4 Grids

A1	B1	C1
A2	B2	C2
A3	B3	C3
A4	B4	C4

D1	E1	F1
D2	E2	F2
D3	E3	F3
D4	E4	F4

G1
G2
G3
G4

A5	B5	C5
A6	B6	C6
A7	B7	C7
A8	B8	C8

D5	E5	F5
D6	E6	F6
D7	E7	F7
D8	E8	F8

G5
G6
G7
G8

A9	B9	C9
A10	B10	C10

D9	E9	F9
D10	E10	F10

G9
G10

Normally, all grids have the same number of cells. The exception is that grids at the right or bottom edge of the spreadsheet may be smaller than the normal size. Grids will never be larger than the requested size. For this reason, grids can easily be navigated by using "up," "down," "left," or "right." One thing that grids cannot do is address individual cells in a spreadsheet (except, of course, in the degenerate case of a grid whose size is 1 x 1).

Dynamic Converter does not force deck/page breaks between each grid. Therefore, if the template writer wants to limit each deck/page to only one grid, they should force the break in the template.

Grid Support When Tables Are Not Available

Not all output flavors supported by Dynamic Converter support the creation of tables. If the output flavor does not support tables, Dynamic Converter will still support grids. However, Dynamic Converter's normal non-table output will be what is presented in grid form. For example, if "[A1]" represents the contents of cell A1, then we would export the following for a grid of size (2 x 2):

If `grids.1.body` is:

```
[A1]
[A2]
[B1]
[B2]
```

then `grids.right.body` is:

```
[C1]
[C2]
[D1]
[D2]
```

and `grids.down.body` is:

```
[A3]
[A4]
[B3]
```

[B4]

This section covers the following topics:

- ["About HTML Snippets"](#) on page 8-1
- ["Portal-Style Website Sample"](#) on page 8-1
- ["Combining HTML Snippets Into a Web Page"](#) on page 8-2
- ["Inline Dynamic Conversion"](#) on page 8-4
- ["Displaying Content Server Metadata on a Web Page"](#) on page 8-4

8.1 About HTML Snippets

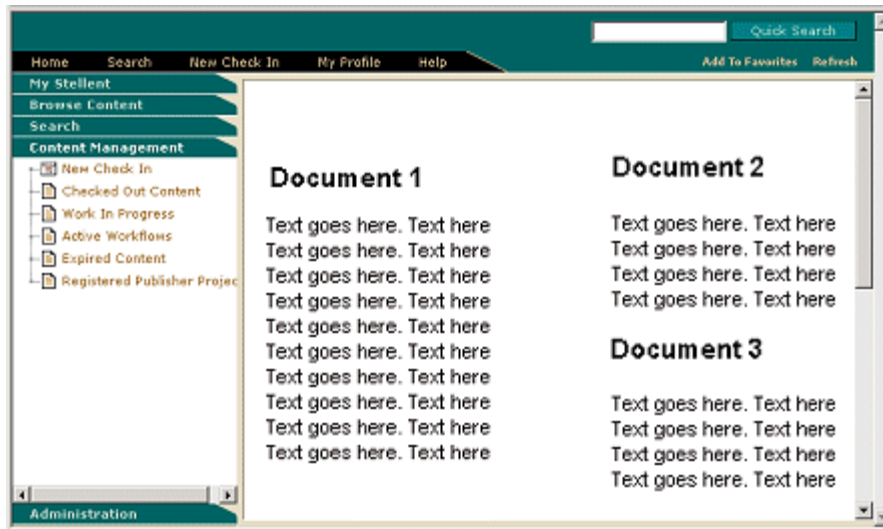
In earlier versions of Dynamic Converter (prior to version 6.0) and Content Server, a content item could be checked in, dynamically converted to HTML, and displayed as a web page, by itself, to the user. For the purposes of turning a native business document into a web-viewable version (for everyone to see), the solution was there. However, if you wanted to combine information from numerous source content items onto one web page (perhaps marketing information from one document, press releases from another document, or client feedback from a form) you would need to manually create such a document and convert it to a web page. There was no method for combining multiple content items and displaying them on the same web page.

Dynamic Converter now resolves this problem and, as a result, allows you to create powerful, information-rich web pages for your users. With the current version templates and the new Idoc Script function, you can now pull dynamically converted material from multiple content items or even portions of those content items and display them back to the user as a single web page (combined HTML snippets). Furthermore, you can specify a particular version, template, and layout file for each of the included content items.

The result is a portal-style website (see ["Portal-Style Website Sample"](#) on page 8-1) that draws dynamically generated content from any number of sources.

8.2 Portal-Style Website Sample

The illustration below demonstrates a simple portal-style web page that wraps a Content Server environment (borders and navigation) around four pieces of dynamically converted content. Each piece of content is actually a document checked into Content Server. There are many pieces coming together "on the fly," but to the user, it is a single and seamless web page.

Figure 8–1 Portal-style Web Page With Html Snippets

To create a page like this or a similar one where you are combining HTML snippets of code, you need to customize the dynamic conversion of your content items so that they can be displayed on the same web page. You do this by creating a template or layout file that will strip the TOP, HEAD, and BODY tags out of the dynamically converted HTML file (see step 1 in ["Combining HTML Snippets Into a Web Page"](#) on page 8-2).

Then use the new Idoc Script function to call your HTML snippets by content ID, version, template, and layout. All of this can be specified in the following Idoc Script tag:

```
<$incDynamicConversion(Content ID, revisionselectionmethod, template, layout)$>
```

8.3 Combining HTML Snippets Into a Web Page

To combine your HTML snippets into a single web page, complete the following steps:

1. Generate a snippet of HTML from a content item (see ["Generating a Snippet of HTML"](#) on page 8-2).
2. Call the snippet of HTML into the host page (HCST file) using Idoc Script (see page ["Include HTML Snippet Using Idoc Script Function"](#) on page 8-3).

8.3.1 Generating a Snippet of HTML

You can generate a snippet of HTML from a content item using either of two methods:

1. Making the conversion template XML-compliant
2. Create a layout template with body content only

Method 1: Make the Conversion Template XML-Compliant

By making your template XML-compliant, you remove the standard HTML tags that are placed at the beginning and end of a web page (<HTML>, <HEAD>, <BODY>, etc.). To create an XML-compliant Classic HTML Conversion template, perform the following steps:

1. Open the Dynamic Converter Admin page (see ["Dynamic Converter Admin Page"](#) on page A-1).

2. Click Edit Template.

The Edit Templates page is displayed (see ["Edit Templates Page"](#) on page A-14).

3. Enter the content ID for the desired template in the **Template text box or select your desired template from the **Available Templates** menu.****4. Click Edit Template.**

(Instead of updating an existing Classic HTML Conversion template, you may want to create a new template designed specifically for HTML snippets.

The Template Editor is started (see ["Classic HTML Conversion Template Editor"](#) on page 5-9).

5. Click Globals.**6. In the Globals dialog, click the **Options** tab.****7. Select the **Generate XML compliant output** checkbox to enable this Element.****8. Click **OK** to close the Globals dialog and click **OK** again to close the Template Editor.**

Your conversion template will now create the required HTML code from a content item so that it can be easily included in a separate web page (using an Idoc Script function; see ["Include HTML Snippet Using Idoc Script Function"](#) on page 8-3).

Method 2: Create a Layout Template with Body Content Only

Another way to remove the standard HTML tags that are placed at the beginning and end of a web page (<HTML>, <HEAD>, <BODY>, etc.) is to specify a layout template that places only the contents of the BODY tag in your converted HTML file. You can accomplish this by placing the following code (called a token), by itself, in a layout template:

```
<!-- TRANSIT - CUSTOMLAYOUT(BODY) -->
```

You can also use the sample layout template "snippet_layout.txt" (see ["snippet_layout.txt"](#) on page 6-4), which contains the necessary code. This file is located in the */ucm/Distribution/DynamicConverterSamples* directory.

Whether you choose to create a Classic HTML Conversion template or layout template for the purpose of HTML snippets, it is not necessary to associate either template with your content items. You can specify the appropriate template to use with an Idoc Script function; see ["Include HTML Snippet Using Idoc Script Function"](#) on page 8-3).

8.3.2 Include HTML Snippet Using Idoc Script Function

After you have created your snippets of HTML from your content items (see ["Generating a Snippet of HTML"](#) on page 8-2), you are ready to reference the content items from another web page. The way to do this is to use the following Idoc Script function in an HCST file:

```
<$incDynamicConversion(Content ID, revisionselectionmethod, template, layout)$>
```

This Idoc Script function references your content items by content ID, version, template, and layout (layout template). For example, if you want to include the latest version of a "Sales" document using the "Business" Classic HTML Conversion template and "snippet_layout" layout template into your web page, you would use the following code:

```
<$incDynamicConversion("Sales", "latest", "Business", "snippet_layout")$>
```

If you used an XML-compliant template to create an HTML snippet of code from a content item (see ["Method 1: Make the Conversion Template XML-Compliant"](#) on page 8-2), then you do not need to specify a layout template that creates the same HTML snippet effect. You can, instead, pass a blank parameter in the Idoc Script function: `<$incDynamicConversion("Sales","latest","Business","")$>`.

For your convenience, we have included a sample layout template called "snippet_layout.txt" (see ["snippet_layout.txt"](#) on page 6-4), which is located in the `/ucm/Distribution/DynamicConverterSamples` directory. This sample file includes the basic ingredients for a portal-style web page that draws information (HTML snippets) from other content items stored in the Content Server. (The results appear very similar to the illustration in [Portal-Style Website Sample](#)).

The file contains the following parts:

- Header and footer information that displays Content Server borders and navigation
- HTML tables to control the layout of the portal page (two columns in the top table cell and one column in the bottom)
- Three Idoc Script functions to pull three separate pieces of content into the portal page (referencing a version, template, and layout file in each)

You can start with this portal web page example and then customize it to fit your needs.

8.4 Inline Dynamic Conversion

Dynamic Converter includes an Idoc Script extension that allows you to convert a native document into an HTML snippet without referencing a Classic HTML Conversion template or layout template. The conversion is the same as using a blank Classic HTML Conversion template with a layout template that specifies body content only. (You cannot, however, modify the conversion template or layout template used in this conversion.)

You can convert native documents this way by using the following Idoc Script code:

```
incInlineDynamicConversion(dDocName, Revision_Selection_Method)
```

Place the content ID of the native document in the parenthesis along with the revision. For example, if the native document has the content ID "SalesDoc," then the complete Idoc Script syntax would be:

```
<$incInlineDynamicConversion("SalesDoc", "Latest")$>
```

or

```
<$incInlineDynamicConversion("SalesDoc", "LatestReleased")$>
```

This type of conversion is useful for converting native documents into HTML snippets without having to specify a Classic HTML Conversion template and a layout template in the Content Server.

8.5 Displaying Content Server Metadata on a Web Page

Dynamic Converter includes an Idoc Script extension that allows you to make Content Server metadata for a document available on the converted page. To use this Element, insert the following code into your conversion template:


```
dcLoadDocInfo()
```

You can add this code to a layout template (see [Chapter 6, "Classic HTML Conversion Layout Templates"](#)) or a Classic HTML Conversion template (see [Chapter 5, "HTML Conversion Templates"](#)), depending on how you are using these templates for your document conversion. It is important that this code be placed before any part of the web page attempts to use the document metadata.

The simplest solution is to add it to the very top of a layout template. This way, any part of the web page that tries to use the Content Server metadata for the document will work. You can also add this code to any of the sections of a Classic HTML Conversion template (for more information, see "Including HTML or scripting code in a web page" in the Template Editor help).

Working With Converted Content

This section covers the following topics:

- "Viewing Content Information" on page 9-1
- "Viewing a Converted File" on page 9-3
- "Previewing a Document Before Check-In" on page 9-5

9.1 Viewing Content Information

Every content item checked into the Content Server has its own content information page, which can be used to view and verify the metadata information about the content item, such as the content ID, title, author, and other metadata. You will frequently visit the content information page of your source documents in order to specify your template selection rule criteria.

The Info icon on the search results page is used to access the content information page of a content item, where you can view the metadata for the content item. Use this page to view and verify information about a specific content item. For example, you can identify the release date of a file or the user login of the author.

Figure 9-1 Content Information Page

Content Information Content Actions E-mail

Content ID: Executive_overview
Revision: 1
Type: ADACCT - Acme Accounting Department
Title: Executive Overview
Author: sysadmin
Comments:
Template Type:
Security Group: Public
Checked Out By:
Status: Released
Formats: application/msword

Links

Web Location: http://SCSTEST7/stellent/groups/public/documents/adacct/executive_overview.doc
Native File: [dc_sample.doc](#)

Revision	Release Date	Expiration Date	Status	Actions
[1]	3/6/07 5:02 PM	None	Released	Delete

This page shows a lot of information about the content item, including:

- Values for all the metadata fields that were completed when the file was checked into the Content Server
- The author's name (user login)
- The file status indicating where the file is in its life cycle
- The file format, which is the native application that the file was created with. The file format is expressed as the MIME content type.
- The current web location, which is an active link that points to the web-viewable rendition (for example, PDF) of the checked-in content item, if such a rendition was generated. This URL uniquely refers to the web-viewable rendition of the content item's latest revision.
- A native file link, which you can use to get a copy of the content item in its native format (that is, the one it was originally created in). If you click the link, you can open the file in its native application (if you have it installed on your computer) or you can save it to your local hard drive. You can also right-click the link and save the file locally. This enables you to make a copy of the file for reuse. You can then check it back into the Content Server as a new revision.
- The complete revision history.

Note: The content information can be displayed for any revision of the content item by clicking the revision link that is displayed in the Revision column of the Revision History section. The currently displayed content item is enclosed in square brackets: [].

Figure 9–2 Revision History of Content Item

Revision History				
Revision	Release Date	Expiration Date	Status	Actions
[2]	1/23/07 5:39 PM	None	Done	Delete
1	8/23/06 1:19 PM	None	Released	Delete

The content information page has other functions in addition to viewing a file's metadata, status, and revision history. The available options depend on your assigned privileges and the Content Server configuration, and may include any of the following:

Action	Definition
Check Out	Enables you to check out a file for edit and later check it in with the same content ID and the revision number incremented by one (if you are a contributor).
Undo Check Out	Cancels the check-out of the content item. Your name will no longer appear next to "Checked out by: on the content information page. You can only undo a check-out of a content item that you checked out if you have the "admin" role or have administrator permissions for the security group that the content item belongs to.
Check In	Checks in a new revision of a content item currently checked out.

Action	Definition
Update	Enables you to change the metadata fields for a content item already checked into the Content Server. For example, you can use Update to correct a misspelled word in the title field or select the correct content type if you initially entered it incorrectly.
Check In Similar	Enables you to check in another content item with the same metadata of the content item you have just checked in.
Send link by e-mail	Opens your e-mail program with a new message that contains a link to the URL (web address) of the web-viewable file.
Subscribe	Enables you to tag a content item so that you are automatically notified by e-mail about any changes to it (i.e., if a new revision is checked in). If the software does not know your e-mail address, you are prompted to enter it.
Unsubscribe	Enables you to cancel your subscription to the content item (i.e., no longer be notified of new revisions).
Create Shortcut	Enables you to create a shortcut to the content item in the Content Server and store the shortcut in a folder under Browse Content.
Delete Revision	Enables you to remove a revision of a file from the system. To delete a revision, you must have delete permission for the security group the file belongs to.
Revision Number	Displays the content information for the specified revision.

To access the content information page of a content item, complete the following steps:

1. Search for the content item.
The search results page is displayed.
2. Click the Info icon (Figure 9-3) that corresponds to the file for which you want to see the content information.

Figure 9-3 Info Icon



The content information page is displayed.

Note: See the *Content Server User Guide* for more information on searching for content.

9.2 Viewing a Converted File

Dynamic Converter provides a solution to the problem of requiring a client workstation to have native applications installed (such as Microsoft Word, Excel, or other applications) in order to open source documents created with those applications. It does this by creating a web-viewable version of the source document on demand and on the fly.

