



# **ePublisher Platform Documentation**

Published date: 04/05/2025





# Table of Contents

<b>ePublisher Platform Documentation.....</b>	<b>1</b>
What's New in ePublisher 2024.1.....	3
Adapters.....	3
Formats.....	3
Reverb 2.0.....	3
PDF - XSL-FO.....	4
User Interface.....	4
Evaluation Materials.....	5
Build Improvements.....	5
Stationery.....	5
General Improvements.....	5
<b>ePublisher 2024.1 Release Notes.....</b>	<b>6</b>
Improvements.....	6
Fixed Issues.....	8
Summary.....	15
<b>Contacting Quadralay.....</b>	<b>20</b>
<b>Conventions.....</b>	<b>21</b>
<b>Introduction to the WebWorks ePublisher Platform.....</b>	<b>23</b>
What Is ePublisher?.....	23
Workflow.....	24
WebWorks ePublisher Platform Components.....	24
Supported Input Formats.....	25
Supported Output Formats.....	26
How ePublisher Helps You.....	26
Streamline and Automate the Content Publishing Process.....	26
Produce High Quality Deliverables with Fewer Individual Dependencies.....	27
Reduce Support Costs and Increase Customer Satisfaction.....	27
Quickly Update and Deliver Content More Often.....	28
Reduce Content Management Life Cycle Costs.....	28
How Organizations Use ePublisher.....	29
Automatically Update Content on Web Sites.....	29
Deliver Full-Featured, Context-Sensitive Help Systems.....	30

Produce Single-Sourced Print and Online Optimized Content.....	30
<b>Planning and Installing ePublisher.....</b>	<b>32</b>
Licensing Considerations.....	32
Components and Supported Configurations.....	32
Requirements.....	33
ePublisher Express, ePublisher Designer, and ePublisher AutoMap Requirements.....	33
Additional Source Document Requirements.....	34
Additional Output Format Requirements.....	35
WebWorks Reverb 2.0.....	35
Configuring web server for Reverb.....	35
Reverb browser requirements.....	36
WebWorks Reverb 1.0 Limitation.....	36
Dynamic HTML.....	36
PDF -XSL-FO.....	37
eBook - ePUB 2.0.....	37
Eclipse Help.....	37
Microsoft HTML Help 1.x.....	37
Oracle Help.....	38
PDF.....	38
Sun JavaHelp 2.0.....	38
WebWorks Help 5.0.....	38
Downloading ePublisher Installers.....	39
Microsoft Windows Requirements.....	41
Downloading and Installing the Microsoft .NET 4.7.2 Framework.....	41
Installing ePublisher.....	41
Installation Order for ePublisher Components.....	42
Installing ePublisher Components.....	42
Installing Ghostscript.....	44
Ghostscript not Installed Warnings.....	45
Configuring AutoMap for Microsoft Source Document Inputs.....	45
Understanding Installed Sample Projects and Stationery.....	48
Working with Contract IDs.....	48
Viewing Licensing and Contract ID Information.....	49
Obtaining Contract IDs.....	50
Entering Contract IDs.....	50
Managing Licensing in Environments without Internet Connectivity.....	51



Updating Licensing.....	51
Deactivating Licensing.....	53
Upgrading from Previous Versions.....	54
Updating ePubublisher installation.....	54
Preparing existing projects for ePubublisher Upgrade.....	55
Upgrading Typical ePubublisher Implementations.....	56
Upgrading Implementations with Advanced Customizations.....	58
Upgrading Advanced Customizations of WebWorks Reverb 2.0.....	59
Uninstalling ePubublisher.....	61
Troubleshooting Installation, License Keys, and Uninstallation.....	64
Problems Installing ePubublisher.....	64
Error: Please Close all Running Sessions of Microsoft Word.....	64
Problems with FrameMaker or Microsoft Word.....	66
Error: Error Communicating with Adobe FrameMaker.....	67
Error: Cannot Duplicate Document.....	68
Problems with Contract IDs and Licensing.....	69
No Contract ID Received.....	69
Error: No Valid License Key Found.....	69
Other Contract ID and Licensing Problems.....	69
<b>Exploring ePubublisher.....</b>	<b>70</b>
Understanding the ePubublisher Workflow.....	70
Stationery Designers and ePubublisher Designer.....	70
Writers and ePubublisher Express.....	72
Automating Output Generation with ePubublisher AutoMap.....	73
Exploring the ePubublisher User Interfaces.....	73
Exploring the ePubublisher Express User Interface.....	74
Exploring the ePubublisher Designer User Interface.....	75
Understanding the Start Page.....	75
Understanding Document Manager.....	76
Including or Excluding Files.....	76
Understanding Output Explorer.....	77
Understanding the Log Window.....	79
Understanding Style Designer.....	79
Understanding the Preview Window.....	80
Exploring the ePubublisher AutoMap User Interface.....	81
Customizing Your ePubublisher Workspace.....	82
Specifying General ePubublisher Preferences.....	82

<b>Miscellaneous ePublisher Windows.....</b>	<b>85</b>
Add New Target Window.....	85
Conditions Window.....	86
Classic Tab.....	86
Expressions Tab (FrameMaker Only).....	87
Cross Reference Rules Window.....	87
Deployment Configuration (Name) Window.....	89
Deployment Editor Window.....	89
Documents Window.....	89
Edit Target Window.....	90
File Mapping Editor Window.....	90
Folder Deployment Editor Window.....	91
Target Settings Window.....	91
Generated output location.....	91
Deploy to.....	92
List of Target Settings.....	92
Job Info Window.....	92
License Information Window.....	93
Main ePublisher AutoMap Window.....	94
Main ePublisher Window.....	95
Document Manager.....	95
Output Explorer.....	96
Reports.....	97
Reports - printable.....	97
Start Page.....	98
Log Window.....	98
Preview Window.....	98
Document Designer.....	99
Style Designer.....	100
Font Family Picker Window.....	100
Manage Targets Window.....	100
Merge Settings Window.....	101
New ePublisher AutoMap Job Window.....	102
New Project Wizard.....	102
New Project Window (New Project Wizard).....	102
Browse For Folder Window (New Project Wizard).....	103
Source Documents Window (New Project Wizard).....	103

Preferences Window.....	103
General Tab (Preferences Window).....	104
File Mappings Tab (Preferences Window).....	105
Notification Tab (Preferences Window).....	106
Project Settings Window.....	107
File Mappings Tab (Project Settings Window).....	107
General Tab (Project Settings Window).....	108
Input Configurations Tab (Project Settings Window).....	109
Save As Stationery Window.....	112
Script Editor Window.....	113
Target Configuration Window.....	113
Info Tab (Target Configuration Window).....	114
Conditions Tab (Target Configuration Window).....	115
Variables Tab (Target Configuration Window).....	116
Target Settings Tab (Target Configuration Window).....	117
Merge Settings Tab (Target Configuration Window).....	117
Target Selection Window.....	118
User Information Window.....	119
Variables Window.....	120
WebWorks ePublisher Preferences Window.....	120
General Tab (WebWorks ePublisher Preferences Window).....	121
File Mappings Tab (WebWorks ePublisher Preferences Window).....	124
Editor Preferences Tab (WebWorks ePublisher Preferences Window).....	125
Diff Preferences Tab (WebWorks ePublisher Preferences Window).....	125
Log Window Tab (WebWorks ePublisher Preferences Window).....	125
WebWorks Licensing Info Window.....	125
<b>Producing Output from Stationery.....</b>	<b>127</b>
What Makes an ePublisher Project.....	127
ePublisher Projects.....	127
Project Folder Structure.....	127
Source Documents.....	130
Targets.....	130
Stationery.....	131
Creating Projects Based on Stationery.....	132
Working with Source Documents.....	133
Adding Source Documents to Projects.....	133
Opening Source Documents from Document Manager.....	136

Scanning Source Documents.....	138
Scanning and Scanning Options.....	138
Setting Scanning Options.....	139
Scanning Selected Documents.....	140
Scanning All Documents.....	141
Relinking Source Documents.....	142
Removing Source Documents from Projects.....	143
Source Documents Groups.....	144
Organizing Source Documents Using Groups.....	145
Creating Top-Level Groups.....	145
Creating Subgroups.....	146
Renaming Groups.....	147
Rearranging Source Documents in Groups.....	148
Removing Groups.....	149
Working with Targets.....	150
Specifying Active Targets.....	150
Adding Targets to Projects Based on Stationery.....	151
Renaming Targets.....	152
Deleting Targets.....	153
Working with Projects.....	154
Saving Projects.....	154
Opening Existing Projects.....	155
Closing Projects.....	156
Synchronizing Projects with Stationery.....	157
Manifest Files.....	158
Stationery Files.....	159
When to Synchronize.....	159
Automatically Synchronizing ePublisher Express Projects with Stationery....	159
Manually Synchronizing ePublisher Express Projects with Stationery.....	160
Project Information that is not Synchronized.....	161
Deleting Projects.....	161
Generating and Regenerating Output.....	162
Output Generation and Regeneration.....	162
Generating Output.....	163
Regenerating Output.....	165
Generating Output from FrameMaker or Microsoft Word.....	167
Modifying Help System Title Bars.....	168
Viewing Output.....	169

Viewing Output by Automatically Opening Generated Output.....	170
Viewing Output in Output Explorer.....	171
Viewing Output in the Output Folder.....	175
Changing the Location of the Output Folder.....	176
Working with Output Log Files.....	177
Validating Output Using Reports.....	178
Accessibility Reports.....	179
Baggage Files Reports.....	179
Conditions Reports.....	180
Filenames Reports.....	180
Links Reports.....	181
Styles Reports.....	182
Topics Reports.....	182
Images Reports.....	183
Printable Reports.....	183
Configuring Reports.....	183
Generating Reports.....	185
Report Messages.....	187
Accessibility Report Messages.....	187
Baggage Files Report Messages.....	188
Filename Report Messages.....	189
Links Report Messages.....	190
Topics Report Messages.....	193
Images Report Messages.....	194
Merging Top-level Groups (Multivolume Help).....	195
Deploying Output.....	198
Output Deployment.....	198
Creating Output Destinations.....	199
Specifying Output Destinations for Targets.....	201
Deploying Output to Output Destinations.....	202
Working with Target Settings.....	203
Specifying Accessibility Settings.....	204
Specifying Baggage Files Settings.....	206
Baggage files info list.....	206
Copy baggage file dependents.....	209
Index baggage files.....	210
Index external links.....	210
Specifying Company Information.....	211

Specifying File Processing Behavior for Front Matter, Index, and Table of Contents Files.....	212
Specifying Page Breaks Settings.....	213
Specifying Page, Image, and Table File Naming Patterns.....	214
Specifying Index Settings.....	215
Specifying How Links to Files or External URLs Display in Browser Windows....	216
Specifying Unknown File Links Behavior in Reverb.....	217
Specifying Character Encoding for Targets.....	218
Specifying the Language Used by Targets.....	219
Specifying PDF Generation Settings.....	220
Specifying Table of Contents Settings.....	221
Specifying Report Settings.....	224
Specifying Output Format-Specific Settings.....	224
Setting Variables in Projects.....	226
Setting Conditions in Projects.....	228
Setting Cross-References in Projects.....	229
Modifying Cross-Reference Formats in Projects.....	230
Adding Cross-Reference Formats to Projects.....	231
Deleting Cross-Reference Formats from Projects.....	232
File Mappings for Source Documents.....	233
File Mappings.....	233
Modifying File Mappings.....	234
Creating New File Mappings.....	235
Deleting File Mappings.....	237
<b>Scheduling and Integrating Processes with AutoMap.....</b>	<b>239</b>
How ePublisher Supports Automation.....	239
What Is ePublisher AutoMap?.....	239
Benefits of Using ePublisher AutoMap.....	239
Version Control System (VCS) Integration.....	240
Content Management System (CMS) Integration.....	240
Preparing Projects, Stationery, and Source Files.....	241
Starting ePublisher AutoMap.....	241
Setting ePublisher AutoMap Preferences.....	243
Specifying the Job, Staging, and User Formats Folder Locations.....	243
Job Folder.....	243
Staging Folder.....	243
User Formats Folder.....	244