The web-viewable version of the source document can be seen by clicking an HTML link on these Content Server pages:

- [Search Results Page](#)
- [Content Information Page](#)

9.2.1 Search Results Page

You can use Content Server's extensive search Element to find content items. You can search by metadata and/or perform a full-text search (depending on the Content Server setup). The results of a search are shown on a search results page. If a content item in the list is of a file type that is supported and enabled for HTML conversion, then an **HTML Rendition** link is included in the actions popup menu. You can use this link to view an HTML rendition of the content item.

Figure 9–4 *Html Rendition Link on Search Results Page*

Search Results Found 7 items

Change View Query Actions

ID	Title	Date	Author	Actions
ppt_sample	PP Sample	3/6/07	sysadmin	[Icons]
plainscript	plain script	3/6/07	sysadmin	[Icons]
default	default	3/6/07	sysadmin	[Icons]
acclaim	Acclaim	3/6/07	sysadmin	[Icons]
Executive_overview	Executive Overview	3/6/07		[Icons] [Menu]
install_guide	Acme Engineering Installation Guide	2/23/07		[Icons]
exec_temp	Executive Template	2/22/07		[Icons]

Content Information
 Check Out
 Get Native File
 Check In Similar
 Send link by e-mail
HTML Rendition

When you click the **HTML Rendition** link, the file is converted and displayed using the rules and templates specified on the Template Selection Rules page (see "[Template Selection Rules Page](#)" on page A-9).

9.2.2 Content Information Page

Every content item checked into Content Server has its own content information page, which shows the metadata information of the content item, such as the content ID, title, author, and other metadata.

If the content item is of a file type that is supported and enabled for HTML conversion by Dynamic Converter, then the content information page will display an **(HTML)** link beside the text "Get Conversion." You can use this link to view an HTML rendition of the content item.

Figure 9–5 *Html Link on Content Information Page*

Content Information

Content ID: Executive_overview
Revision: 1
Type: ADACCT - Acme Accounting Department
Title: Executive Overview
Author: sysadmin
Comments:
Template Type:
Security Group: Public
Checked Out By:
Status: Released
Formats: application/msword

Links

Web Location: http://SCSTEST7/stellent/groups/public/documents/adacct/executive_overview.doc
Get Conversion (HTML)
Native File: [dc_sample.doc](#)

Revision	Release Date	Expiration Date	Status	Actions
[1]	3/6/07 5:02 PM	None	Released	Delete

When you click the **(HTML)** link, the file is converted and displayed using the rules and templates specified on the Template Selection Rules page (see "[Template Selection Rules Page](#)" on page A-9).

Subscription and Workflow Notifications

You can also open the content information page using the **View Info** link in the e-mail messages that you receive when you subscribe to a content item stored in the Content Server.

Figure 9–6 *View Info Link in Subscription E-mail Notification Message*

Content Release Notification

New revisions of the following content items have recently been checked in:

- **Content Publisher 8.0 (SCP-8.0)**
[[View Content](#)] [[View Info](#)]

This same link is available in workflow notification messages, which eliminates the need for content reviewers to have the native application used to create the source file.

9.3 Previewing a Document Before Check-In

Content contributors can preview the HTML rendition of a document before checking it into the Content Server. This enables them to see if there are problems with the document or the template associated with the document, and notify the site webmaster or developer. Problems can then be resolved before more users or customers view the converted content. Both the content authors and the site developers gain from the ability to preview documents this way.

The dynamic contributor preview is displayed as an **(HTML)** button on Content Server's content check-in page.

Figure 9–7 *Html Preview Button on Content Check-in Screen*



Once a document has been selected and all metadata assigned to the document, click the preview button to see how the document will appear as a web page. The resulting screen displays a **Complete Check In** link in the left frame and the converted document in the right frame.

Figure 9–8 *Dynamic Conversion Preview*



If you are satisfied with the HTML rendition of the document, you can click **Complete Check In** to check the document into the Content Server (at which time you are brought to the check-in confirmation screen). Click the Back button in your web browser to cancel the process and return to the content check-in screen.

If you check in a document using metadata that has no template associated with it, a blank Classic HTML Conversion template is assigned. This template contains no special formatting instructions, other than to convert your document into a web page.

Tip: As a site administrator, you can also preview how a content item will appear with a particular template using the Change Preview button in the Template Editor.

Implementation Considerations

This section covers some of the more pragmatic concerns when dealing with Dynamic Converter. The following topics are explained:

- ["Metadata Fields With Multi-Byte Characters"](#) on page 10-1
- ["Conversion of PDF Files in UNIX"](#) on page 10-2
- ["Embedded Graphics on UNIX"](#) on page 10-2
- ["Use of Vector Versus Raster Graphics Formats"](#) on page 10-2
- ["Converting Vector Graphics and Spreadsheet Text in UNIX"](#) on page 10-3
- ["URL Rewriting"](#) on page 10-3
- ["Relative URLs in Templates and Layout Files"](#) on page 10-4
- ["Browser Caching"](#) on page 10-4
- ["Image Sizing Rules"](#) on page 10-5
- ["CSS Considerations"](#) on page 10-5
- ["Style Names Used by Dynamic Converter"](#) on page 10-5
- ["Overriding Dynamic Converter Styles"](#) on page 10-6
- ["Pragma.CSSFile and {## LINK}"](#) on page 10-6
- ["Well-Formed HTML"](#) on page 10-7
- ["Positional Frames Support"](#) on page 10-7
- ["Template Writing Tips"](#) on page 10-7

10.1 Metadata Fields With Multi-Byte Characters

For Classic HTML Conversion templates, it is recommended that you do not use multi-byte characters in your content IDs, security groups, content types, and account names, even if Dynamic Converter is used in a multi-byte environment (Japanese, Korean, or other non-Roman alphabets). This content metadata information is included in the URL of a content item, and limitations in current web technology may prevent web servers and web browsers from handling multi-byte character URLs correctly. (Dynamic Converter, for example, will fail to locate content items if the links are broken.)

If you want to use multi-byte characters in content IDs, security groups, content types, or accounts, you need to make sure that the *entire* Content Server environment (servers and clients) runs on operating systems that support multi-byte languages (for example, Japanese or Korean versions of Microsoft Windows).

Note: The new Dynamic Converter HTML Template Editor will not support working with multi-byte content IDs for the templates or the content items being converted.

10.2 Conversion of PDF Files in UNIX

Conversion of PDF files under UNIX may be slow and may time out after three minutes (the default timeout value for the conversion process).

To increase the conversion timeout, complete the following tasks:

1. Open the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).
2. Click **Configuration Settings**.
The Dynamic Converter Configuration page is displayed (see "[Dynamic Converter Configuration Page](#)" on page A-2).
3. Enter a new value in the **Time Out** field (increasing it from 3 minutes, which is the default).
4. Click **Update** to enable your changes.

The changed setting takes effect immediately, so you do not need to restart the Content Server.

10.3 Embedded Graphics on UNIX

Some source documents contain embedded OLE objects. Embedded OLE objects are usually accompanied by a graphic "snapshot" in the form of a Windows metafile. On both Windows and UNIX, Dynamic Converter can use the metafile snapshot to convert the OLE object. When the metafile is not available, Dynamic Converter reverts to OLE technologies for the conversion. In that event, the conversion will still succeed on Windows, but it will fail on UNIX.

10.4 Use of Vector Versus Raster Graphics Formats

If you are converting vector graphics, Dynamic Converter requires access to a running X-Server. This is because Dynamic Converter depends on the X-Server system to draw the pixels.

Vector graphics formats describe lines and fills. Common formats are WMF, EMF, CorelDRAW, Adobe Illustrator, Excel charts, Word autosshapes, and PowerPoint presentations. Raster graphics, on the other hand, contain pixel information of an image. Common file formats are BMP, JPEG, and GIF.

One way to tell the difference between a vector and a raster graphic is to try to stretch the image. Since vector graphics describe lines, they will re-compute the placement of the lines and the image should still look nice. Raster graphics, however, will become pixelated when you resize.

See the *Dynamic Converter Installation Guide* for instructions on how to set up rendering of graphics and fonts in UNIX.

10.5 Converting Vector Graphics and Spreadsheet Text in UNIX

Dynamic Converter requires access to a running X-Server in UNIX in order to convert vector graphics and to properly measure text that spans multiple columns in spreadsheets.

See the *Dynamic Converter Installation Guide* for instructions on how to set up rendering of graphics and fonts in UNIX.

10.6 URL Rewriting

Dynamic Converter wraps the `dcUrl('url', reserved_type)` Idoc Script extension function around all hyperlinks and image source links (`src`). The default implementation of this script function is to do a simply pass-through, but external integration technologies (such as CIS) can modify this behavior by defining a filter plug-in for "dcUrlFilter."

Dynamic Converter evaluates the link URL, applies the "dcUrlFilter" filter if it exists, and then return the URL value. If the dcUrlFilter filter is not defined, then the original URL is unchanged. Links to internal bookmarks always remain unchanged.

Reserved Types

The `reserved_type` function argument is a number 1001, 1002, etc., which indicates where in the Dynamic Converter core engine the URL is being written. This value can be used to distinguish the type of URL. For example, gallery graphic, inter-document link, etc. The reserved type values and their meanings are as follows:

Value	Description
1001	Link (different split)
1002	Previous element (different split)
1003	Previous page (TOC frame)
1004	Previous page
1005	Next page (TOC frame)
1006	Next page
1007	Next element (different split)
1008	Previous page (TOC frame)
1009	Previous page
1010	Next page (TOC frame)
1011	Next page
1018	Image link
1019	Image link
1020	Image link
1021	Image link
1022	Background graphic (not from source)
1023	Background graphic (from source)

10.7 Relative URLs in Templates and Layout Files

Consider the following image tag: ``. In most implementations of Dynamic Converter, it is likely that the output files will end up in a different location than the template files. If the developer uses the template above in this scenario, the output files produced will have a reference to *image.gif*, which the browser will assume has the same path as the output files. The problem is that *image.gif* is likely to be back in the directory where the template file is located. This is a problem for anything referenced in the template using a relative URL. There are several possible solutions to this problem.

Solution 1: Ensure That the References Are Good

If the developer knows exactly which files all of the templates reference, the correct files (such as *image.gif*) can be moved to or located in the output directory or directories. This solution requires the developer to have exact knowledge of the contents of the templates, and may propagate the same set of files into many output locations.

Solution 2: Use Absolute URLs

The developer can design templates to contain absolute URLs to any referenced files. The template in the example would then look something like this.

```
<HTML>
<BODY>
<P><IMG SRC="http://www.company.com/templates/image.gif"></P>
{## INSERT ELEMENT=Sections.1.Body}
</BODY>
</HTML>
```

If `<${HTTPWEBROOT}>` is used instead, you eliminate the problem of output files tied to a specific domain.

Solution 3: Make Path Statements in a Separate File

The developer can create a separate Idoc Script file that states the path, for example:

```
<@dynamichtml Image_Dir@><${HttpWebRoot}>groups/public/documents/graphic/<@end@>
```

The developer can then load the Idoc resource and reference the path statement from the included Idoc Script file as follows:

```

```

All long as the graphics (or related files) are checked in with the security group and document type to match the stated path (in this example, a security group "Public" and a document type "Graphic"), then the paths will resolve, and the page will display properly.

10.8 Browser Caching

In the process of building and debugging templates, you are likely to run the same source file through Dynamic Converter repeatedly with slightly different templates. Depending on how you are naming the output files, this may have a tendency to produce the same set of file names repeatedly. In this scenario, especially if the output is being read directly from a file system and not through a web server, browsers will have the tendency to show the old cached results and not the new ones.

If it looks like bad output, click **Refresh** on every frame before deciding that it is a problem with the template or the software.

Tip: You may find it simpler to empty and turn off caching in your browser while creating and testing your templates.

10.9 Image Sizing Rules

There are a large number of factors that affect the size of the final exported image. The precedence of rules for how those factors work is as follows:

1. Any images that the template specifies with the `{## graphic}` macro are subtracted from the space available for graphics on that particular deck. In general, you should be wary of templates that require images on every deck as they will eat into the overall amount of room available for document graphics.
2. The `SCCOPT_EX_GRAPHICBUFFERSIZE` option, which is only used to reduce image size if necessary. It preserves the image aspect ratio.
3. The `SCCOPT_GRAPHIC_SIZELIMIT` option, which is only used to reduce image size if necessary. It preserves the image aspect ratio.
4. The `SCCOPT_GRAPHIC_WIDTHLIMIT` and `SCCOPT_GRAPHIC_HEIGHTLIMIT` options. These are only used to reduce image size if necessary. They preserve the image aspect ratio, even if both are specified.
5. `'width='` and `'height='` parameters in the `{## INSERT}` statement of the template. This reduces or enlarges the image to match the specified dimension(s). The image aspect ratio is changed if both are specified. The aspect ratio does not change if only one or none of these parameters is specified.
6. Original image dimensions based on the information in the source file and the DPI setting, if applicable.

10.10 CSS Considerations

The styles discussed in this section relate only to script templates (see [Chapter 7, "Script Templates"](#)). Styles in Classic HTML Conversion templates (see [Chapter 5, "HTML Conversion Templates"](#)) are handled differently.

One of the most powerful features of cascading style sheets (CSS) is the ability to override the styles suggested in various ways. Dynamic Converter has designed its CSS support to permit users to override the style sheets that it produces. This, in turn, enables the user to help blend documents from many authors into a collection that has a more unified look. In order to make this override work, one first needs to understand style names.

In addition, it should be remembered that the output from Dynamic Converter might be placed into many HTML files. Special attention must be paid to ensure that `<LINK REL=STYLESHEET HREF="{## INSERT ELEMENT=Pragma.cssFile}">` statements are placed in the appropriate locations.

10.11 Style Names Used by Dynamic Converter

Style names are taken from the original style names in the source document. There is an inherent limitation in the style names the CSS standard permits. The standard only permits the characters a-z, A-Z, 0-9, and dash (-). Source document style names do not

necessarily have this restriction. In fact, they may even contain Unicode characters at times. For this reason, the original style names may need to be modified to conform to this standard. To avoid illegal style names, Dynamic Converter performs the following substitutions on all source style names:

- If the character is "-", then it is replaced with "--."
- If the character is not one of the remaining characters (a-z, A-Z, or 0-9), then it is replaced by "-xxxx" where "xxxx" is the hexadecimal Unicode value of the character.
- If neither of the preceding situations is applicable, the character appears in the style name normally.

An example of one of the most common examples of this substitution is that spaces in style names are replaced with "-0020." For a more complete example of this character substitution in style names, consider the source style name "My Special H1-Style!" (with a space and an exclamation mark in its name). This would be transformed to "My-0020Special-0020H1--Style-0021."

While admittedly this system lacks a certain aesthetic, it avoids the problem of how the document looks when the browser gets duplicate or invalid style names. Developers should also appreciate the simplicity of the code needed to parse or create these style names.

In addition, Dynamic Converter creates special list versions of styles. These have the same name as the style they are based on with "--List" appended to the end. These styles differ from their original counterparts in that they contain no block-level CSS.

10.12 Overriding Dynamic Converter Styles

Once style names are understood, it is easy to override the CSS file produced by Dynamic Converter. Follow the CSS file link in the template with another link to the CSS override file. For more information on the link to Dynamic Converter's CSS file, see "[Pragma.CSSFile and {## LINK}](#)" on page 10-6. This override file should then contain styles with the same names as the ones used by Dynamic Converter's CSS file.

Remember that many file formats allow styles to be based on other previously defined styles. Dynamic Converter supports this by nesting styles. In this way each nested style inherits and may override items defined in the styles that surround it.

10.13 Pragma.CSSFile and {## LINK}

One `{## INSERT Element=Pragma.CSSFile}` statement should appear at the top of each HTML file produced when a CSS flavor of HTML is used. It should therefore be remembered that the `## LINK` statement may be used to trigger the creation of additional HTML files. As a result, each `## Linked` template will typically contain a `<Link>` tag to the CSS file generated.

Using a `## LINK` statement, it is possible, though, to link to a template that does not have any `{##}` statements that would need to reference the CSS file. In that case, the `<Link>` to the CSS file may safely be omitted. Consider, for example, a template that has only two `##` statements, both of which are `## links` (perhaps to put the results into two separate frames). This template file would not need a `<Link>` to the CSS file.

Regardless of how many HTML files are produced by Dynamic Converter, only one CSS file is generated. It is also worth repeating here that the `<Link>` to the CSS file must occur in the `<HEAD>` section of the document and each resulting HTML file may have only one `<HEAD>` section.

10.14 Well-Formed HTML

The output of Dynamic Converter has been tested to ensure that it is well-formed. This is meaningless, however, unless the template used by Dynamic Converter is also well-formed. To assist with creating well-formed templates, here is a list of common problems that may cause documents to not be well formed:

- All tags must be properly nested.
- All tags that are opened must also be closed. This includes tags that are not normally thought of as needing closing tags, including `<META>`, `<LINK>`, `<FRAME>`, `<HR>`, and `
`.
- Everything after an is-equal-to sign (=) must be in double quotes. Hence, `` is OK, but `` is not.
- In order for ` ` to appear in a document, a `<!DOCTYPE>` statement must be in the HTML code. Since Dynamic Converter cannot know if the template included the `<!DOCTYPE>` statement when the `SCCHTML_FLAG_STRICT_DTD` flag is set, ` ` is always used instead of ` `.
- Characters in the range 0x80 - 0xFF are to be written in the form `&#xxx;`.
- The only three characters `<0x20` allowed in any document are: `\t`, `\n`, and `\r`.
- All attributes of a tag must be followed by `"=value"`. Thus, the `NoWrap` in `<Table NoWrap>` is not well formed. Dynamic Converter uses `<Table NoWrap=NoWrap>` instead.

10.15 Positional Frames Support

Dynamic Converter 7.7 and higher uses DHTML to position objects. However, only two types of object positioning are supported: **paragraph anchored objects** and **page anchored objects**. Here are some important notes about this initial support for positional frames:

- Dynamic Converter generates paragraph objects separately from page objects even if it appears that they should be placed in the same location.
- Transparency is not supported when separate graphics items are placed on top of one another. The `SCCOPT_EX_PREVENTGRAPHICOVERLAP` option does not apply to these graphics. The graphics will appear relative to where the anchor point is, not relative to the text in the document. Additionally, Dynamic Converter does not support certain graphics effects, such as rotation or stretching.
- It is important to note that the `SCCOPT_EX_GRAPHICOUTPUTDPI` option must be set properly to achieve best results.
- In some cases, Dynamic Converter will produce output with inaccurately placed objects when the input document features positional frame objects. However, this end result is no worse than the end result when handling positional frame objects in pre-7.7 versions of Dynamic Converter (i.e., the graphics would be placed in a long column).
- This Element only works in the 4.0 flavors of HTML.

10.16 Template Writing Tips

Given the limited amount of space in each deck, it is important to maximize the amount of usable data in each deck produced by Dynamic Converter. Some ways to reduce the amount of space wasted in each deck include the following:

- Eliminate unnecessary whitespace characters in the template. While the presence of these characters makes reading, editing and maintaining the template easier, they also get written to each output deck "as is." When writing templates for devices with small deck sizes, it may prove worthwhile to remove the extra whitespace characters to increase the amount of usable data in each deck. Please note that the `SCCOPT_EX_COLLAPSEWHITESPACE` option does not affect white space coming from the template.
- Eliminate any extra links between decks. While good navigation is essential, redundant or unnecessary links eat into the amount of space left in each deck for data. In addition to the markup used for navigation, space is set-aside for the URL of the link, which is determined by the `SCCOPT_EX_MAXURLLENGTH` option. Currently, space is not reclaimed if URLs are shorter than this length. In addition, if URLs are longer than this length, deck overflow may happen.

User Interface

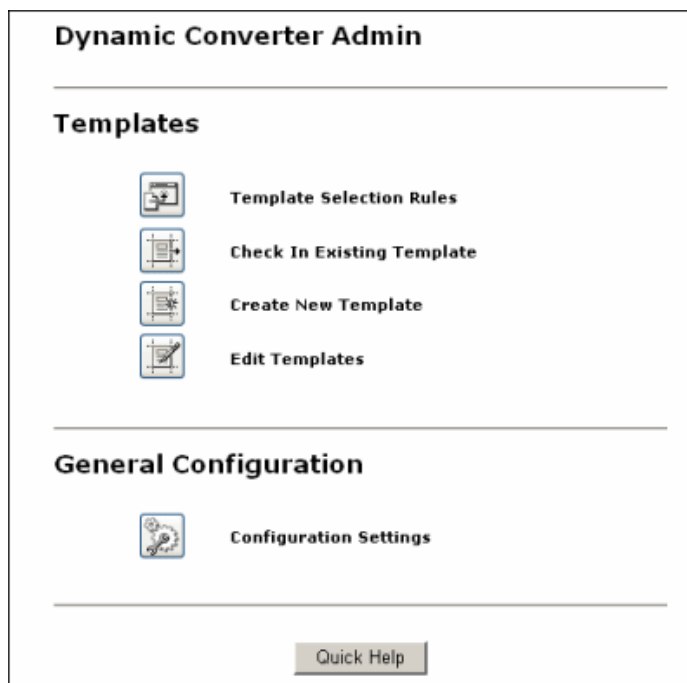
Dynamic Converter includes administrative pages that you use to create and administer your dynamic conversion environment:

- ["Dynamic Converter Admin Page"](#) on page A-1
- ["Dynamic Converter Configuration Page"](#) on page A-2
- ["Template Selection Rules Page"](#) on page A-9
- ["Template Check-In Form"](#) on page A-12
- ["Edit Templates Page"](#) on page A-14

A.1 Dynamic Converter Admin Page

If you click either of the Dynamic Converter Admin links (see ["Dynamic Converter Admin Link"](#) on page 1-9), the Dynamic Converter Admin page is displayed.

Figure A-1 Dynamic Converter Admin Page



You can do the following on the Dynamic Converter Admin page:

- Create and manage template selection rules (see [Chapter 3, "Template Rules"](#))
- Check in existing templates (see [Chapter 4, "Conversion Templates"](#))
- Create new templates (see [Chapter 5, "HTML Conversion Templates"](#))
- Edit existing templates (see [Chapter 5, "HTML Conversion Templates"](#))
- Configure Dynamic Converter settings (see [Chapter 2, "Configuring Dynamic Converter"](#))

A.2 Dynamic Converter Configuration Page

There are several configuration settings that determine how Dynamic Converter handles source documents. You can specify the default template to use for source documents, the file types to convert, the number of criteria fields available on the template selection rules page, and a number of other Dynamic Converter translation options.

You set these configuration options on the Dynamic Converter Configuration page, which you access by clicking **Configuration Settings** on the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).

This page is presented in several sections:

- "[General Conversion Settings](#)" on page A-3
- "[UNIX Configuration Settings](#)" on page A-5
- "[Classic HTML Template Conversion Configuration Settings](#)" on page A-6
- "[Script Template Conversion Configuration Settings](#)" on page A-7
- "[Conversion and Caching Optimizations](#)" on page A-7

When you are done, click the **Update** button at the bottom of the page to apply any configuration changes you made. Changes made on the Dynamic Converter Configuration page take effect immediately and do not require a restart of the Content Server.

A.2.1 General Conversion Settings

Figure A–2 General Settings on Dynamic Converter Configuration Page

Dynamic Converter Configuration

[Administration](#) --> [Dynamic Converter Admin](#) --> Dynamic Converter Configuration

Default Template
 Template that will be used if none of the other selection rules match.

Template Available Templates Template Types

Default Layout
 Layout that will be used if none of the other selection rules match.

Layout Available Layouts

Conversion Formats
 These are the formats that are eligible for dynamic conversion. Note that only these formats will display an (HTML) conversion link on the Search Result page.

Maximum File Size **Bytes**
 The maximum size of file that will be processed by Dynamic Converter.

Time Out **Minutes**
 Dynamic Conversions that take longer than this amount of time will fail.

Rule Criteria **Criteria per rule**
 The number of individual criteria that can be specified per template selection rule.

Rendition

The rendition of a content item to be converted.

The general settings section of the Dynamic Converter Configuration page enables you to set a number of conversion settings, such as the default template and layout, the supported conversion formats, etc.

The following options are available (for each page section):

Default Template

Option	Definition
Template	This is the name of the template that is applied to source documents that fall outside of your template selection rules. A default template can be especially useful when you are still setting up your template selection rules. You might, for example, create a blank, or stripped-down, template as the default.
Available Templates	This is a list of all the available templates currently stored in the Content Server.
Template Types	This is a list of the different types of templates: HTML Conversion Template, Classic HTML Conversion Template, and Script Template. When you choose a template type, a list of available templates of that type are displayed in the Available Templates field. For more information, see " Template Types " on page 4-1.

Default Layout

Option	Definition
Layout	This is the name of the layout template that is applied to source documents that fall outside of your template selection rules. This only applies to Classic HTML Conversion templates. You might create a default layout template that includes the Content Server borders and navigation.
Available Layouts	This is a list of all the available layout templates currently stored in the Content Server.

Conversion Format

Option	Definition
Formats	<p>These are the supported file formats that are converted by Dynamic Converter and as a result, will include an (HTML) link next to them. A file format is the same as a MIME type, and it can be specified using the same comma delimited values (application/rtf, application/msword, etc.). It is important to note that this Format setting serves a different purpose than the Format field on the Template Selection Rules page (see "Template Selection Rules Page" on page A-9), which assigns templates to source documents based on their file type.</p> <p>When Dynamic Converter is used with Inbound Refinery (or another conversion add-on), a list of file formats similar to the Available Templates list may be available. If Dynamic Converter is used as a stand-alone system, the list may be empty, and you will need to manually add them. For more information, see "Adding File Formats For Dynamic Conversion" on page 2-2.</p>

Other Settings

Option	Definition
Maximum File Size	<p>This is the maximum size of the source file that Dynamic Converter will process. The value must be entered in bytes.</p> <p>The default is 20,000,000 bytes (just over 19 MB).</p>
Time Out	<p>This is the amount of time that Dynamic Converter will spend processing a source document. If the conversion takes longer than this specified time, Dynamic Converter will quit the conversion and generate an error.</p> <p>The default is three minutes.</p>
Rule Criteria	<p>This is the number of individual criteria fields that is available per rule on the Template Selection Rules page.</p> <p>The default is two criteria per rule.</p>

Option	Definition
Rendition	<p>This is the source content item that is converted by Dynamic Converter. The options include: native (the source document), alternate file (the alternative file available for the source document), and web-viewable (the web-compatible version of a source document).</p> <p>The default choice is "native."</p> <p>Please note that the default treatment of primary files versus alternate files is slightly different in more recent versions of Dynamic Converter. Prior to version 6.0, the alternate file was used as long as it was found in the supported file formats list. Dynamic Converter now lets you specify the exact version of the source document to convert: native (primary), alternate, or web-viewable.</p>

A.2.2 UNIX Configuration Settings

Figure A-3 UNIX Settings on Dynamic Converter Configuration Page

Unix System Configuration
These settings apply to Unix-like systems.

DISPLAY
The DISPLAY environment variable tells the X-Windows application where to send its data.

Font Path
The colon delimited directory paths containing fonts. The directory should contain TrueType and FreeType fonts. **The Font Path value must reference a valid directory with these fonts or conversions may fail.**

The UNIX System Configuration section on the configuration page enables you to configure a number of UNIX-specific settings. These settings do not appear on the configuration page if you are running Dynamic Converter on a Windows system.

Option	Definition
DISPLAY	<p>Enter a value for the DISPLAY environment variable (for example, "10.133.91.193:0.0"). This value determines where display information is sent. This variable tells the X Windows application where to send its data.</p> <p>This setting only applies if the "Use X Windows for Rasterization" check box is selected.</p>
Font Path	<p>This value sets a directory path or paths which have TrueType or FreeType fonts. If this font path is empty or does not have valid fonts in it, your conversion will fail. Use colons to separate directory paths, for example:</p> <p><code>/usr/X11R6/lib/X11/fonts/truetype:/usr/share/fonts/default/TrueType:/usr/X11R6/lib/X11/fonts/TTF</code></p>

A.2.3 Classic HTML Template Conversion Configuration Settings

Figure A-4 Classic HTML Template Conversion Configuration Settings on Dynamic Converter Configuration Page

Classic HTML Template Conversion Configuration

These settings alter how conversions are rendered.

Use X-Windows for Rasterization

When checked, the X-Windows graphics device is used for rendering graphics and fonts during Classic HTML Template conversions. For Linux and Solaris, you can uncheck this box to use software rendering.

Use Services For Intradocument Hyperlinks

When checked, hyperlinks to targets within the same document use service calls instead of file locations in the weblayout conversion cache.

The Classic HTML Template Conversion Configuration section on the configuration page enables you to configure a number of settings related to the conversion of Classic HTML Conversion templates (for more information, see [Chapter 5, "HTML Conversion Templates"](#)).

Option	Definition
Use X Windows for Rasterization	<p>This option is displayed on UNIX systems only.</p> <p>If you select this check box, the X Windows graphics device is used for rendering graphics and fonts during template conversions. For Linux and Solaris, you can uncheck this box to use Dynamic Converter's internal software rendering mechanism.</p>
Use Services For Intradocument Hyperlinks	<p>If you select this check box, hyperlinks within a converted document are written as URLs with service calls (for example, using GET_DYNAMIC_CONVERSION) to refer to the targets rather than file locations in the Web Layout conversion cache. This can help prevent links from becoming broken if the referenced converted item (for example, a sheet within a PowerPoint presentation) is no longer available in the conversion cache (for example, because its maximum caching period expired and the cached file was deleted). The called service will regenerate the referenced converted item if it no longer exists in the cache.</p> <p>This option is selected by default.</p>

A.2.4 Script Template Conversion Configuration Settings

Figure A-5 Script Template Conversion Configuration Settings on Dynamic Converter Configuration Page

```

# HTML Export
# Sample Application configuration file
#
# Copyright (c) Stellent, Inc. 1996 - 2005
# All rights reserved.
#
#
# Some comments on the format of this file.
#
# 1) Options are set with lines of the form:
#
#    option value
#
# 2) Lines that begin with the '#' character are comments. The '#'
#    must appear at the beginning of the line. If it appears
#    anywhere else it is treated as normal text. Any text can appear
#    in a comment. Lines that set options MAY NOT also contain
#    comments.
#
#

```

The Script Template Conversion Configuration section on the configuration page enables you to directly access the global script template settings (for more information, see [Chapter 7, "Script Templates"](#)). Any changes you make by adding or commenting out parts of this file will override other conversion options. This file contains comments that explain each of the available options.

Text in this box is not verified for correct syntax.

A.2.5 Conversion and Caching Optimizations

Figure A-6 Conversion and Caching Optimizations on Dynamic Converter Configuration Page

Conversion and Caching Optimizations
These settings modify how Dynamic Converter convert and cache files and may significantly improve the overall performance of your intranet or external Web site.

Use Upfront and Forced Conversions
When checked, documents will be converted when indexed and multiple conversions may occur based on the Forced Conversion Rules found on the Template Selection Rules page.

Reevaluate conversion rules during re-indexing
When checked, re-evaluation of all upfront and forced conversions will occur during a re-index cycle. You might temporarily enable this feature to retranslate all relevant content items using a new conversion rule by enabling this option, rebuilding the content server index, and then disabling this option.

Reconvert when metadata is updated
This option permits reconversion of content items when their metadata is updated.

The Conversion and Caching Optimizations settings have been present in the *config.cfg* file, but are now exposed on the configuration page. The authoritative location for these settings is on the configuration page. If you had these set in a previous *config.cfg*, those values will be used to populate the values on the configuration page.