Automatic Scanning for Conditions and Variables.....	245
Keeping or Deleting Temporary Files.....	246
Defining File Mappings.....	247
Defining Output Destinations.....	248
Defining Email Notifications.....	250
Selecting Console Language (English, German, French, and Japanese).....	251
Working with Jobs.....	252
Creating a Project-Based Job.....	253
Creating a Stationery-Based Job.....	254
Duplicating an Existing Job.....	257
Editing an Existing Job.....	259
Scheduling Jobs with Windows Scheduler.....	260
Deleting an Existing Schedule for a Job.....	261
Running an Existing Job.....	262
Viewing a Job Log File.....	263
Canceling a Job.....	264
Deleting an Existing Job.....	265
Using Scripts for Additional Custom Processing.....	266
Writing Scripts.....	266
Working Folder.....	266
Opening and Using the Script Editor.....	267
Scripting Variables.....	268
Scripting Examples.....	270
Show Time and Date Example.....	270
Using Scripting Variables Example.....	271
CVS Version Control Checkout Example.....	272
Using the Command-Line Interface.....	275
Running ePubublisher AutoMap from the Command Line.....	275
CLI Syntax and Reference.....	276
CLI Examples.....	279
Running a Project and Updating the Express project file.....	279
Running a Project and Generating Only One Target.....	279
Running a Project from Scratch and Deploying to a Clean Location.....	279
Running a Project and Deploying to an Alternate Location.....	279
Running a Job Without Sending Notification When Done.....	279
Running a Job and Deploying to a Clean Location.....	280
Running a Job Without Deploying the Content.....	280

<b>Markdown++ Source Documents.....</b>	<b>281</b>
Introduction.....	281
Getting Started with Markdown.....	281
Learning Markdown.....	281
Paragraphs.....	282
Titles.....	285
Headings.....	288
Lists.....	300
Tables.....	313
Blockquotes.....	323
Code Fences.....	329
Code Blocks.....	332
Horizontal Rules.....	335
Block HTML.....	337
Bold, Italic, Strikethrough, Code.....	340
Links.....	345
Images.....	348
Link References.....	349
Inline HTML.....	351
Learning Markdown++.....	352
Markdown++ Basics.....	352
Multiline Tables in Markdown++.....	354
Custom Styles.....	363
Custom Aliases.....	366
Markers in Markdown++.....	368
Conditions.....	372
File Includes.....	377
Variables.....	379
<b>Adobe FrameMaker.....</b>	<b>382</b>
Adobe FrameMaker Formats and Standards.....	382
Standards for Single-Sourcing.....	382
Planning for Importing Elements Across Files.....	383
Paragraph Formats in FrameMaker.....	383
Character Formats in FrameMaker.....	386
Bulleted and Numbered Lists in FrameMaker.....	388
Image Formats and Considerations in FrameMaker.....	388



Table Formats in FrameMaker.....	391
Cross Reference Formats in FrameMaker.....	391
Markers in FrameMaker.....	392
Variables and Conditions in FrameMaker.....	397
Page Layouts in FrameMaker.....	397
Reference Pages, Table of Contents, and Indexes in FrameMaker.....	398
Implementing Online Features in FrameMaker.....	398
Custom Marker Types in FrameMaker.....	399
Paragraph and Character Formats in FrameMaker.....	405
Obtaining and Applying the Latest Adobe FrameMaker Template.....	407
Importing Custom Marker Types in FrameMaker.....	408
Creating Custom Marker Types in FrameMaker.....	409
Creating a Passthrough Marker in FrameMaker.....	410
Creating Cross-References and Links in FrameMaker.....	411
Working with Tables in FrameMaker.....	413
Applying Table Formats in FrameMaker.....	413
Creating Table Header Rows in FrameMaker.....	414
Creating Table Footer Rows in FrameMaker.....	415
Working with Images in FrameMaker.....	416
Inserting Images in FrameMaker.....	417
Creating Image Links in FrameMaker.....	419
Creating Clickable Regions for Image Maps in FrameMaker.....	421
Creating Image Maps for Single Images in FrameMaker.....	421
Creating Image Maps for Composite Images in FrameMaker.....	424
Assigning Image Scales in FrameMaker.....	428
Assigning Image Styles in FrameMaker.....	431
Working with Videos in FrameMaker.....	433
Creating Index Entries in FrameMaker.....	437
Using Variables in FrameMaker.....	440
Importing or Creating Variables in FrameMaker.....	440
Inserting Variables into FrameMaker.....	443
Changing Variable Values in FrameMaker.....	444
Deleting Variables in FrameMaker.....	445
Using Conditions in FrameMaker.....	446
Creating Conditions in FrameMaker.....	447
Applying Conditions in FrameMaker.....	448
Removing Conditions in FrameMaker.....	449
Modifying Conditions in FrameMaker.....	450

Showing and Hiding Conditions in FrameMaker.....	451
Using Passthrough Conditions in FrameMaker.....	452
Deleting Conditions in FrameMaker.....	453
Conditional Output Using Expressions in FrameMaker.....	454
Specifying Output File Names in FrameMaker.....	455
Creating Context-Sensitive Help in FrameMaker.....	457
Context-Sensitive Help in FrameMaker.....	457
Planning for Context-Sensitive Help in FrameMaker.....	458
Specifying Context-Sensitive Help Links in FrameMaker.....	459
Creating Popup Windows in FrameMaker.....	462
Creating Popup Window Links in FrameMaker.....	464
Using Markers to Create Popup Windows in FrameMaker.....	466
Using Paragraph Formats to Create Popup Windows in FrameMaker.....	469
Creating Expand/Collapse Sections (Drop-Down Hotspots) in FrameMaker.....	470
Creating Related Topics in FrameMaker.....	473
Creating See Also Links in FrameMaker.....	476
Creating Meta Tag Keywords in FrameMaker.....	480
Assigning Custom Page Styles in FrameMaker.....	483
Opening Topics in Custom Windows in FrameMaker.....	484
Customizing TOC Entry in FrameMaker.....	486
Customizing Table of Contents Icons in FrameMaker.....	491
Specifying Context Plug-ins in FrameMaker.....	494
Creating Accessible Online Content in FrameMaker.....	498
Accessible Content in FrameMaker.....	498
Accessible Content Navigation in FrameMaker.....	499
Validating Accessible Content in FrameMaker.....	503
Assigning Alternate Text to Images and Image Maps in FrameMaker.....	503
Image and Image Map Alternate Text in FrameMaker.....	503
Assigning Alternate Text to Images in FrameMaker.....	504
Assigning Alternate Text to Image Maps in FrameMaker.....	507
Assigning Long Descriptions to Images in FrameMaker.....	508
Image Long Descriptions in FrameMaker.....	509
Specifying Long Descriptions for Images in FrameMaker.....	510
Using Text in External Files to Assign Long Descriptions to Images in FrameMaker.....	513
Excluding Images from Accessibility Report Checks in FrameMaker.....	516
Assigning Alternate Text (Summaries) to Tables in FrameMaker.....	517
Excluding Tables from Accessibility Report Checks in FrameMaker.....	519