Option	Definition
Dated Cache Interval Days	<p>This option defines the frequency (in days) with which the conversion cache is evaluated and cached items may be considered for deletion, depending on how long they have been in the cache and their conversion status.</p> <p>The default is 7 days (i.e., the cache is evaluated once a week).</p> <p>This setting applies to a number of Content Server related products (not just Dynamic Converter).</p>
Conversion Cache Expiration Period	<p>This option defines the number of days that must pass before converted items in the cache may be considered for deletion. Date expiration only applies to cached items for documents that are no longer present and to cached items that were not generated by forced conversion (for more information, see "Forced Conversions" on page 1-4).</p> <p>The default is 7 days (i.e., cached items are not considered for deletion unless they are more than one week old).</p>
Max. Conversion Cache Size	<p>This option defines the number of megabytes of the file cache that is allowed for dynamic conversions. If it is exceeded, then during the next clean-up cycle (which, by default, is seven days) the caches that have not been accessed for the longest period of time are deleted first. (The list for deleting is sorted by the "last accessed" date in ascending order.) If the cache size limit is not exceeded, then the caches are examined for potential deletion in the same order, but caches that are forced conversions of existing documents are not deleted.</p> <p>The default is 10,000 (about 9.8 GB).</p>
Use Upfront and Forced Conversions	<p>Choose whether all upfront conversion and forced conversions should be enabled (for more information, see "Upfront Conversions" on page 1-3 and "Forced Conversions" on page 1-4).</p>
Reevaluate conversion rules during re-indexing	<p>Choose whether all upfront and forced conversions should be re-evaluated during a re-index cycle. You might temporarily enable this Element, for example, to re-translate all relevant content items using a new conversion rule that you created. To do so, enable this option, rebuild the Content Server index, and then disable this option again.</p>
Reconvert when metadata is updated	<p>Choose whether to convert items again if their metadata is updated. For more information, see "Metadata Changes" on page 1-5.</p>
Timestamp Check Frequency	<p>Specify the frequency with which Dynamic Converter checks the timestamp of converted content items. For more information, see "Timestamp Checking Frequency" on page 1-6.</p>
Use database method to determine if content items need reversion	<p>This option is available only if the "Reconvert when metadata is updated" option is enabled.</p> <p>It allows you to specify that the database method should be used to determine if a content item's metadata has been updated.</p> <p>For more information, see "Metadata Changes" on page 1-5.</p>

A.3 Template Selection Rules Page

Use the Template Selection Rules page to add, remove, and re-order rules. To access this page, click **Template Selection Rules** on the Dynamic Converter Admin page (see "Dynamic Converter Admin Page" on page A-1).

Figure A-7 Template Selection Rules Page

Template Selection Rules

Administration --> Dynamic Converter Admin --> Template Selection Rules

Template Selection Rules (In order of evaluation)

Test_Academy	Move Up
bjc	Move Down
bjc97.doc	
ANOTHER_DC_SAMPLE	Delete Rule
myrule	
Nav_demo	

New rule name:

Criteria for selected rule

Field	Value
Content ID	005428

Template and layout for selected rule

Template	Available Templates	Template Types
ACADEMY-GUI-TEMPLATE		HTML Conversion Template

Layout	Available Layouts

File Extension

File extension of the dynamic converted page.

Forced Conversion

Indicates that this rule is to be used for forced conversions. This rule will always be applied to content items as long as the content items match the rule criteria. The content can be retrieved using the incDynamicConversionByRule Idoc Script function or the GET_DYNAMIC_CONVERSION service with the conversionRule parameter specified.

Exclude From User Request

Use this option to prevent the rule to be used when a user clicks on the HTML Rendition link or menu item. Rules designed for fragments and used by the incDynamicConversion Idoc Script function should be excluded from Dynamic Converter's rule evaluation during a user request.

Rules are processed from the top down. When a user requests a source document in Content Server, the rule that appears first in this list is processed. If the rule does not apply (for example, the source document might contain metadata not specified in the rule), then the next rule is processed. This process continues until it reaches the last rule in your list.

Once a rule has been added, you can use the criteria section to define or target your rule toward the source documents stored in the Content Server. You might, for example, choose "Type" as a category and "Report" as the name so that your rule applies to all documents with the type "Report."

Lastly, you will choose a template for your rule. In a second template field, you can specify a layout template that complements a Classic HTML Conversion template by establishing a consistent page layout (borders, navigation, scripting, etc.) for your converted documents.

Note: Rules that were created in a Dynamic Converter version prior to 6.1 appear as a numbered rule in this version of Dynamic Converter. You can continue using that rule or delete it and re-create the rule in Dynamic Converter 11gR1 (you cannot rename a rule).

Page Features

The following options are available (for each page section):

Template Selection Rules

Option	Definitions
Move Up	Click this button to move a rule up the list, giving it precedence over the rules below it. Template selection rules are processed from the top-down.
Move Down	Click this button to move a rule down the list, giving it less importance than the rules above it. Template selection rules are processed from the top-down.
Delete Rule	Click this button to remove a rule from the list. When you remove a rule, you are removing the customized settings for that rule (metadata criteria and template).
Add New Rule	Click this button to add a new rule to the Template Selection Rules list. Once you add a rule, you can move it up or down in the list, changing the order that it is processed by Dynamic Converter.

Criteria for Selected Rule

Option	Definitions
Field	<p>These are the metadata fields that you can base a template selection rule on (all fields are case-sensitive). The number of criteria that you can specify for each rule is controlled by a setting on the Dynamic Converter Configuration page (see "Dynamic Converter Configuration Page" on page A-2).</p> <p>The following metadata fields are available:</p> <ul style="list-style-type: none"> ■ Content ID: The unique identifier of a content item. The content ID can be assigned by a user or automatically generated by Content Server. Your template, as a result, is assigned to only one content item. ■ Title: The descriptive name assigned to a content item by the user. Your template, as a result, is assigned to only one content item. ■ Author: The person who created or revised the content item. Your template is assigned to all content items created by this author. ■ Type: The category of content items (a category is created in the Configuration Manager in Content Server). Your template is assigned to all content items matching this category. ■ Security Group: The set of files with the same access privileges, generally 'public' and 'secure.' Your template is assigned to all content items matching this security group. ■ Template Type: Facilitates searches for templates in the Content Server. ■ Format: The file format for a content item (this is determined by the application used to create the file). A file format is the same as a MIME type, and it can be specified using the same comma delimited values (application/rtf, application/msword, etc.). Please note that the Conversion Formats field on the Dynamic Converter Configuration page (see "Dynamic Converter Configuration Page" on page A-2) serves a different purpose, and that is to control which file formats will actually be converted by Dynamic Converter. ■ User Agent: The viewing device that requests and downloads content from a web server. Typically, this is a web browser that runs on a computer, such as Microsoft Internet Explorer, Mozilla Firefox, or Opera. You can target a particular web browser by entering its value in this text box (such as *msie 7* for Internet Explorer 7, *firefox 2* for Mozilla Firefox 2, *webtv 1* for WebTV viewers, and so on). This setting is particularly useful for targeting your content to mobile devices. <p>If you have added custom metadata to the Content Server, those values will appear in this list too.</p>
Value	<p>This is the specific metadata value for your criteria. Source documents are converted with the associated template if their metadata value matches the value listed here.</p> <p>You can select the desired metadata from the menu to the right of the Value field. You can also use wildcards in the Value field (for example <i>*report*</i>). An * (asterisk) wildcard represents any number of characters, and a ? (question mark) represents a single character. For example, the value <i>report*</i> includes <i>report2001</i>, <i>reporting</i>, and <i>reports</i>. The value <i>report?</i> includes <i>reports</i> and <i>report8</i>, but it does not include <i>report10</i>.</p>

Template and Layout for Selected Rule

Option	Definitions
Template	This is the name (content ID) of the template that you want to apply to source documents matching the above criteria.
Available Templates	This is a list of all the available templates currently stored in the Content Server.
Template Types	This is a list of the different types of templates: HTML Conversion Template, Classic HTML Conversion Template, and Script Template. When you choose a template type, a list of available templates of that type will display in the Available Templates field. For more information, see Chapter 4, "Conversion Templates."
Edit Template	This button is activated once you enter a recognized template in the Template text box. Click this button to open the Template Editor. (The Template Editor contains a suite of editing options, all in a graphical user interface.) The first time you click this button, you are prompted to download the Template Editor. For more information, see " Classic HTML Conversion Template Editor " on page 5-9.
Layout	This is the name of the layout template. This only applies for the Classic HTML Conversion templates. A layout template is commonly used along with another template to control the placement of items on a web page, in particular, the areas outside of the converted content (borders, navigation, company logo, custom script, etc.).
Available Layouts	This is a list of all the available layout templates currently stored in Content Server.

Other Settings

Option	Definitions
File Extension	Set the file extension of converted pages that use this rule.
Forced Conversion	Indicates that this rule is to be used for forced conversion (see " Forced Conversions " on page 1-4). This rule will always be applied to content items as long as the content items match the rule criteria. The conversion results can be retrieved using the incDynamicConversionByRule Idoc Script function or the GET_DYNAMIC_CONVERSION service with the conversionRule parameter specified.
Exclude From User Request	Indicates that the rule should not be used when a user clicks on the HTML rendition link or menu item. Rules designed for fragments (see " Fragment-Only Conversions " on page 1-4) and used by the incDynamicConversionByRule Idoc Script function should be excluded from Dynamic Converter's rule evaluation during a user request.
Update	Click this button to apply any changes that you have made to the Template Selection Rules page.
Quick Help	Click this button to display context-sensitive help information about this page.

A.4 Template Check-In Form

Use this page to check in an existing Dynamic Converter template file. To access this page, click **Check In Existing Template** on the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).

Figure A-8 Dynamic Converter Template Check-In Form

Dynamic Converter Template Checkin Form

[Administration](#) --> [Dynamic Converter Admin](#) --> Dynamic Converter Template Checkin Form

* Content ID	<input type="text"/>
* Type	ADACCT - Acme Accounting Department <input type="button" value="v"/>
* Title	<input type="text"/>
* Filer	sysadmin <input type="text"/> sysadmin <input type="button" value="v"/>
* Security Group	Public <input type="button" value="v"/>

* Primary File	<input type="text"/> <input type="button" value="Browse..."/>
----------------	---

* Revision	<input type="text" value="1"/>
Folder	<input type="text"/> <input type="button" value="Browse..."/>
Hidden	<input type="text"/> <input type="button" value="v"/>
Comments	<input type="text"/>
User Access List	<input type="text"/>
Trash Delete Old Name	<input type="text"/>
Trash Delete Location	<input type="text"/> <input type="button" value="Browse..."/>
Trash Delete Date	<input type="text"/> <input type="button" value="Calendar"/>
Trash Deleter	<input type="text"/>
Force Folder Security	<input type="text"/> <input type="button" value="v"/>
Inhibit Propagation	<input type="text"/> <input type="button" value="v"/>
Read Only	<input type="text"/> <input type="button" value="v"/>
Group Access List	<input type="text"/>
Template Type	Classic HTML Conversion Template <input type="button" value="v"/>
	<input type="checkbox"/> Is Correspondence
Author Or Originator	sysadmin <input type="text"/>
Addressee(s)	<input type="text"/>

This page is very similar to a typical Content Server check-in form. The main difference is the option to choose a template type. It is very important to select the appropriate template type (for more information, see ["Template Types"](#) on page 4-1) so that your Dynamic Converter menus and the Template Editor function properly.

A.5 Edit Templates Page

Use this page to edit an existing template (that is, one that is already checked into the Content Server). To access this page, click **Edit Existing Template** on the Dynamic Converter Admin page (see "[Dynamic Converter Admin Page](#)" on page A-1).

Figure A-9 *Edit Templates Page*

Dynamic Converter - Edit Templates
Administration --> Dynamic Converter Admin --> Dynamic Converter - Edit Templates

Template to Edit
Select the template that you would like to edit.

Template Available Templates Template Types
HTML Conversion Template

Edit Template

Quick Help

You can either type the content ID of an existing template or, much easier, you can select from the list of templates in the Content Server.

If a known template is not included in the list of available templates, then it was most likely not assigned the correct Template type when it was checked into the Content Server (for more information, see "[Checking In a Template](#)" on page 4-2). You then need to open the content information page of the checked-in template and update its template type.

After you specify a template, the **Edit Template** button becomes available. Click this button to open the Template Editor (for more information, see "[About Templates](#)" on page 5-1). The HTML Conversion Editor will be downloaded to your machine.

With some browsers, such as Firefox, you may be prompted for how to handle the file **dc_hcmapedit.jnlp**. The correct way to open this file is with Java(TM) Web Start Launcher (default).

Note: The Template Editor comes with its own extensive help system, which can be called from the application's user interface.

Conversion Filters

Dynamic Converter uses conversion filters to convert input files:

- ["Application Filters"](#) on page B-1
- ["Graphics Filters"](#) on page B-5

B.1 Application Filters

Dynamic Converter uses the following filters to convert application files (in alphabetical order):

Filter Name	Filter Description
ACD2	AutoCad 2004 / 2005 / 2006 (text only)
ACS	Microsoft Access 1.0, Microsoft Access 2.0
AMI	Ami Pro, Ami, Professional Write Plus
BDR	Microsoft Office Binder 7.0, Microsoft Office Binder 97 (<i>conversion of files contained in the Binder file is supported only on Windows</i>)
DBS	DBase III, DBase IV, DBase V
DEZ	DataEase 4.x
DIF	Navy DIF
DRW	Micrografx Drawing Products
DX	DEC DX 3.0 and DEC DX 3.1
EMF	Enhanced Windows Metafile
EN4	Enable Word Processor 4.x
ENS	Enable Spreadsheet
ENW	Enable Word Processor 3.0
EXE2	DOS Executable, Windows Executable or DLL
FAX	CCITT Group 3 Fax
FCD	First Choice DB
FCS	First Choice SS
FFT	IBM DCA/FFT
FLW	Freelance 1.0 & 2.0 for OS/2, Freelance 1.0 & 2.0 for Windows, Freelance 96 for Windows 95, Freelance 97 for Windows 95, Freelance for SmartSuite Millennium Edition, Freelance for SmartSuite Millennium Edition 9.6

Filter Name	Filter Description
FWK	Framework III
GDSF	Interface for *.FLT filters (see "Graphics Filters" on page B-5)
GIF	CompuServe GIF
GZIP	UNIX GZip
HGS	Harvard Graphics DOS 3.0 Chart, Harvard Graphics DOS 2.0 Chart, Harvard Graphics DOS 3.0 Presentation
HTML	Internet HyperText Markup Language (up to 3.0 with some limitations)
HWP	Hangul 97
HWP2	Hangul 2002
ICH	Ichitaro versions 8.x through 13.x and 2004
ICH6	Ichitaro versions 4.x through 6.x
IWP	Wang IWP
JBG2	JBIG2 graphic embeddings in PDF files
JW	JustWrite 1.0, JustWrite 2.0, Q&A Write 3
LEG	Legacy, Wordstar for Windows
LWP	<i>For Win32 platforms only.</i> Lotus WordPro 96, Lotus WordPro 97, Lotus WordPro for SmartSuite Millennium Edition, Lotus WordPro for SmartSuite Millennium Edition 9.6
LWP7	<i>For non-Win32 platforms only, and only supporting text extraction/viewing.</i> Lotus WordPro 97, Lotus WordPro for SmartSuite for the Millennium, Lotus WordPro for SmartSuite Millennium Edition 9.6
LZH	LZH Compress, LZA Self Extracting Compress
M11	Mass 11
MANU	Lotus Manuscript 1.0, Lotus Manuscript 2.0
MCW	MacWrite II
MIF	FrameMaker MIF versions 3.0, 4.0, 5.0, 5.5 and 6.0 and Japanese 3.0, 4.0, 5.0 and 6.0 (text only)
MIME	MIME-encoded mail messages (See "E-Mail Formats" on page C-7 for detailed information about MIME support.)
MM	MultiMate 3.6, MultiMate Advantage 2
MM4	MultiMate 4.0
MMFN	MultiMate Note
MP	Multiplan 4
MPP	Microsoft Project versions 98 through 2003 (text only)
MSG	Microsoft Outlook Message and Microsoft Outlook Form Template versions 97, 98, 2000, 2002 and 2003
MSW	Microsoft Word 4.x, Microsoft Word 5.x, Microsoft Word 6.x, Windows Write
MWKD	Mac Works 2.0 Database
MWKS	Mac Works 2.0 Spreadsheet
MWP2	Mac WordPerfect 2.0, Mac WordPerfect 3.0

Filter Name	Filter Description
MWPF	Mac WordPerfect 1.x
MWRK	Mac Works 2.0 WP
OW	OfficeWriter
PCL	PC File 5.0 Doc
PCX	Paintbrush, DCX (multi-page PCX)
PDX	Paradox 2 & 3, Paradox 3.5, Paradox 4, Paradox for Windows
PFS	PFS: Write A, PFS: Write B, Professional Write 1, Professional Write 2, IBM Writing Assistant, First Choice word processor, First Choice 3 word processor
PGL	HP Graphics Language
PIC	Lotus PIC
PICT	Macintosh PICT, Macintosh PICT2
PNTG	MacPaint
PP12	PowerPoint 2007
PP2	Microsoft PowerPoint 3.0 for Windows, PowerPoint 4.0 for Windows, PowerPoint 4.0 for the Mac
PP7	Microsoft PowerPoint 7.0 for Windows 95
PP97	Includes Presentation (PPT) and Slideshow (PPS) support. Microsoft PowerPoint 97, Microsoft PowerPoint Dual 95/97, PowerPoint 98 for the Mac, PowerPoint 2000, PowerPoint 2001 for the Mac, PowerPoint 2002 (XP), PowerPoint 2003, PowerPoint 2004 for the Mac, and PowerPoint v.X for the Mac
PPL	PFS: Plan
PSP6	<i>For Windows platforms only.</i> Paint Shop Pro 5.0 and 6.0
PST	Microsoft Outlook Folder and Microsoft Outlook Offline Folder files versions 97, 98, 2000, 2002 and 2003
PSTF	PST filter support
QA	Q&A Write
QAD	Q&A Database
QP6	Quattro Pro 5.0 - 8.0
QP9	Quattro Pro 9.0 - 12.0 (text only)
RAS	Sun Raster
RBS	R:Base System V, R:Base 5000
RFT	IBM DCA/RFT
RFX	Reflex
RTF	Rich Text Format
SAM	Samna
SC5	SuperCalc 5
SDW	Ami Draw
SHW3	Novell Presentations 3.0, Novell Presentations 7.0, Corel Presentations 8.0 - 12.0, WordPerfect Presentations

Filter Name	Filter Description
SMD	Smart DataBase
SMS	Smart Spreadsheet
SMT	SmartWare II
SNAP	Lotus Snapshot
SO6	StarOffice 6.x through 8.x, and OpenOffice 1.1 and 2.0 (Writer is fully supported, Draw and Calc are text only)
SOC	StarOffice Calc 5.2 (text only)
SOI	StarOffice Impress 5.2 (text only)
SOI6	StarOffice Impress 6.x, 7.x and 8.x and Open Office 1.1 and 2.0
SOW	StarOffice Writer 5.2 (text only)
SPT	Sprint
SWF	Macromedia Flash 6.x, Macromedia Flash 7.x, and Macromedia Flash Lite (text only)
TAZ	UNIX compress, UNIX tar
TEXT	Text - DOS character set, Text - ANSI character set, Text - Macintosh character set, Text - Unicode character set, Text - UTF-8, Text - EBCDIC.
TGA	Truevision TGA (TARGA)
TIF6	Tagged Image File Format, EPS (TIFF header only), CCITT Group 3 Fax, CCITT Group 4 Fax, JPEG, JFIF (JPEG not in TIFF format)
TW	Total Word
TXT	IBM DisplayWrite 2 or 3, IBM DisplayWrite 4, IBM DisplayWrite 5
VCRD	vCard, vCalendar
VISO	Visio 4 - Page Preview mode only (WMF/EMF), Visio 5, 2000, 2002 and 2003
VW3	Volkswriter
W12	Microsoft Word 2007
W6	Microsoft Word 6.0 for Windows, Microsoft Word 7.0 for Windows 95, Microsoft WordPad
W97	Microsoft Word 97, Word 98 for the Mac, Word 98-J, Word 2000, Word 2001 for the Mac, Word 2002 (XP), Word 2003, Word 2004 for the Mac, and Word v.X for the Mac
WG2	Lotus 1-2-3 for OS/2 release 2
WK4	Lotus 1-2-3 3.0, Lotus 1-2-3 4.0, Lotus 1-2-3 5.0
WK6	Lotus 1-2-3 for SmartSuite 97, Lotus 1-2-3 for SmartSuite Millennium Edition, Lotus 1-2-3 for SmartSuite Millennium Edition 9.6
WKS	Lotus 1-2-3 1.0, Lotus 1-2-3 2.0, Symphony, Microsoft Works SS, Microsoft Works DB, VP-Planner, Mosaic Twin, Quattro (DOS), Quattro Pro (DOS), Generic WKS, Windows Works Spreadsheet, Windows Works Database
WM	WordMarc
WMF	Windows Metafile
WORD	Word for Windows 1.x, Word for Windows 2.0, Word for Macintosh 4.0, Word for Macintosh 5.0

Filter Name	Filter Description
WORK	Microsoft Works DOS 1.0 WP, Microsoft Works DOS 2.0 WP, Microsoft Works Win 3.0 WP, Microsoft Works Win 4.0 WP
WP5	WordPerfect 5.x
WP6	WordPerfect 6.0 - 12.0
WPF	WordPerfect 4.2
WPG	WordPerfect Graphic 1.0
WPG2	WordPerfect Graphic 2.0
WPL	Dec WPS Plus 4.1
WPW	Novell PerfectWorks 2.0 word processor, Novell PerfectWorks 2.0 draw, Novell PerfectWorks 2.0 spreadsheet
WS	Wordstar 3.0, Wordstar 4.0, Wordstar 5.0, Wordstar 6.0, Wordstar 7.0
WS2	Wordstar 2000
XL12	Microsoft Excel 2007
XL5	Microsoft Excel 2.x, Excel 3.0, Excel 4.0, Excel 5.0, Excel 7.0, Excel 97, Excel 98 for the Mac, Excel 2000, Excel 2001 for the Mac, Excel 2002 (XP), Excel 2003, Excel 2004 for the Mac, v.X for the Mac, Excel 2.x Chart, Excel 3.0 Chart, Excel 4.0 Chart, Excel 5.0 Chart, Excel 7.0 Chart
XML	XML (text only)
XY	XyWrite / Nota Bene, Signature
YIM	Yahoo! Instant Messenger 6.x and 7.x
ZIP	PKZIP format, self-extracting executable files

B.2 Graphics Filters

Dynamic Converter uses the following filters to convert graphics files (in alphabetical order):

Filter Name	Filter Description
ACAD	AutoCAD Drawing Versions 2.5 - 2.6, 9.0 - 14.0, 2000i and 2002
BMP	Windows Bitmap, Windows Bitmap 98/2000, OS/2 Bitmap, OS/2 Warp Bitmap, Windows Cursor, Windows Icon, Corel Draw 2.0 - 11.0
CGM	Computer Graphics Metafile
ESHR	Escher internal Microsoft Office graphics format
IBFPX2.FLT	Kodak Flash Pix
IBGP42.FLT	CALS Raster
IBJPG2.FLT	Progressive JPEG
IBPCD2.FLT	Kodak Photo CD
IBPSD2.FLT	Adobe Photoshop (all versions)
IBXBM2.FLT	X-Windows Bitmap
IBXPM2.FLT	X-Windows Pixmap
IBXWD2.FLT	X-Windows Dump

Filter Name	Filter Description
IMCDR2.FLT	Corel Draw Versions 3, 4, 5, 6, 7, 8
IMCD32.FLT	
IMCD42.FLT	
IMCD52.FLT	
IMCD62.FLT	
IMCD72.FLT	
IMCD82.FLT	
IMCMX2.FLT	Corel Draw Clipart
IMCM52.FLT	
IMCM72.FLT	
IMDSF2.FLT	Micrografx Designer Version 6
IMFMV2.FLT	FrameMaker Vector and Raster Graphics (FMV)
IMG	GEM Image (Bitmap)
IMGDF2.FLT	IBM Graphics Data Format (GDF)
IMGEM2.FLT	Gem File (Vector)
IMIGS2.FLT	IGES Drawing
IMMET2.FLT	OS/2 PM Metafile
IMPIF2.FLT	IBM Picture Interchange Format
IMPS_2.FLT	Postscript (Levels 1-2) and EPS files
IMPSZ2.FLT	
IMPSI2.FLT	
IMRND2.FLT	AutoShade Rendering
IPHGW2.FLT	Harvard Graphics for Windows
PBM	PBM (Portable Bitmap), PGM (Portable Graymap), PPM (Portable Pixmap)
PDF PDFI	PDF versions 1.0 through 1.6 (including Japanese PDF) and Adobe Illustrator versions 7.0 and 9.0
PNG	Portable Network Graphics

Input File Formats

Dynamic Converter can convert a large number of input file formats:

- [Word Processing Formats](#)
- [Desktop Publishing Formats](#)
- [Database Formats](#)
- [Spreadsheet Formats](#)
- [Presentation Formats](#)
- [Graphic Formats](#)
- [Compressed Formats](#)
- [E-Mail Formats](#)
- [Other Formats](#)

C.1 Word Processing Formats

File Format	Comments
ANSI Text	7 & 8 bit
ASCII Text	7 & 8 bit
DEC WPS Plus (DX)	Versions through 3.1
DEC WPS Plus (WPL)	Versions through 4.1
DisplayWrite 2 & 3 (TXT)	All versions
DisplayWrite 4 & 5	Versions through 2.0
EBCDIC	All versions
Enable	Versions 3.0, 4.0 and 4.5
First Choice	Versions through 3.0
Framework	Version 3.0
Hangul	Versions 97 and 2002
IBM FFT	All versions
IBM Revisable Form Text	All versions
IBM Writing Assistant	Version 1.01
Just System Ichitaro	Versions 4.x through 6.x, 8.x through 13.x and 2004

File Format	Comments
JustWrite	Versions through 3.0
Legacy	Versions through 1.1
Lotus AMI/AMI Professional	Versions through 3.1
Lotus Manuscript	Version 2.0
Lotus Word Pro (non-Windows)	Versions SmartSuite 97, Millennium, and Millennium 9.6 (text only)
Lotus Word Pro (Windows)	Versions SmartSuite 96, 97 and Millennium and Millennium 9.6
MacWrite II	Version 1.1
MASS11	Versions through 8.0
Microsoft Rich Text Format (RTF)	All versions
Microsoft Word (DOS)	Versions through 6.0
Microsoft Word (Mac)	Versions 4.0 - 2004
Microsoft Word (Windows)	Versions through 2007
Microsoft WordPad	All versions
Microsoft Works (DOS)	Versions through 2.0
Microsoft Works (Mac)	Versions through 2.0
Microsoft Works (Windows)	Versions through 4.0
Microsoft Windows Write	Versions through 3.0
MultiMate	Versions through 4.0
Navy DIF	All versions
Nota Bene	Version 3.0
Novell Perfect Works	Version 2.0
Novell/Corel WordPerfect (DOS)	Versions through 6.1
Novell/Corel WordPerfect (Mac)	Versions 1.02 through 3.0
Novell/Corel WordPerfect (Windows)	Versions through 12.0
Office Writer	Versions 4.0 - 6.0
OpenOffice Writer (Windows and UNIX)	OpenOffice version 1.1 and 2.0
PC-File Letter	Versions through 5.0
PC-File+ Letter	Versions through 3.0
PFS:Write	Versions A, B and C
Professional Write (DOS)	Versions through 2.1
Professional Write Plus (Windows)	Version 1.0
Q&A (DOS)	Version 2.0
Q&A Write (Windows)	Version 3.0
Samna Word	Versions through Samna Word IV+
Signature	Version 1.0
SmartWare II	Version 1.02
Sprint	Versions through 1.0

File Format	Comments
StarOffice Writer	Version 5.2 (text only) and 6.x through 8.x
Total Word	Version 1.2
Unicode Text	All versions
UTF-8	All versions
Volkswriter 3 & 4	Versions through 1.0
Wang PC (IWP)	Versions through 2.6
WordMARC	Versions through Composer Plus
WordStar (DOS)	Versions through 7.0
WordStar (Windows)	Version 1.0
WordStar 2000 (DOS)	Versions through 3.0
XyWrite	Versions through III Plus

C.2 Desktop Publishing Formats

File Format	Comments
Adobe FrameMaker (MIF)	Versions 3.0, 4.0, 5.0, 5.5 and 6.0 and Japanese 3.0, 4.0, 5.0 and 6.0 (text only)

C.3 Database Formats

File Format	Comments
Access	Versions through 2.0
dBASE	Versions through 5.0
DataEase	Version 4.x
dBXL	Version 1.3
Enable	Versions 3.0, 4.0 and 4.5
First Choice	Versions through 3.0
FoxBase	Version 2.1
Framework	Version 3.0
Microsoft Works (Windows)	Versions through 4.0
Microsoft Works (DOS)	Versions through 2.0
Microsoft Works (Mac)	Versions through 2.0
Paradox (DOS)	Versions through 4.0
Paradox (Windows)	Versions through 1.0
Personal R:BASE	Version 1.0
R:BASE 5000	Versions through 3.1
R:BASE System V	Version 1.0
Reflex	Version 2.0
Q & A	Versions through 2.0

File Format	Comments
SmartWare II	Version 1.02

C.4 Spreadsheet Formats

File Format	Comments
Enable	Versions 3.0, 4.0 and 4.5
First Choice	Versions through 3.0
Framework	Version 3.0
Lotus 1-2-3 (DOS & Windows)	Versions through 5.0
Lotus 1-2-3 (OS/2)	Versions through 2.0
Lotus 1-2-3 Charts (DOS & Windows)	Versions through 5.0
Lotus 1-2-3 for SmartSuite	Versions 97 - Millennium 9.6
Lotus Symphony	Versions 1.0, 1.1 and 2.0
Mac Works	Version 2.0
Microsoft Excel Charts	Versions 2.x - 7.0
Microsoft Excel (Mac)	Versions 3.0 – 4.0, 98, 2001, 2002, 2004, and v.X
Microsoft Excel (Windows)	Versions 2.2 through 2007
Microsoft Multiplan	Version 4.0
Microsoft Works (Windows)	Versions through 4.0
Microsoft Works (DOS)	Versions through 2.0
Microsoft Works (Mac)	Versions through 2.0
Mosaic Twin	Version 2.5
Novell Perfect Works	Version 2.0
PFS:Professional Plan	Version 1.0
Quattro Pro (DOS)	Versions through 5.0 (text only)
Quattro Pro (Windows)	Versions through 12.0 (text only)
SmartWare II	Version 1.02
StarOffice/OpenOffice Calc (Windows and UNIX)	StarOffice versions 5.2 through 8.x and OpenOffice version 1.1 and 2.0 (text only)
SuperCalc 5	Version 4.0
VP Planner 3D	Version 1.0

C.5 Presentation Formats

File Format	Comments
Corel/Novell Presentations	Versions through 12.0
Harvard Graphics (DOS)	Versions 2.x & 3.x
Harvard Graphics (Windows)	Windows versions
Freelance (Windows)	Versions through Millennium 9.6

File Format	Comments
Freelance (OS/2)	Versions through 2.0
Microsoft PowerPoint (Windows)	Versions 3.0 through 2007
Microsoft PowerPoint (Mac)	Versions 4.0 through v.X
StarOffice/OpenOffice Impress (Windows and UNIX)	StarOffice versions 5.2 (text only) and 6.x through 8.x (full support) and OpenOffice version 1.1 and 2.0 (text only)

C.6 Graphic Formats

File Format	Comments
Adobe Photoshop (PSD)	All versions
Adobe Illustrator	Versions 7.0 and 9.0
Adobe FrameMaker graphics (FMV)	Vector/raster through 5.0
Adobe Acrobat (PDF)	Versions 1.0, 2.1, 3.0, 4.0, 5.0, 6.0 and 7.0 (including Japanese PDF)
Ami Draw (SDW)	Ami Draw
AutoCAD Interchange and Native Drawing formats (DXF and DWG)	AutoCAD Drawing Versions 2.5 - 2.6, 9.0 - 14.0, 2000i and 2002
AutoShade Rendering (RND)	Version 2.0
Binary Group 3 Fax	All versions
Bitmap (BMP, RLE, ICO, CUR, OS/2 DIB & WARP)	All versions
CALS Raster (GP4)	Type I and Type II
Corel Clipart format (CMX)	Versions 5 through 6
Corel Draw (CDR)	Versions 3.x - 8.x
Corel Draw (CDR with TIFF header)	Versions 2.x - 11.0
Computer Graphics Metafile (CGM)	ANSI, CALS NIST version 3.0
Encapsulated PostScript (EPS)	TIFF header only
GEM Paint (IMG)	All versions
Graphics Environment Mgr (GEM)	Bitmap & vector
Graphics Interchange Format (GIF)	All versions
Hewlett Packard Graphics Language (HPGL)	Version 2
IBM Graphics Data Format (GDF)	Version 1.0
IBM Picture Interchange Format (PIF)	Version 1.0
Initial Graphics Exchange Spec (IGES)	Version 5.1
JBIG2	JBIG2 graphic embeddings in PDF files
JFIF (JPEG not in TIFF format)	All versions
JPEG (including EXIF)	All versions
Kodak Flash Pix (FPX)	All versions