Assigning Alternate Text to Abbreviations in FrameMaker.....	521
Assigning Alternate Text to Acronyms in FrameMaker.....	523
Providing Citations for Quotes in FrameMaker.....	525
Troubleshooting FrameMaker issues.....	528
<b>Microsoft Word.....</b>	<b>532</b>
Microsoft Word Templates and Standards.....	532
Word Standards to Support Single-Sourcing.....	532
Microsoft Word Template File.....	533
Creating a Clean Base Template File.....	533
Paragraph Styles in Word.....	534
Character Styles in Word.....	537
Bulleted and Numbered Lists in Word.....	538
Bulleted Lists in Word.....	539
Numbered Lists in Word.....	539
Image Styles and Considerations in Word.....	539
Table Styles in Word.....	542
Field Codes.....	543
AutoText, AutoCorrect, and User-Defined Hotkeys.....	548
Toolbars and Menus in Word.....	548
Variables and Conditions in Word.....	549
Page Layouts and Sections in Word.....	549
Table of Contents and Index in Word.....	549
Automation with Macros in Word.....	549
Implementing Online Features in Word.....	550
Custom Marker Types in Word.....	550
Paragraph and Character Formats in Word.....	557
Obtaining and Applying the Latest Microsoft Word Template.....	559
Working with the WebWorks Transit Menu for Word.....	560
WebWorks Transit Menu for Word.....	560
Installing the WebWorks Transit Menu for Word.....	560
Running Transit Menu in Secure Environments.....	561
Initializing the WebWorks Transit Menu for Microsoft Word.....	565
Displaying and Hiding the WebWorks Transit Menu in Word.....	567
Creating Custom Marker Types Using the WebWorks Transit Menu in Word.....	568
Creating a Passthrough Marker in Word.....	569
Working with Tables in Word.....	570
Applying Table Styles in Word.....	570

Creating Table Header Rows in Word.....	572
Working with Images in Word.....	573
Inserting Images in Word.....	574
Validating Images in Word.....	575
Creating Image Links in Word.....	577
Creating Clickable Regions for Image Maps in Word.....	578
Creating Image Maps for Single Images in Word.....	579
Creating Image Maps for Composite Images in Word.....	581
Assigning Image Scales in Word.....	583
Assigning Image Styles in Word.....	587
Creating Index Entries in Word.....	588
Using Variables in Word.....	590
Creating Variables in Word.....	591
Inserting Variables into Word.....	592
Changing Variable Values in Word.....	593
Deleting Variables in Word.....	594
Using Conditions in Word.....	595
Creating Conditions in Word.....	596
Applying Conditions in Word.....	597
Validating Conditions in Word.....	598
Removing Conditions in Word.....	600
Modifying Conditions in Word.....	601
Highlighting All Conditions in Word.....	602
Displaying Conditionalized Content with Conflicting Settings in Word.....	603
Using Passthrough Conditions in Word.....	604
Deleting Conditions in Word.....	605
Specifying Output File Names in Word.....	606
Specifying Page Output File Names in Word.....	606
Specifying Image Output File Names in Word.....	608
Creating Context-Sensitive Help in Word.....	612
Context-Sensitive Help in Word.....	612
Planning for Context-Sensitive Help in Word.....	613
Specifying Context-Sensitive Help Links in Word.....	614
Creating Popup Windows in Word.....	617
Creating Popup Window Links in Word.....	618
Using Markers to Create Popup Windows in Word.....	622
Using Paragraph Styles to Create Popup Windows in Word.....	624
Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word.....	625

Creating Related Topics in Word.....	628
Creating Links to PDF in Word.....	631
Creating See Also Links in Word.....	632
Creating Meta Tag Keywords in Word.....	636
Assigning Custom Page Styles in Word.....	638
Creating What's This (Field-Level) Help in Word.....	640
Opening Topics in Custom Windows in Word.....	642
Customizing TOC Entry in Word.....	644
Customizing Table of Contents Icons in Word.....	647
Specifying Context Plug-ins in Word.....	650
Creating Accessible Online Content in Word.....	654
Accessible Content in Word.....	654
Accessible Content Navigation in Word.....	655
Validating Accessible Content in Word.....	659
Assigning Alternate Text to Images and Image Maps in Word.....	659
Image and Image Map Alternate Text in Word.....	659
Assigning Alternate Text to Images in Word.....	660
Assigning Alternate Text to Image Maps in Word.....	661
Assigning Long Descriptions to Images in Word.....	662
Image Long Descriptions.....	662
Specifying Long Descriptions for Images in Word.....	664
Using Text in External Files to Assign Long Descriptions to Images in Word.....	667
Excluding Images from Accessibility Report Checks in Word.....	670
Assigning Alternate Text (Summaries) to Tables in Word.....	674
Excluding Tables from Accessibility Report Checks in Word.....	675
Assigning Alternate Text to Abbreviations in Word.....	677
Assigning Alternate Text to Acronyms in Word.....	679
Providing Citations for Quotes in Word.....	681
Troubleshooting Word issues.....	682
Word warning dialogs that interrupt conversions.....	683
<b>DITA - XML.....</b>	<b>687</b>
DITA Usage Standards.....	687
DITA Standards for Single-Sourcing.....	687
Mapping DITA Classes to ePublisher Styles.....	688
Defining Online Features with DITA.....	690
Configuring DITA Open Toolkit Version.....	691

Customizing the DITA DTD.....	692
DITA Specialization.....	693
DITA Support.....	693
Keyref elements.....	693
Conref extensions.....	694
Using Ditaval files in DITA.....	694
Using Passthrough outputclass in DITA.....	695
Embedding a Video in DITA Source Documents.....	696
Embedding a Video file.....	696
Linking to a Youtube Video.....	697
Creating Context-Sensitive Help in DITA Source Documents.....	698
Context-Sensitive Help.....	698
Map Files.....	699
Planning for Context-Sensitive Help.....	700
Topic ID and File Name Requirements.....	701
Output Formats that support Creating Context-Sensitive Help Links In DITA Source Documents.....	701
Specifying Context-Sensitive Help Links in DITA Source Documents.....	703
Creating Hyperlinks in DITA Source Documents.....	704
Creating Popups in DITA Source Documents.....	705
Popups.....	705
Requirements for Creating Popups in DITA Source Documents.....	706
Creating Popup Links in DITA Source Documents.....	707
Using Paragraph Styles to Create Popups in DITA Source Documents.....	707
Creating Related Topics in DITA Source Documents.....	708
Related Topics.....	708
Requirements for Creating Related Topics Links in DITA Source Documents....	709
Specifying Related Topics Links in DITA Source Documents.....	710
Creating See Also Links in DITA Source Documents.....	711
See Also Links.....	711
Requirements for Creating See Also Links in DITA Source Documents.....	712
Specifying See Also Links in DITA Source Documents.....	713
Using the data element.....	713
Assigning Custom Page Styles to Pages in DITA Source Documents.....	713
Page Styles.....	714
Requirements for Specifying Custom Page Styles for Pages in DITA Source Documents.....	714
Specifying Custom Page Styles for Pages in DITA Source Documents.....	715