File Format	Comments
Kodak Photo CD (PCD)	Version 1.0
Lotus PIC	All versions
Lotus Snapshot	All versions
Macintosh PICT1 & PICT2	Bitmap only
MacPaint (PNTG)	All versions
Micrografx Draw (DRW)	Versions through 4.0
Micrografx Designer (DRW)	Versions through 3.1
Micrografx Designer (DSF)	Windows 95, version 6.0
Novell PerfectWorks (Draw)	Version 2.0
OS/2 PM Metafile (MET)	Version 3.0
Paint Shop Pro 6 (PSP)	Windows only, versions 5.0 - 6.0
PC Paintbrush (PCX and DCX)	All versions
Portable Bitmap (PBM)	All versions
Portable Graymap (PGM)	No specific version
Portable Network Graphics (PNG)	Version 1.0
Portable Pixmap (PPM)	No specific version
Postscript (PS)	Levels 1-2
Progressive JPEG	No specific version
Sun Raster (SRS)	No specific version
StarOffice/OpenOffice Draw for Windows and UNIX	StarOffice versions 5.2 through 8.x and OpenOffice version 1.1 and 2.0 (text only)
TIFF	Versions through 6
TIFF CCITT Group 3 & 4	Versions through 6
Truevision TGA (TARGA)	Version 2
Visio (preview)	Version 4
Visio	Versions 5, 2000, 2002 and 2003
WBMP	No specific version
Windows Enhanced Metafile (EMF)	No specific version
Windows Metafile (WMF)	No specific version
WordPerfect Graphics (WPG & WPG2)	Versions through 2.0
X-Windows Bitmap (XBM)	x10 compatible
X-Windows Dump (XWD)	x10 compatible
X-Windows Pixmap (XPM)	x10 compatible

C.7 Compressed Formats

File Format	Comments
GZIP	

File Format	Comments
LZA Self Extracting Compress	
LZH Compress	
Microsoft Binder	Versions 7.0-97 (<i>conversion of files contained in the Binder file is supported only on Windows</i>)
UUEncode	
UNIX Compress	
UNIX TAR	
ZIP	PKWARE versions through 2.04g

C.8 E-Mail Formats

File Format	Comments
Microsoft Outlook Folder (PST)	Microsoft Outlook Folder and Microsoft Outlook Offline Folder files versions 97, 98, 2000, 2002 and 2003
Microsoft Outlook Message (MSG)	Microsoft Outlook Message and Microsoft Outlook Form Template versions 97, 98, 2000, 2002 and 2003
MIME	MIME-encoded mail messages. (See below for detailed information about MIME support.)

MIME Support Notes

Here is detailed information about support for MIME-encoded mail message formats.

- MIME formats, including:
 - EML
 - MHT (Web Archive)
 - NWS (Newsgroup single-part and multi-part)
 - Simple Text Mail (defined in RFC 2822)
- TNEF Format
- MIME encodings, including:
 - base64 (defined in RFC 1521)
 - binary (defined in RFC 1521)
 - binhex (defined in RFC 1741)
 - btoa
 - quoted-printable (defined in RFC 1521)
 - utf-7 (defined in RFC 2152)
 - uue
 - xxe
 - yenc

Additionally the body of a message can be encoded several ways. We support the following encodings:

- Text
- HTML
- RTF
- TNEF
- Text/enriched (defined in RFC1523)
- Text/richtext (defined in RFC1341)
- Embedded mail message (defined in RFC 822). This is handled as a link to a new message.

Note: The attachments of a MIME message can be stored in many formats. All attachments of supported file formats can be converted.

C.9 Other Formats

File Format	Comments
Executable (EXE, DLL)	
HTML	Versions through 3.0, with some limitations
Macromedia Flash	Macromedia Flash 6.x, Macromedia Flash 7.x, and Macromedia Flash Lite (text only)
Microsoft Project	Versions 98 through 2003 (text only). (MPP files are treated as database files.)
vCard, vCalendar	Version 2.1
Windows Executable	
XML	Text only
Yahoo! Instant Messenger	Versions 6.x and 7.x

Office 2007 Considerations

This section provides a number of considerations related to conversion of Office 2007 files:

- ["All Office Applications"](#) on page D-1
- ["Word 2007"](#) on page D-1
- ["Excel 2007"](#) on page D-2
- ["PowerPoint 2007"](#) on page D-2
- ["Examples of Unsupported Objects"](#) on page D-3

D.1 All Office Applications

Please note the following conversion limitations that currently apply for all Office 2007 applications:

- Smart art (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
- VB controls and macros (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
- Table cell formatting
- Word art (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
- Vector graphics (Office art & VML) transparency, picture styles, effects, etc. (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
- Password-protected documents

D.2 Word 2007

Please note the following conversion limitations that currently apply for Word 2007 documents:

- Picture bullets
- Tint support
- Table styles (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
 - Different column and row definitions (even/odd, etc.)
 - Different header row and column definitions
- List level overrides
- Alternate text

- OLE objects
- Equations (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
- Theme effects (in Office art)
- Line numbers
- Watermarks
- Page color (not supported in the viewer)
- Footnote and end note reference numbers
- Revision delete attributes (text is supported)
- Controls (only last edited text is output for legacy controls)
- Custom XML (structure, schemas, expansion packs), cfChunk/altChunks are not supported

D.3 Excel 2007

Please note the following conversion limitations that currently apply for Excel 2007 spreadsheets:

- Conditional formatting (highlight cells with rules, top bottom rules, data bars, color scale icon sets, and custom rules; see ["Examples of Unsupported Objects"](#) on page D-3 for an example).
- Formatting as tables (the data in the cell is output, but the formatting is not retained)
- Headers and footers (different even/odd page headers are not supported)
- Protected workbooks

D.4 PowerPoint 2007

Please note the following conversion limitations that currently apply for PowerPoint 2007 presentations:

- Table formatting (similar to Excel)
- Actions are currently not supported
- "Objects" (this is represented as VML; currently not supported)
- Movies/sounds are not supported
- Complex gradients are not supported (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
- Animation is currently not supported
- Only solid fills are supported for text
- Only left-to-right text direction is supported (not related to bidi)
- Shading and fills of certain shapes are not supported (see ["Examples of Unsupported Objects"](#) on page D-3 for an example)
- Transparency of lines/vector objects is not supported

D.5 Examples of Unsupported Objects

This section provides some examples of Office 2007 objects that cannot be converted at this point.

Figure D-1 Smart Art

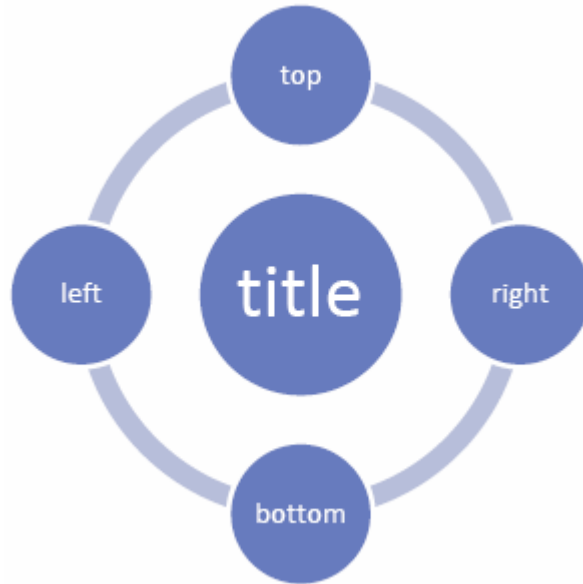


Figure D-2 Table Styles

A horizontal table with a dark blue header row and a light blue body. The table has four columns. The first column is wider than the others. The body consists of two rows of four cells each.

Figure D-3 Picture Styles/effects



Figure D-4 Word Art



Figure D-5 Equations

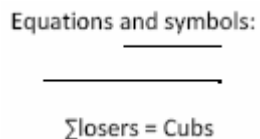


Figure D-6 Controls

Date picker: 9/22/2006 (this date is in a date picker control)

Figure D-7 Data Bars With Conditional Formatting, Color Scales, and Icon Sets

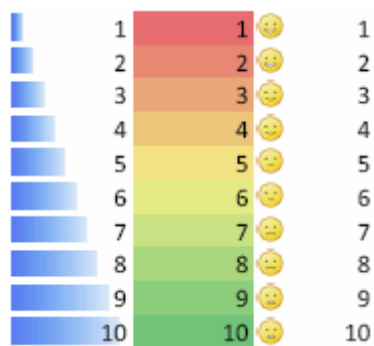


Figure D-8 3D Effects in PowerPoint



Figure D-9 *Complex Gradients*

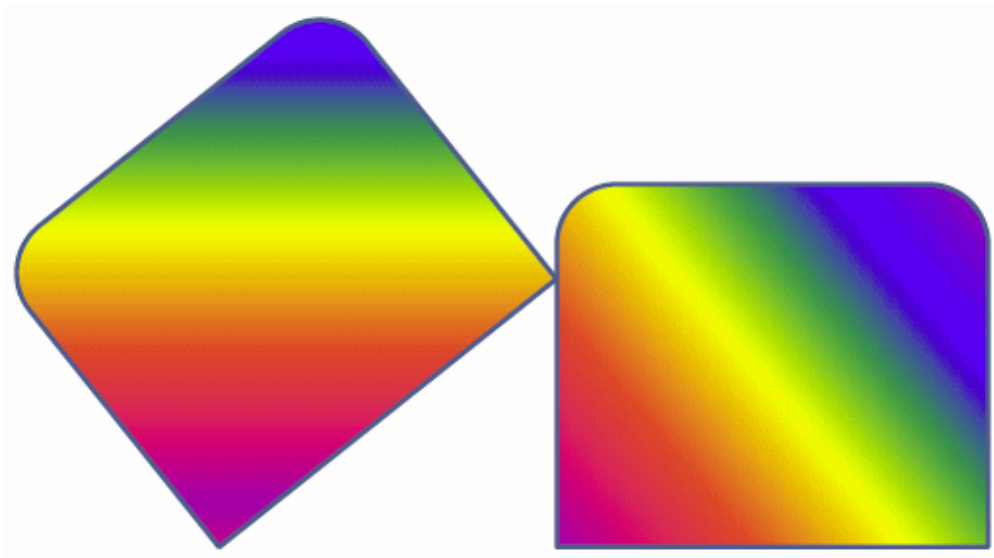
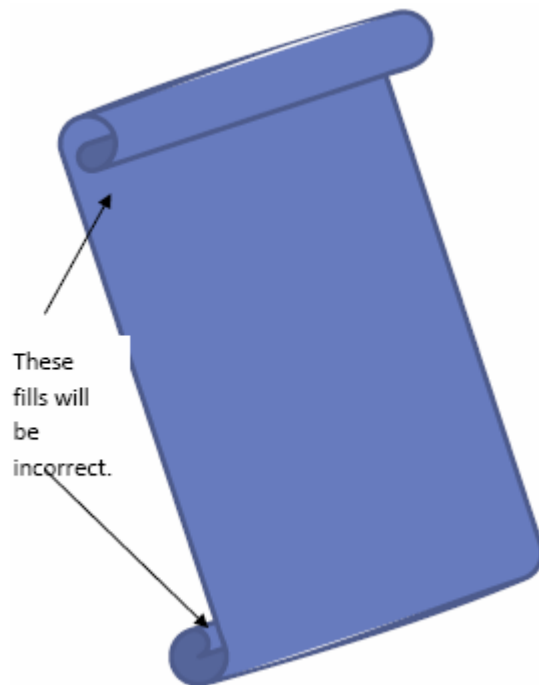


Figure D-10 *Complex Shapes With Varying Fills (1)*



Figure D-11 *Complex Shapes With Varying Fills (2)*



Elements Script Template

This section covers the following topics:

- ["About the Elements Script Template"](#) on page E-1
- ["Elements Script Template Code"](#) on page E-1

E.1 About the Elements Script Template

The Element script template separately defines all the elements of a source file:

- Standard properties (author, title, subject, keywords, comments).
- Other properties that might be included by the author of the source file.
- Sections of the source file.
- All other properties (footnotes, endnotes, annotations, comments, headers, footers, bookmarks).

The template is not called by any other template, and simply acts to separate all source file elements.

E.2 Elements Script Template Code

Each of the code segments in this section (in bold) is followed by explanatory text.

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML//EN">
<html>

<head>
<meta http-equiv="Content-Type"
content="text/html; charset={## INSERT ELEMENT=pragma.charset}">
{## IF ELEMENT=Property.Title}
<TITLE>{## INSERT ELEMENT=Property.Title}</TITLE>
{## ELSE}
<TITLE>Converted {## INSERT ELEMENT=Pragma.SourceFileName}</TITLE>
{## /IF}
```

If a title property exists in this document, then insert it into the HTML title. Otherwise, insert the name of the document into the HTML title.

```
</head>

<body bgcolor="#FFFFFF">
<div align="left">

<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
```

```

    <tr>
      <td><font size="7">Dynamic Converter</font><br>
        <font color="#FFFFFF" size="5">Sample Template - All Elements</font>
      </td>
    </tr>
  </table>
</div>

```

Defines the table at the top of the output page, and the text within the table ("Dynamic Converter Sample Template – All Elements"), as well as the font color and size of this text.

```

<p>&nbsp;</p>
<div align="left">

<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
  <tr>
    <td><font color="#FFFFFF" size="5">Standard Properties</font></td>
  </tr>
</table>
</div>

```

Defines the second table of the output page.

```

<p><strong>Author: </strong>
{## INSERT ELEMENT=Property.Author}<br>
<strong>Title: </strong>
{## INSERT ELEMENT=Property.Title}<br>
<strong>Subject: </strong>
{## INSERT ELEMENT=Property.Subject}<br>
<strong>Keywords: </strong>
{## INSERT ELEMENT=Property.Keywords}<br>
<strong>Comment:</strong>
{## INSERT ELEMENT=Property.Comment}</p>

```

These ## Insert macros insert the source file's respective properties (Author, Title, Subject, Keywords, Comments), as written by the author of the source file being exported.

```

<div align="left">

<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
  <tr>
    <td><font color="#FFFFFF" size="5">Other Properties</font></td>
  </tr>
</table>
</div>

```

Defines the third table of the output page.

```

<p>{## REPEAT ELEMENT=Property.Others}
<strong>
{## INSERT
ELEMENT=Property.Others.Current.Name}: </strong>
{## INSERT ELEMENT=Property.Others.Current.Body}<br>
{## /REPEAT}</p>

```

Inserts the name and body of the exported source file's other properties, as defined by the author of the source file. The "Name" and "Body" properties are referenced differently than the initial properties (Author, Title, Subject, Keywords, Comments). The author of a source file can create separate properties that do not have the

keywords of the initial properties, and the above `## REPEAT` macro allows these separate properties to be read. This is achieved by looping through the source file's other, unspecified properties by referencing and outputting both the name and body elements for these unspecified properties.

```
<p>{## REPEAT ELEMENT=Sections}</p>
```

A loop on a repeatable element, "Sections" is used to represent the highest level of abstraction within the source file. This repeatable element would allow a loop to be performed on a three-sheet spreadsheet, for example, so that each sheet is shown in the output. It is necessary to loop through sections in order to output each separate part of a section. In general:

- Word processor documents will have only one section, that being the document itself.
- Spreadsheets will have one section for each sheet or chart.
- Presentations have one section for each slide.
- Graphics, in most cases, have only one section, but many have multiple sections, such as a multi-page TIFF. For convenience and readability, "Sheets" and "Slides" are synonymous with "Sections" in Dynamic Converter.