Using Custom Graphic Styles for Images in DITA Source Documents.....	716
Assigning Graphic Styles.....	716
Default Graphic Styles for DITA.....	717
Customizing TOC Entry in DITA.....	718
Customizing Table of Contents Icons for Topics in DITA Source Documents Using Legacy Outputs.....	720
Requirements for Specifying Custom Table of Contents Icons in DITA Source Documents.....	721
Specifying Custom Table of Contents Icons in DITA Source Documents.....	722
Using markopen and markclose.....	725
Configuring markopen and markclose entries for dropdowns in ePublisher.....	726
Troubleshooting DITA issues.....	728
<b>Markdown++ Output Format.....</b>	<b>731</b>
Introduction.....	731
Setting up a Markdown++ Target in ePublisher.....	731
Basic Workflow for Adding and Converting Documents.....	732
Configuration Tasks for Markdown++.....	733
Markdown++ Tables and Layout.....	736
<b>WebWorks Reverb 2.0.....</b>	<b>739</b>
Choosing a Skin.....	740
Using SASS To Customize Reverb Interface.....	744
Using Custom SASS files in Reverb Projects.....	745
Previewing Reverb Output.....	745
Delivering Reverb Output.....	746
Top-Level Groups in Reverb.....	746
Searching Output.....	747
Using Baggage Files.....	747
Indexing Baggage Files and External URLs.....	748
Using Tidy for Indexing HTML Pages.....	748
Assigning Relevance Weight to Your Source Documents Styles.....	750
Assigning Relevance Weight to Your HTML and PDF Baggage Files.....	751
Search Highlighting in Baggage Files.....	753
Searching with URL Method.....	754
TOC or Index with URL Method.....	754
Launching Context Sensitive Help for WebWorks Reverb 2.0.....	754
End-user requirements in WebWorks Reverb 2.0.....	755
Analytics Event Tracking in Reverb 2.0.....	755

'Was This Helpful?' Buttons.....	756
'Was This Search Helpful?' Buttons.....	756
Document Last Modified Date in Reverb.....	757
Drop-down Expand/Collapse All Toggle Button.....	758
Google Translate Button.....	759
Customizable Header and Footer.....	760
Custom TOC Menu Items.....	761
Customizing a Bullet Icon using Font Awesome.....	762
Using the url_maps.xml reference file.....	763
<b>PDF - XSL-FO.....</b>	<b>764</b>
Why use PDF - XSL-FO output format?.....	764
PDF-XSL-FO Page Regions.....	764
PDF XSL-FO Font Inclusions.....	766
<b>Dynamic HTML.....</b>	<b>769</b>
Dynamic HTML Output Viewer.....	770
Delivering Dynamic HTML.....	771
<b>ePUB.....</b>	<b>773</b>
ePUB Platforms.....	773
ePUB Considerations.....	773
Meta Data.....	774
Book Title and ID.....	774
Long Content.....	774
Page Styles.....	774
Tables.....	774
Cover.....	775
Syncing with Apple iPad.....	775
<b>Eclipse Help.....</b>	<b>777</b>
Eclipse Help Viewer.....	777
Delivering Eclipse Help.....	778
<b>HTML Help.....</b>	<b>779</b>
Benefits of Microsoft HTML Help.....	779
Restrictions and Requirements for Microsoft HTML Help.....	780
HTML Help Viewer.....	780
Toolbar Pane in HTML Help.....	781



Navigation Pane in HTML Help.....	784
Topic Pane in HTML Help.....	784
Topic Only View in HTML Help.....	784
HTML Help Workshop.....	785
HTML Help Project File (.hhp).....	785
HTML Help Contents File (.hhc).....	785
HTML Help Index File (.hhk).....	785
HTML Help Mapping File (.h).....	786
Delivering HTML Help.....	786
<b>Oracle Help.....</b>	<b>787</b>
Oracle Help Viewer.....	788
Oracle Help Files.....	788
Helpset .hs File in Oracle Help.....	788
Control .xml Files.....	789
Full Text Search .idx Index File.....	789
Manifest .mft File.....	789
Delivering Oracle Help.....	789
<b>Sun JavaHelp.....</b>	<b>790</b>
Sun JavaHelp Files.....	790
Helpset .hs File in Sun JavaHelp.....	790
Contents toc.xml File in Sun JavaHelp.....	791
Index ix.xml File in Sun JavaHelp.....	792
Map .jhm File in Sun JavaHelp.....	792
Delivering Sun JavaHelp.....	792
<b>WebWorks Help.....</b>	<b>794</b>
The Frameset View in WebWorks Help.....	794
Navigation Pane in WebWorks Help.....	795
Toolbar Pane in WebWorks Help.....	796
Topic Pane in WebWorks Help.....	797
Topic Only View in WebWorks Help.....	797
WebWorks Help Output Files.....	798
Delivering WebWorks Help.....	798
Searching WebWorks Help.....	798
<b>Designing, Deploying, and Managing Stationery.....</b>	<b>800</b>
Understanding Stationery.....	800