```
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
  <td><font color="#FFFFFF" size="5">Section
  {## INSERT NUMBER=Sections.Current.Value}
  The fourth table of the output page is defined and the index value that is
  inserted into the table is incremented once through each loop.

  {## IF ELEMENT=Sections.Current.Type VALUE=WP}
  - Document
  {## ELSE}
  {## IF ELEMENT=Sections.Current.Title} - {## INSERT ELEMENT=Sections.Current.Title
  SUPPRESSTAGS}
  {## /IF}

  {## /IF}

</font>
</td>
```

The template determines if the source file is word processing (WP) and:

- If the source file is word processing, "Document" is placed in the output table following "Section" and the index value.
- If the source file is not word processing, the template will check for the title of the source file and place that title in the output. Finally, tags that might be present within the source file are suppressed from the output of the source file. This suppression is done to strip HTML tags that Dynamic Converter adds in order to duplicate the source file's original font. This is done in situations where either
 - Plain text is appropriate, or
 - The template author wishes to control the appearance of the output text.

```
</tr>
</table>
</div>
```

```
<p>
{## INSERT ELEMENT=Sections.Current.BodyOrImage}</p>
```

This macro line allows the template author to account for any document type. See Sections.x.BodyOrImage in the "Elements" section.

```
<p>
{## IF ELEMENT=Sections.Current.Type VALUE=WP}
</p>
<p>
{## IF ELEMENT=Sections.Current.Footnotes.1.Body}
</p>
<div align="left">
<table border="0" cellpadding="2"width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">Footnotes</font>
</td>
</tr>
</table>
```

The template first determines if the section type is word processing and then if there are footnotes included with this section. If footnotes are included, a table is defined that introduces "Footnotes."

```
</div>
<p>
{## REPEAT ELEMENT=Sections.Current.Footnotes}
{## INSERT ELEMENT=Sections.Current.Footnotes.Current.Body}
<br>
{## /REPEAT}
{## ELSE}
</p>
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">No Footnotes</font></td>
</tr>
</table>
</div>
<p>
{## /IF}

</p>
```

A repeatable element outputs all footnotes associated with the current section. If there are no footnotes associated with the current section, a table is created indicating "No Footnotes."

The rest of the HTML coding for this template concerns elements that are similar in their coding to the recently described "Footnotes" repeatable element. For example, for the next repeatable element "Endnotes," the coding acts to present the endnotes in the source file output in the same way as that for the footnotes:

- The template determines if there are endnotes included in the current section.
- If endnotes are included, a table is defined that introduces "Endnotes."

A repeatable element outputs all endnotes associated with the section.

- If there are no endnotes associated with the section, a table is created indicating "No Endnotes."

The following HTML coding and resulting output will follow the same procedure for annotations, comments, headers, footers and bookmarks.

```

<p>
{## IF ELEMENT=Sections.Current.Endnotes.1.Body}
</p>
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">Endnotes</font></td>
</tr>
</table>
</div>
<p>
{## REPEAT ELEMENT=Sections.Current.Endnotes}
{## INSERT ELEMENT=Sections.Current.Endnotes.Current.Body}<br>
{## /REPEAT}

{## ELSE}
</p>
<div align="left"><table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF"
size="5">No Endnotes</font></td>
</tr>
</table>
</div>
<p>
{## /IF}

</p>
<p>
{## IF ELEMENT=Sections.Current.Annotations.1.Body}
</p>
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><fontcolor="#FFFFFF" size="5">Annotations</font></td>
</tr>
</table>
</div>
<p>
{## REPEAT ELEMENT=Sections.Current.Annotations}
{## INSERT ELEMENT=Sections.Current.Annotations.Current.
Body}<br>
{## /REPEAT}

{## ELSE}
</p>
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">No Annotations</font></td>
</tr>
</table>
</div>
<p>
{## /IF}

</p>
<p>

```

```
{## IFELEMENT=Sections.Current.Headers.1.Body}
</p>
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">Headers</font></td>
</tr>
</table>
</div>
<p>
{## REPEAT ELEMENT=Sections.Current.Headers}
{## INSERT ELEMENT=Sections.Current.Headers.Current.Body} <br>
{## /REP}

{## ELSE}
</p>
<div align="left"><table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">No Headers</font></td>
</tr>
</table>
</div>
<p>
{## /IF}

</p>
<p>
{## IF ELEMENT=Sections.Current.Footers.1.Body}
</p>
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">Footers</font></td>
</tr>
</table>
</div>
<p>
{## REPEAT ELEMENT=Sections.Current.Footers}
{## INSERT ELEMENT=Sections.Current.Footers.
Current. Body}<br>
{## /REP}

{## ELSE}
</p>
<div align="left">
<table border="0" cellpadding="2" width="600" bgcolor="#8080FF">
<tr>
<td><font color="#FFFFFF" size="5">No Footers</font></td>
</tr>
</table>
</div>
<p>
{## /IF}