Stationery Components.....	801
Understanding Stationery Synchronization.....	802
Designing Stationery.....	802
Creating a Stationery Design Project.....	802
Adding Output Formats to Your Stationery Design Project.....	804
Adding a Target to Your Stationery Design Project.....	805
Selecting an Active Target in Your Stationery Design Project.....	806
Updating a Project to Include All Styles.....	807
Understanding Style Designer.....	808
Modifying Output with CSS Properties and Attributes.....	808
Understanding the CSS Box Model.....	809
Inheriting Style Properties and Options.....	810
Understanding Options in Style Designer.....	810
Organizing and Managing Styles.....	811
Previewing the Output from a Source File.....	812
Defining New Pages (Page Breaks).....	813
Defining TOCs and Mini-TOCs.....	814
Defining the Table of Contents Structure (Levels).....	814
Generating and Naming the Table of Contents File.....	817
Defining the Table of Contents from an Irregular Heading Hierarchy.....	819
Understanding the Table of Contents and Merge Settings.....	822
Defining the Icon for a Table of Contents Entry.....	822
Creating a Miniature Table of Contents.....	823
Modifying the Appearance of Mini-TOC Entries.....	824
Modifying the Appearance of Paragraphs.....	826
The Prototype Style for Paragraphs.....	827
Setting the Background Color of a Paragraph.....	827
Setting the Border Style and Color of a Paragraph.....	828
Setting the Font for a Paragraph.....	829
Setting the Width, Height, and Positioning of a Paragraph.....	830
Adjusting the Space Around a Paragraph.....	831
Setting the Text Color and Other Characteristics of a Paragraph.....	832
Modifying Paragraphs for Bidirectional Languages.....	833
Disabling Autonumbering in Output.....	834
Defining the Appearance of Notes, Tips, Cautions, and Warnings.....	835
Defining the Appearance of Bulleted Lists.....	835
Defining the Appearance of Numbered Lists.....	837
Fixing Paragraph Indentation Including Hanging Indent.....	838

Modifying the Appearance of Characters.....	840
The Prototype Style for Characters.....	840
Setting the Background Color of a Character.....	840
Setting the Border Style and Color of Characters.....	841
Setting the Font for a Character.....	842
Adjusting the Space Around Characters.....	843
Setting the Color and Other Characteristics of Characters.....	844
Modifying Characters for Bidirectional Languages.....	845
Defining the Appearance of Tables.....	846
The Prototype Style for Tables.....	846
Setting the Background Color or Image of a Table.....	847
Setting the Border Style and Color of a Table.....	848
Setting the Width and Height of a Table.....	849
Setting the Vertical and Horizontal Alignment within a Table.....	850
Adjusting the Space Within and Around a Table.....	851
Modifying Header, Footer, and Body Rows of a Table.....	852
Modifying Cells of a Table.....	853
Defining the Appearance of Images.....	854
Supported Image Formats.....	854
Image Quality and Processing.....	855
The Prototype Style for Images.....	855
Defining Graphic Styles.....	855
Setting the Border Style and Color of an Image.....	856
Modifying the Width, Height, and Positioning of an Image.....	857
Adjusting the Space Around Images.....	858
Using Thumbnails for Images.....	859
Setting the Maximum Width and Height for Images.....	860
Modifying Image Size by Scale.....	861
Modifying Image Resolution.....	862
Setting Color Bit Depth.....	863
Choosing an Image File Format and Quality Level.....	864
Creating Grayscale Images.....	865
Setting Transparency for .gif and .png Images.....	866
Defining the Appearance of Pages.....	867
The Prototype Style for Pages.....	867
Displaying Company Logo and Information on a Page.....	867
Modifying the Appearance of the Company Information.....	869
Setting the Background Color or Image of a Page.....	871

Setting the Border Style and Color of a Page.....	872
Adjusting the Space Around a Page.....	873
Using a Custom CSS to Modify the Appearance of Content.....	874
Modifying the Location and Separators of Breadcrumbs.....	875
Modifying the Appearance of Breadcrumbs.....	876
Choosing the Location of Navigation Browse Buttons.....	877
Modifying the Navigation Browse Buttons.....	879
Associating a Page with a Page Style.....	881
Defining the Appearance of Links.....	882
Saving a Snapshot (Backup Copy) of Your Project.....	883
Defining Marker Types.....	885
Defining File Names.....	890
Specifying File Names for Pages Using Page Naming Patterns.....	890
Specifying File Names for Images Using Graphic Naming Patterns.....	894
Using Markers to Define File Names.....	899
Defining Context-Sensitive Help Links.....	900
Defining Filename Markers for Context-Sensitive Help Links.....	901
Defining TopicAlias Markers for Context-Sensitive Help Links.....	903
Defining Expand/Collapse Sections (Drop-Down Hotspots).....	905
Using Styles and Markers for Expand/Collapse Sections.....	905
Modifying Images for Expand/Collapse Sections.....	906
Defining Popup Windows.....	908
Using Marker Styles to Create Popup Windows.....	908
Using Paragraph Styles To Create Popup Windows.....	909
Assigning a Page Style to Popup Windows.....	910
Defining Related Topics.....	911
Using a Paragraph Style for Related Topics Lists.....	912
Defining See Also Links.....	913
Enabling See Also Functionality.....	914
Modifying the Appearance of the See Also Button.....	915
Define the Default Settings for Each Target.....	917
Defining the Default Index Settings.....	917
Defining the Default Processing of Variables.....	919
Defining the Default Processing of Conditions.....	920
Defining the Default Processing of Cross References.....	921
Defining Default PDF Generation Settings.....	922
Defining the Accessibility Report to Validate Content.....	924
Defining Other Reporting Options.....	926

Saving and Testing Stationery.....	927
Backing Up Your Stationery Design Project, Stationery, and Projects.....	928
Deploying Stationery.....	929
Managing and Updating Stationery.....	931
<b>Target Settings Reference.....</b>	<b>934</b>
Accessibility Settings.....	934
Accessibility Report Settings.....	935
Analytics Settings.....	936
Baggage Files Settings.....	936
Baggage Files Report Settings.....	937
Company Information Settings.....	937
Conditions Report Settings.....	938
Cover Settings (eBook - ePub 2.0).....	938
Eclipse Settings.....	939
ePUB Settings (eBook - ePub 2.0).....	940
File Processing Settings.....	940
Filenames Report Settings.....	941
Files Settings.....	941
Footer Settings.....	954
Header Settings.....	955
HTML Help Settings.....	955
Images Report Settings.....	956
Index Settings.....	956
JavaHelp Settings.....	956
Links Settings.....	957
Links Report Settings.....	959
Locale Settings.....	961
Menu Settings.....	962
Oracle Help Settings.....	963
Page Settings.....	963
PDF Settings.....	964
Result Options Settings (PDF - XSL-FO).....	968
Search Settings.....	968
Social Settings.....	969
Styles Settings (PDF - XSL-FO).....	969
Styles Report Settings.....	971
Table of Contents Settings.....	971

Title Page Settings (PDF - XSL-FO).....	972
Toolbar Settings.....	975
Topics Report Settings.....	976
WebWorks Help Settings.....	976
WebWorks Reverb Settings.....	978
WebWorks Reverb 2.0 Settings.....	982
<b>Style Designer Reference.....</b>	<b>985</b>
Advanced Properties.....	985
Aural Properties.....	990
Background Properties.....	991
Body Properties.....	996
Body Background Properties (Tables).....	998
Border Properties.....	1008
Bullet Properties.....	1009
Font Properties.....	1012
Footer Properties (Tables).....	1012
Footer Background Properties (Tables).....	1014
Header Properties (Tables).....	1024
Header Background Properties (Tables).....	1026
HTML (Layout) Properties.....	1036
Margin Properties.....	1046
Markdown++ Properties.....	1047
Master Page Properties (Pages).....	1050
Navigation Properties (Pages).....	1052
Padding Properties.....	1053
Pagination Properties.....	1055
Table Properties (Tables).....	1059
Text Properties.....	1063
Markdown++ Options.....	1070
Paragraph Styles Options.....	1072
Character Styles Options.....	1085
Table Styles Options.....	1088
Page Styles Options.....	1091
Page Styles Options (PDF - XSL-FO).....	1093
Graphic Styles Options.....	1096
Marker Styles Options.....	1101

<b>Customizing WebWorks Reverb 2.0.....</b>	<b>1103</b>
Changing the Appearance of WebWorks Reverb 2.0.....	1103
Using SASS To Customize WebWorks Reverb 2.0.....	1104
SASS Variables In WebWorks Reverb 2.0.....	1107
Migrating SASS Overrides To Newer Format Versions.....	1110
Layout Colors In WebWorks Reverb 2.0 Skins.....	1111
Target Settings for WebWorks Reverb 2.0.....	1129
WebWorks Reverb 2.0 Target Settings.....	1130
WebWorks Reverb 2.0 Toolbar Target Settings.....	1131
WebWorks Reverb 2.0 Analytics Target Settings.....	1133
WebWorks Reverb 2.0 Menu Target Settings.....	1134
WebWorks Reverb 2.0 Page Target Settings.....	1135
WebWorks Reverb 2.0 Footer Target Settings.....	1137
Social Target Settings.....	1138
Selecting an Alternate Skin for WebWorks Reverb 2.0.....	1139
Customizing the Top-Level Entry File.....	1140
Specifying the Entry Page Name.....	1140
Specifying the Entry Page Style.....	1141
Customizing TOC Menu Item Display.....	1141
Specifying TOC Item CSS Class.....	1141
Customizing the Splash Page in WebWorks Reverb 2.0.....	1141
Specifying the Splash Page Style.....	1141
Replacing the Splash Image.....	1141
Modifying the Splash Page.....	1143
Removing the Splash Page.....	1143
Using Context-Sensitive Help in WebWorks Reverb 2.0.....	1144
Mapping Files in WebWorks Reverb 2.0.....	1144
Opening Context-Sensitive Help in WebWorks Reverb 2.0 using Standard URLs.....	1144
URL Commands Support by WebWorks Reverb 2.0.....	1145
Opening Context-Sensitive Help in WebWorks Reverb 2.0 using JavaScript...	1146
Opening Context-Sensitive Help in WebWorks Reverb 2.0 using the WebWorks Help API.....	1146
Configuring Client-Side Search for Reverb 2.0.....	1147
Configuring Synonyms.....	1147
Searching WebWorks Reverb 2.0 - URL Method.....	1148
Incorporating Google Analytics for Your Reverb 2.0 Files.....	1148
Configuring Commenting and End-User Feedback for Reverb 2.0.....	1149

Customizing Icons in Your Reverb 2.0 Output Using Font Awesome.....	1149
Incorporating Web Fonts in Your Reverb 2.0 Output.....	1150
Steps to Create Your First Disqus Site.....	1152
'Was This Helpful?' Buttons.....	1152
Dropdown Collapse/Expand All Toggle Button.....	1152
Document Last Modified Date/Publish Date.....	1153
Customizing Related Topics.....	1154
Customizing Related Topic Styling/Appearance.....	1154
Enabling/Disabling Related Topic Dropdown Behavior.....	1154
<b>Customizing PDF - XSL-FO.....</b>	<b>1155</b>
Custom Header and Footers.....	1155
Document Last Modified Date/Publish Date.....	1157
Page Template Customizations.....	1158
<b>Customizing Dynamic HTML.....</b>	<b>1159</b>
Using SASS to change the Appearance of Dynamic HTML.....	1159
Modifying the Appearance of the Table of Contents in Dynamic HTML.....	1159
Modifying the Appearance of the Index in Dynamic HTML.....	1161
Other Changes to Text in the TOC and Index in Dynamic HTML.....	1163
Document Last Modified Date/Publish Date.....	1164
<b>Customizing Eclipse Help.....</b>	<b>1166</b>
Using Markers to Specify Context Plug-ins in Eclipse Help.....	1166
Using Markers to Specify Topic Descriptions for Context-Sensitive Help Topics in Eclipse Help.....	1167
<b>Customizing Oracle Help and Sun JavaHelp.....</b>	<b>1169</b>
Defining the Navigation Pane in Oracle Help.....	1169
Using Custom Windows in Oracle Help.....	1170
Defining the Navigation Pane in Sun JavaHelp.....	1171
Using Context-Sensitive Help in Oracle Help and Sun JavaHelp.....	1172
Mapping Files in Oracle Help and Sun JavaHelp.....	1173
Testing Context-Sensitive Oracle Help and Sun JavaHelp.....	1174
<b>Customizing WebWorks Help.....</b>	<b>1175</b>
Renaming the Top-Level Entry File.....	1175
Selecting a Theme.....	1176
Customizing the Splash Page in WebWorks Help.....	1177
Replacing the Splash Image.....	1177