</p>
<p>
{## /IF}

</p>
```



```
<p>  
{## /REPEAT}  
  
</p>  
</body>  
</html>
```


Symbols

%%TRANSIT-BODYATTRIBUTES%%, 6-3
(HTML) link, 9-4
<!-- TRANSIT - CUSTOMLAYOUT(BODY)-->, 6-3
<!--TRANSIT - CUSTOMLAYOUT(HEAD)-->, 6-3
<!--TRANSIT - CUSTOMLAYOUT(TOP)-->, 6-3
{## ANCHOR}, 7-23
{## COMMENT}, 7-24
{## COPY}, 7-28
{## ELSE}, 7-18
{## ELSEIF}, 7-18
{## FOOTER}, 7-13
{## HEADER}, 7-13
{## IF}, 7-18
{## IGNORE}, 7-24
{## INCLUDE}, 7-25
{## INSERT}, 7-14
{## LINK}, 7-10, 7-20
{## LOOP}, 7-19
{## OPTION}, 7-25
{## REPEAT}, 7-10
{## UNIT}, 7-13

A

Academy (GUI template), 5-12
Acclaim CSS (GUI template), 5-12
Account (GUI template), 5-13
Adagio CSS (GUI template), 5-13
adding
 file formats for conversion, 2-2
 new GUI template, 5-2
 new layout template, 6-5
 template selection rule, 3-2, A-10
Adding Document Properties, 5-4
adding file formats for --, 2-2
Adding Navigation Elements, 5-5
Adding Output Markup Items, 5-6
Adding Output Page Layouts, 5-8
Adding Output Text Formats, 5-6
Adding Text Elements, 5-5
Admin link in Dynamic Converter, 1-9
Administration (GUI template), 5-13
administration page for Dynamic Converter, A-1
Analysis (GUI template), 5-14

ANCHOR macro, 7-23
application filters, B-1
Archive CSS (GUI template), 5-14
Archives, 5-4
associating templates with rules, 3-3, 6-6

B

Basic (script template), 7-32
Blank (GUI template), 5-14
body content in layout templates, 8-3
breaking documents by content size, 7-46
breaking documents by structure, 7-43
browser caching, 10-4
bulleted lists, 7-43
Business (GUI template), 5-15

C

caching, 1-5
 browsers, 10-4
 cache interval, A-8
 duration, 1-6, A-8
 expiration period, 1-6, A-8
 maximum cache size, 1-6, A-8
 optimizations, A-7
 timestamps, 1-5
cascading style sheets (CSS), 10-5
Ceremonial (GUI template), 5-15
changes to metadata, 1-5, A-8
Check In Existing Template link, A-1
check-in
 preview before --, 9-5
check-in form for templates, A-12
checking frequency of timestamps, 1-6, A-8
checking in templates, 4-2
choosing a template, 4-2
Classic HTML Conversion Layout Templates, 6-1
Classic HTML Conversion Layout templates, 4-2
classic HTML conversion layout templates
 overview, 6-1
Classic HTML Conversion Template, 5-9
Classic HTML Conversion templates, 4-1
combining HTML snippets into a web page, 8-2
COMMENT macro, 7-24
comments in script templates, 7-24

- Complete Check In link, 9-5
- compressed formats, C-6
- conditionals in script templates, 7-18
- configuration
 - general conversion settings, A-3
 - Slideshow templates, 2-3
 - wireless template type, 2-5
- configuration page of Dynamic Converter, A-2
- configuration settings
 - caching optimization, A-7
 - conversion formats, 2-2, A-4
 - conversion optimizations, A-7
 - database method for reconversion checking, A-8
 - default layout, 2-1, 6-6, A-4
 - default template, 2-1, A-3
 - display information, A-5
 - font path, A-5
 - general settings, A-3
 - GUI template conversion, A-6
 - maximum file size, A-4
 - reconvert after metadata updates, A-8
 - reevaluate rules during re-indexing, A-8
 - rendition, A-5
 - rule criteria, A-4
 - script template conversion, A-7
 - timeout, 10-2, A-4
 - timestamp check frequency, A-8
 - UNIX, A-5
- Configuration Settings link, A-1
- Configuring HTML Settings, 5-5
- content information, 9-1
- content information page, 9-1, 9-4
- Content Server
 - Dynamic Converter interface, 1-9
 - metadata on web pages, 8-4
- content size, breaking documents by --, 7-46
- contents of layout templates, 6-2
- conversion, 2-2, 2-3
 - file extension, A-12
 - forced --, 1-4
 - fragment-only --, 1-4, A-12
 - HTML forms, 1-7
 - inline --, 8-4
 - metadata changes, 1-5, A-8
 - paragraphs as graphics, 1-8
 - PDF files under UNIX, 10-2
 - previewing documents, 9-5
 - rendition for --, A-5
 - script template settings, A-7
 - script templates to GUI templates, 5-19
 - timeout, 10-2, A-4
 - upfront, 1-3
 - upfront --, A-8
 - viewing converted files, 9-4
 - XML, 1-8
- conversion filters
 - applications, B-1
 - graphics, B-5
- conversion formats, 2-2, A-4
 - adding --, 2-2

- removing --, 2-3
- conversion process, 1-2
- COPY macro, 7-28
- Count and CountB0, 7-17
- Courtesy (GUI template), 5-16
- Create New Template link, A-1
- creating
 - GUI template, 5-2
 - layout template, 6-5
- criteria for template selection rules, 3-3
- CSS in layout templates, 6-7
- CSS, see 'cascading style sheets'
- Current, Next, and Previous, 7-10

D

- Database, 5-4
- database files, 7-48
- database formats, C-3
- database method of checking metadata updates, 1-6, A-8
- dated cache interval, A-8
- dcLoadDocInfo(), 8-4
- dcURL(), 10-3
- dcURL() function, 10-3
- default_layout.txt (sample layout), 6-3
- defaults
 - GUI template, 5-14
 - layout, 2-1, 6-6, A-4
 - template, 2-1, A-3
- Defining, 3-1
 - Templates, 3-1
- deleting
 - cached items, 1-6, A-8
 - file formats for conversion, 2-3
 - template selection rule, 3-2
 - wireless template type, 2-5
- desktop publishing formats, C-3
- developer, 1-2
- DHTML, 10-7
- display information on UNIX, A-5
- displaying metadata on web pages, 8-4
- duration of caching, 1-6, A-8
- Dynamic Converter
 - basic concepts, 1-2
 - caching, 1-5
 - configuration, 2-1
 - conversion process, 1-2
 - interface, 1-9
 - overriding styles, 10-6
- Dynamic Converter Admin link, 1-9
- Dynamic Converter Admin page, A-1
- Dynamic Converter Configuration page, A-2

E

- Edit Existing Template link, A-14
- Edit Templates link, A-1
- Edit Templates page, A-14
- editing GUI template, 5-3

- editor for GUI templates, 5-9
- element tree, 7-2
- Elements (script template), 7-33, E-1
- elements in GUI templates, 5-10
- elements in script templates, 7-2, 7-5
 - see also 'leaf elements' and 'repeatable elements'
 - Pragma.Charset, 7-8
 - Pragma.CSSFile, 7-8
 - Pragma.EmbeddedCSS, 7-9
 - Pragma.JsFile, 7-9
 - Pragma.SourceFileName, 7-8
 - Property.Author, 7-5
 - Property.Comments, 7-5
 - Property.Keywords, 7-5
 - Property.Others, 7-5
 - Property.Others.x.Body, 7-5
 - Property.Others.x.Name, 7-5
 - Property.Subject, 7-5
 - Property.Title, 7-5
 - Sections, 7-5
 - Sections.x.Annotations, 7-7
 - Sections.x.Annotations.x.Body, 7-7
 - Sections.x.Annotations.x.Content, 7-7
 - Sections.x.Annotations.x.Reference, 7-7
 - Sections.x.Body, 7-6
 - Sections.x.Body.Contents, 7-6
 - Sections.x.Body.Contents.Headings, 7-6
 - Sections.x.Body.Contents.Headings.x.Annotations, 7-6
 - Sections.x.Body.Contents.Headings.x.Body, 7-6
 - Sections.x.Body.Contents.Headings.x.Endnotes, 7-6
 - Sections.x.Body.Contents.Headings.x.Footnotes, 7-6
 - Sections.x.Body.Contents.Preface, 7-6
 - Sections.x.Body.OrImage, 7-7
 - Sections.x.Body.Title, 7-6
 - Sections.x.Endnotes.x.Body, 7-7
 - Sections.x.Endnotes.x.Content, 7-7
 - Sections.x.Endnotes.x.Reference, 7-7
 - Sections.x.Footers, 7-8
 - Sections.x.Footers.x.Body, 7-8
 - Sections.x.Footnotes, 7-7
 - Sections.x.Footnotes.x.Body, 7-7
 - Sections.x.Footnotes.x.Content, 7-7
 - Sections.x.Footnotes.x.Reference, 7-7
 - Sections.x.Grids, 7-6
 - Sections.x.Grids.x.Body, 7-6
 - Sections.x.Headers, 7-7
 - Sections.x.Headers.x.Body, 7-8
 - Sections.x.Image, 7-6
 - Sections.x.Slidenotes, 7-7
 - Sections.x.Slidenotes.x.Body, 7-7
 - Sections.x.Title, 7-7
 - Sections.x.Type, 7-7
 - Sheets, 7-5
 - Slides, 7-5
 - ELSE macro, 7-18
 - ELSEIF macro, 7-18
 - e-mail formats, C-7

- e-mail notifications, 9-5
- embedded graphics, 10-2
- excluding rules from user requests, A-12
- Executive (GUI template), 5-16
- expiration period of cache, 1-6, A-8
- extension of converted files, A-12

F

- file extension
 - script templates, 4-2
- file extension of converted files, A-12
- file format, 7-43
- file formats
 - adding -- for conversion, 2-2
 - compressed formats, C-6
 - criterion for template selection rule, A-11
 - database formats, C-3
 - desktop publishing formats, C-3
 - e-mail formats, C-7
 - graphic formats, C-5
 - Office 2007, D-1
 - other formats, C-8
 - presentation formats, C-4
 - removing -- for conversion, 2-3
 - spreadsheet formats, C-4
 - word processing formats, C-1
- file formats for conversion, 2-2
- file size, maximum --, A-4
- filters
 - applications, B-1
 - graphics, B-5
- fonts under UNIX, A-5
- FOOTER macro, 7-13
- forced conversion
 - conversion
 - forced --, A-8, A-12
- forced conversions, 1-4
- format of graphics, 7-43
- formats for conversion, 2-2, A-4
- Formatting Different File Types, 5-4
- formatting options for script templates, 7-42
- forms, conversion of --, 1-7
- fragment-only conversions, 1-4, A-12
- frames, positional --, 10-7
- frequency of cache evaluations, A-8
- frequency of checking timestamps, 1-6, A-8

G

- generating HTML snippets, 8-2
- graphic formats, C-5
- graphics
 - embedded, 10-2
 - rendering paragraphs as --, 1-8
- graphics file format, 7-43
- graphics filters, B-5
- grids, 7-48
- GUI templates, 4-1
 - Academy, 5-12

- Acclaim CSS, 5-12
- Account, 5-13
- Adagio CSS, 5-13
- Administration, 5-13
- Analysis, 5-14
- Archive CSS, 5-14
- Blank, 5-14
- Business, 5-15
- Ceremonial, 5-15
- configuration settings, A-6
- Courtesy, 5-16
- creating --, 5-2
- displaying metadata, 8-4
- edit template page, A-14
- editing --, 5-3, 5-9
- elements, 5-10
- Executive, 5-16
- Introduction CSS, 5-17
- Lotus 1-2-3, 5-17
- Lotus Freelance, 5-17
- migrating from script templates to --, 5-19
- MS Excel, 5-18
- MS PowerPoint, 5-18
- patterns, 5-11
- Purple Frost, 5-18
- ranks, 5-10
- relative URLs, 10-4
- Retrofied! CSS, 5-19
- samples, 5-11
- styles, 5-11
- Template Editor, 5-9

H

- hcst files, 4-2
- HEADER macro, 7-13
- HTML Conversion templates, 4-2
 - overview, 5-1
- HTML forms, 1-7
- (HTML) link, 9-4
- HTML Rendition link, 9-4
- HTML snippets, 6-3, 6-4, 8-1
 - combining -- into a web page, 8-2
 - generating --, 8-2
 - overview, 8-1
 - referencing --, 8-3
- HTML, well-formed --, 10-7
- hyperlinks
 - dcUrl() tags around --, 10-3
 - service calls for -- within documents, A-6

I

- Idoc Script, 6-7
 - dcLoadDocInfo(), 8-4
 - dcURL(), 10-3
 - incDynamicConversion(), 8-3
 - incInlineDynamicConversion(), 8-4
- IF macro, 7-18
- IGNORE macro, 7-24

- image sizing, 10-5
- Images, 5-4
- images
 - embedded, 10-2
- images in layout templates, 6-7
- incDynamicConversionByRule Idoc, A-12
- incInlineDynamicConversion(), 8-4
- INCLUDE macro, 7-25
- index variable keywords, 7-9
- indexes, 7-9, 7-46, 7-48
- inline dynamic conversion, 8-4
- input file formats
 - compressed formats, C-6
 - database formats, C-3
 - desktop publishing formats, C-3
 - e-mail formats, C-7
 - graphic formats, C-5
 - Office 2007, D-1
 - other formats, C-8
 - presentation formats, C-4
 - spreadsheet formats, C-4
 - word processing formats, C-1
- INSERT macro, 7-14
- inserting elements in script templates, 7-14
- interface of Dynamic Converter, 1-9
- interval of cache evaluations, A-8
- intradocument links, A-6
- Introduction CSS (GUI template), 5-17

L

- layout templates
 - associating -- with rules, 6-6
 - contents, 6-2, 8-3
 - creating --, 6-5
 - default, 6-6, A-4
 - default_layout.txt, 6-3
 - displaying metadata, 8-4
 - including scripts, images, and css, 6-7
 - relative URLs, 10-4
 - samples, 6-3
 - selection rules for --, A-12
 - snippet_layout.txt, 6-4
 - tokens, 6-3
 - with body content only, 8-3
- leaf elements, 7-4
- LINK macro, 7-10, 7-20
- Link Mapping Rules, 5-5
- linking with content size breaking in script templates, 7-23
- linking with structured breaking in script templates, 7-20
- links
 - Check In Existing Template, A-1
 - Complete Check In, 9-5
 - Configuration Settings, A-1
 - Create New Template, A-1
 - Edit Existing Template, A-14
 - Edit Templates, A-1
 - (HTML), 9-4

- HTML Rendition, 9-4
- native file, 9-2
- Template Selection Rules, A-1
- View Info, 9-5
- web-viewable file, 9-2
- links, service calls for -- within documents, A-6
- LOOP macro, 7-19
- loops in script templates, 7-19
- Lotus 1-2-3 (GUI template), 5-17
- Lotus Freelance (GUI template), 5-17

M

- macros, 7-12
 - {## ANCHOR}, 7-23
 - {## COMMENT}, 7-24
 - {## COPY}, 7-28
 - {## ELSE}, 7-18
 - {## ELSEIF}, 7-18
 - {## FOOTER}, 7-13
 - {## HEADER}, 7-13
 - {## IF}, 7-18
 - {## IGNORE}, 7-24
 - {## INCLUDE}, 7-25
 - {## INSERT}, 7-14
 - {## LINK}, 7-10, 7-20
 - {## LOOP}, 7-19
 - {## OPTION}, 7-25
 - {## REPEAT}, 7-10
 - {## UNIT}, 7-13
- maximum cache size, 1-6, A-8
- maximum file size, A-4
- metadata, 9-1
 - conversion after -- changes, 1-5, A-8
 - criteria for rules, 3-3
 - displaying -- on web pages, 8-4
 - multibyte characters, 10-1
- MIME support, C-7
- MIME type, A-11
- MS Excel (GUI template), 5-18
- MS PowerPoint (GUI template), 5-18
- multibyte characters, 10-1

N

- names of styles, 10-5
- native file link, 9-2
- navigation in databases, 7-48
- navigation in presentations, 7-12, 7-22
- navigation in spreadsheets, 7-48
- notifications, 9-5
- numbered lists, 7-43

O

- Office 2007 conversion considerations, D-1
- Office 2007 files, 2-2
- OPTION macro, 7-25
- options in script templates
 - SCCOPT_EX_FONTFLAGS, 7-27
 - SCCOPT_EX_GRIDADVANCE, 7-28

- SCCOPT_EX_GRIDCOLS, 7-28
- SCCOPT_EX_GRIDROWS, 7-27
- SCCOPT_EX_GRIDWRAP, 7-28
- SCCOPT_GIF_INTERLACED, 7-26
- SCCOPT_GRAPHIC_HEIGHTLIMIT, 7-27
- SCCOPT_GRAPHIC_OUTPUTDPI, 7-27
- SCCOPT_GRAPHIC_SIZELIMIT, 7-27
- SCCOPT_GRAPHIC_SIZEMETHOD, 7-26
- SCCOPT_GRAPHIC_TYPE, 7-26
- SCCOPT_GRAPHIC_WIDTHLIMIT, 7-27
- SCCOPT_JPEG_QUALITY, 7-26

- output files, 1-2
- overriding Dynamic Converter styles, 10-6

P

- paragraphs, rendering -- as graphics, 1-8
- path to fonts, A-5
- patterns (GUI templates), 5-11
- PDF files, 10-2
- Plain (script template), 7-34
- portal-style website, 8-1
- positional frames, 10-7
- PowerPoint, 7-12, 7-22
 - configuring Slideshow templates, 2-3
- Pragma.Charset, 7-8, 7-30
- Pragma.CSSFile, 7-8, 7-30, 10-6
- Pragma.EmbeddedCSS, 7-9, 7-31
- Pragma.JsFile, 7-9, 7-31
- pragmas
 - Pragma.Charset, 7-8, 7-30
 - Pragma.CSSFile, 7-8, 7-30, 10-6
 - Pragma.EmbeddedCSS, 7-9, 7-31
 - Pragma.JsFile, 7-9, 7-31
 - Pragma.SourceFileName, 7-8, 7-31
- Pragma.SourceFileName, 7-8, 7-31
- presentation, 7-22
- presentation formats, C-4
- Presentations, 5-4
- presentations, 7-12
- previewing documents before check-in, 9-5
- Previewing Your Content, 5-8
- process, 1-2
- Property.Author, 7-5
- Property.Comments, 7-5
- Property.Keywords, 7-5
- Property.Others, 7-5
- Property.Others.x.Body, 7-5
- Property.Others.x.Name, 7-5
- Property.Subject, 7-5
- Property.Title, 7-5
- Purple Frost (GUI template), 5-18

R

- ranks (GUI templates), 5-10
- raster graphics, 10-2
- rasterization, A-6
- reconversion
 - database checking, A-8

- database method update checking, 1-6
- metadata updates, 1-5, A-8
- referencing HTML snippets, 8-3
- re-indexing
 - re-evaluating rule during --, A-8
- relative URLs, 10-4
- removing file formats for --, 2-3
- rendering paragraphs as graphics, 1-8
- rendition for conversion, A-5
- reordering template selection rule, 3-2
- REPEAT macro, 7-10
- repeatable elements, 7-5
- Retrofied! CSS (GUI template), 5-19
- revision history, 9-2
- rewriting of URLs, 10-3
- rule criteria, A-4
- Rules, 3-1
- rules, see 'template selection rules'

S

- samples
 - GUI templates, 5-11
 - layout templates, 6-3
 - script templates, 7-32, E-1
- Saving Your Template, 5-9
- SCCOPT_EX_FONTFLAGS, 7-27
- SCCOPT_EX_GRIDADVANCE, 7-28
- SCCOPT_EX_GRIDCOLS, 7-28
- SCCOPT_EX_GRIDROWS, 7-27
- SCCOPT_EX_GRIDWRAP, 7-28
- SCCOPT_GIF_INTERLACED, 7-26
- SCCOPT_GRAPHIC_HEIGHTLIMIT, 7-27
- SCCOPT_GRAPHIC_OUTPUTDPI, 7-27
- SCCOPT_GRAPHIC_SIZELIMIT, 7-27
- SCCOPT_GRAPHIC_SIZEMETHOD, 7-26
- SCCOPT_GRAPHIC_TYPE, 7-26
- SCCOPT_GRAPHIC_WIDTHLIMIT, 7-27
- SCCOPT_JPEG_QUALITY, 7-26
- script templates, 4-2
 - breaking documents by content size, 7-46
 - breaking documents by structure, 7-43
 - bulleted lists, 7-43
 - comments, 7-24
 - conditionals, 7-18
 - conversion configuration, A-7
 - element tree, 7-2
 - elements, 7-2, 7-5
 - file extension, 4-2
 - formatting options, 7-42
 - graphics file format, 7-43
 - indexes, 7-9, 7-46, 7-48
 - inserting elements, 7-14
 - leaf elements, 7-4
 - linking with content size breaking, 7-23
 - linking with structured breaking, 7-20
 - loops, 7-19
 - macros, 7-12
 - migrating to GUI templates, 5-19
 - numbered lists, 7-43

- overview, 7-1
- relative URLs, 10-4
- repeatable elements, 7-5
- samples, 7-32, E-1
- units, 7-13
- scripts in layout templates, 6-7
- scripts templates
 - Basic, 7-32
 - Elements, 7-33, E-1
 - Plain, 7-34
 - SimpleToc, 7-35
 - Slideshow, 7-37
 - Textout, 7-40
- search results page, 9-4
- Sections element, 7-5
- Sections.x.Annotations, 7-7
- Sections.x.Annotations.x.Body, 7-7
- Sections.x.Annotations.x.Content, 7-7
- Sections.x.Annotations.x.Reference, 7-7
- Sections.x.Body, 7-6
- Sections.x.Body.Contents, 7-6
- Sections.x.Body.Contents.Headings, 7-6
- Sections.x.Body.Contents.Headings.x.Annotations..., 7-6
- Sections.x.Body.Contents.Headings.x.Body..., 7-6
- Sections.x.Body.Contents.Headings.x.Endnotes..., 7-6
- Sections.x.Body.Contents.Headings.x.Footnotes..., 7-6
- Sections.x.Body.Contents.Preface, 7-6
- Sections.x.BodyOrImage, 7-7
- Sections.x.Body.Title, 7-6
- Sections.x.Endnotes.x.Body, 7-7
- Sections.x.Endnotes.x.Content, 7-7
- Sections.x.Endnotes.x.Reference, 7-7
- Sections.x.Footers, 7-8
- Sections.x.Footers.x.Body, 7-8
- Sections.x.Footnotes, 7-7
- Sections.x.Footnotes.x.Body, 7-7
- Sections.x.Footnotes.x.Content, 7-7
- Sections.x.Footnotes.x.Reference, 7-7
- Sections.x.Grids, 7-6
- Sections.x.Grids.x.Body, 7-6
- Sections.x.Headers, 7-7
- Sections.x.Headers.x.Body, 7-8
- Sections.x.Image, 7-6
- Sections.x.Slidenotes, 7-7
- Sections.x.Slidenotes.x.Body, 7-7
- Sections.x.Title, 7-7
- Sections.x.Type, 7-7
- service calls for intradocument links, A-6
- settings, see 'configuration settings'
- Sheets element, 7-5
- SimpleToc (script template), 7-35
- size of files, maximum --, A-4
- size of images, 10-5
- Slides element, 7-5
- Slideshow templates, 2-3, 7-37
- snippet_layout.txt (sample layout), 6-4
- snippets, 6-3, 6-4, 8-1

- combining -- into a web page, 8-2
- generating --, 8-2
- overview, 8-1
- referencing, 8-3
- source file, 1-2
- spreadsheet formats, C-4
- Spreadsheets, 5-4
- spreadsheets, 7-48
- SRC image tags, 10-3
- structure, breaking documents by --, 7-43
- style names, 10-5
- style overrides, 10-6
- styles (GUI templates), 5-11
- subscription notifications, 9-5

T

- template, 1-2
 - associating -- with rules, 3-3, 6-6
 - selection of --, 4-2
- template check-in form, A-12
- Template Editor, 5-9
- template rules, 1-2
- template selection rules, A-4
 - adding --, 3-2, A-10
 - associating templates with --, 3-3, 6-6
 - deleting --, 3-2
 - excluding -- from user requests, A-12
 - file extension, A-12
 - layout associated with --, A-12
 - metadata criteria, 3-3
 - overview, 3-1
 - re-evaluating -- during re-indexing, A-8
 - reordering --, 3-2
 - template associated with --, A-12
- Template Selection Rules link, A-1
- Template Selection Rules page, A-9
- template types, 4-1
 - GUI template, 4-1
 - GUI templates, 5-1
 - script template, 4-2
 - script templates, 7-1
 - wireless, 2-5
- Templates
 - Defining, 3-1
- templates
 - checking in --, 4-2
 - creating new GUI --, 5-2
 - creating new layout --, 6-5
 - default template, 2-1, A-3
 - editing GUI --, 5-3, 5-9
 - GUI templates, 4-1, 5-1
 - overview, 4-1
 - relative URLs, 10-4
 - samples (GUI), 5-11
 - samples (layouts), 6-3
 - samples (script), 7-32, E-1
 - script templates, 4-2, 7-1
 - selection rules for --, A-12
 - Slideshow, 2-3

- strategy, 4-2
- writing tips, 10-7
- XML compliance, 8-2
 - see also 'GUI templates'
 - see also 'script templates'
 - see also 'layouts'
- Textout (script template), 7-40
- Text/Word Processing, 5-4
- timeout, 10-2, A-4
- timestamps
 - caching, 1-5
 - checking frequency, 1-6, A-8
- tips for writing templates, 10-7
- tokens in layout templates, 6-3
- TRANSIT - CUSTOMLAYOUT(BODY), 6-3
- TRANSIT - CUSTOMLAYOUT(HEAD), 6-3
- TRANSIT - CUSTOMLAYOUT(TOP), 6-3
- TRANSIT-BODYATTRIBUTES, 6-3
- types of templates, 4-1

U

- UNIT macro, 7-13
- units in script templates, 7-13
- UNIX
 - configuration settings, A-5
 - conversion of PDF files, 10-2
 - embedded graphics, 10-2
 - rasterization configuration settings, A-6
 - TrueType fonts, A-5
 - vector graphics, 10-2, 10-3
 - X-Server, 10-2, 10-3
 - X-Windows, A-6
- updating script templates to GUI templates, 5-19
- upfront conversion, 1-3, A-8
- URL rewriting, 10-3
- URLs, relative --, 10-4
- user agent, A-11
- user interface
 - Admin link, 1-9
 - Admin page, A-1
 - Dynamic Converter Configuration page, A-2
 - Edit Templates page, A-14
 - Template Check-In Form, A-12
 - Template Selection Rules page, A-9
- user interface of Dynamic Converter, 1-9

V

- Value and ValueB0, 7-17
- vector graphics, 10-2, 10-3
- View Info link, 9-5
- viewing content information, 9-1
- viewing converted files
 - content information page, 9-4
 - search results page, 9-4

W

- web location, 9-2
- websites as portals, 8-1

web-viewable file link, 9-2
well-formed HTML, 10-7
wireless template type, 2-5
word processing formats, C-1
workflow notifications, 9-5

X

XLS files, 1-8

XML

 conversion of --, 1-8

 GUI templates, 5-1

XML-compliant templates, 8-2

X-Server, 10-2, 10-3

X-Windows and rasterization, A-6