Removing the Splash Page.....	1178
Customizing the Toolbar in WebWorks Help.....	1179
Adding and Removing Toolbar Buttons in WebWorks Help.....	1179
Replacing the Toolbar Buttons in WebWorks Help.....	1181
Changing the Background Color of the Toolbar.....	1183
Customizing the Navigation Pane in WebWorks Help.....	1184
Setting the Initial Width of the WebWorks Help Navigation Pane.....	1185
Controlling the Navigation Pane Hover Text Appearance.....	1186
Changing the Font Color on the Navigation Pane Tabs in WebWorks Help.....	1189
Using Custom Icons on the Contents Tab in WebWorks Help.....	1191
Modifying the Appearance of the Search Message in WebWorks Help.....	1192
Modifying the Search Ranking.....	1195
Modifying the Search Highlighting.....	1198
Synonyms.....	1198
Minimum word length & common words.....	1199
Using Context-Sensitive Help in WebWorks Help.....	1201
Mapping Files in WebWorks Help.....	1201
Opening Context-Sensitive Help in WebWorks Help using Standard URLs.....	1201
Opening Context-Sensitive Help with the WebWorks Help API.....	1202
Opening Context-Sensitive Help with the Javascript API.....	1203
<b>Advanced Format and Target Customizations.....</b>	<b>1204</b>
Understanding Customized Processing.....	1204
Format and Target Overrides.....	1205
Creating Format Overrides.....	1205
Creating Target Overrides.....	1208
Managing Overrides.....	1210
Customizing Page Templates (*.asp).....	1212
Page Templates Reference.....	1213
Namespace and Attributes.....	1213
Using ePublisher Style Variables in Page Templates.....	1220
Using ePublisher Project Variables in Page Templates.....	1221
Using Markers in Page Templates.....	1222
<b>ePublisher Pipeline and Transforms.....</b>	<b>1223</b>
Terminology.....	1223
Processing Workflow.....	1224
Transformation Process.....	1225

Adapters Transform Source Documents to WIF.....	1225
WebWorks Intermediate Format (WIF).....	1226
Processing Files by Type.....	1226
Identifying Files to Process.....	1228
TOC Processing Example.....	1229
Stationery, Projects, and Overrides.....	1229
File Locations.....	1229
File Processing.....	1231
ePublisher File Types.....	1232
Format Trait Info (*.fti) Files.....	1232
format.wwfmt Files.....	1233
files.info Files.....	1233
Stationery Design Project .wep File.....	1233
Project .wrp File.....	1234
Stationery .wxsp File.....	1234
XSL Match Templates.....	1234
Root Match Templates.....	1234
Root Match Templates in ePublisher.....	1235
Extension Objects.....	1235
<b>Introduction.....</b>	<b>1239</b>
Audience.....	1239
Help.....	1239
Conventions.....	1239
Formatting.....	1239
Terminology.....	1240
Organization.....	1241
About XML and XSL.....	1241
<b>Architectural Overview.....</b>	<b>1242</b>
Real World Example.....	1243
<b>File Reference.....</b>	<b>1244</b>
File Locations.....	1244
File Processing.....	1246
What This Means For The User.....	1247
<b>File Types.....</b>	<b>1247</b>
Format Trait Info (*.fti).....	1247

Explanation.....	1247
Components.....	1248
Relationships.....	1250
format.wwfmt.....	1251
Explanation.....	1251
Components.....	1251
Relationships.....	1254
files.info.....	1254
Explanation.....	1254
Components.....	1254
Relationships.....	1254
Designer Project File (.wep).....	1254
Stationery File (.wsxp).....	1254
Express Project File (.wrp).....	1255
<b>XSL Match Templates.....</b>	<b>1255</b>
Root Match Templates.....	1255
Root Match Templates in ePublisher Designer.....	1256
Real Life Example.....	1257
<b>Extension Objects.....</b>	<b>1257</b>
<b>Output Customizations.....</b>	<b>1257</b>
<b>Transform Overrides.....</b>	<b>1258</b>
Creating Transform Overrides.....	1258
Information about Overriding files.....	1258
<b>XSLT Reference.....</b>	<b>1261</b>
XSLT Documentation.....	1261
Good to Know.....	1262
Using Extension Objects.....	1262
General XSL Extensions.....	1262
Microsoft Extensions.....	1263
Using ePublisher XSLT Extensions.....	1264
<b>ePublisher Platform XSLT Extensions.....</b>	<b>1265</b>
Class Documentation.....	1268
Adapter.....	1268
void AddToPDFPageNumberOffset (int addToPageNumberOffset).....	1269

bool GeneratePDF (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string pdfJobSettings, string pdfFilePath).....	1271
bool GeneratePDFWithSaveAs (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string pdfJobSettings, string pdfFilePath).....	1273
bool GeneratePostScriptForImage (object input, string postScriptPath).....	1275
long GeneratePostScriptForPDF (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string postScriptFilePath).....	1276
void SetPDFPageNumberOffset (int pageNumberOffset).....	1278
bool TemporaryLicense (string toolAdapterName).....	1279
AdapterConfiguration.....	1280
string GetValue (string name).....	1280
string GetValue (string name, string defaultValue).....	1281
DateTimeUtilities.....	1282
string GetNow (string format).....	1282
string GetGenerateStart (string format).....	1284
string GetFileCreated (string filepath, string format).....	1285
string GetFileLastModified (string filepath, string format).....	1286
string GetFromDateTimeString (string dateTime, string inputFormat, string outputFormat).....	1287
Environment.....	1288
string ApplicationBaseHelpURI ().....	1289
string CurrentUILocale ().....	1290
long GetTotalMemory ().....	1290
long GetTotalMemory (bool forceFullCollection).....	1290
string HTMLHelpWorkshopPath ().....	1291
string JavaBits ().....	1291
string JavaHome ().....	1291
string JavaVersion ().....	1291
string JDKBits ().....	1292
string JDKHome ().....	1292
string JDKVersion ().....	1292
string JREBits ().....	1292
string JREHome ().....	1293
string JREVersion ().....	1293

bool RequestedPipeline (string pipelineName).....	1293
Exec.....	1294
XPathNodeIterator Execute (string commandLine).....	1297
XPathNodeIterator ExecuteCommand (string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20]).....	1298
XPathNodeIterator ExecuteCommandNoReturn (string command).....	1299
XPathNodeIterator ExecuteCommandInDirectory (string directoryPath, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20]).....	1300
XPathNodeIterator ExecuteCommandInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20]).....	1301
XPathNodeIterator ExecuteCommandWithTimeout (long timeoutInSeconds, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20]).....	1302
XPathNodeIterator ExecuteInDirectory (string directoryPath, string commandLine).....	1303
XPathNodeIterator ExecuteInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string commandLine).....	1304
XPathNodeIterator ExecuteProgramWithArguments (string program, string arguments).....	1305
XPathNodeIterator ExecuteProgramWithArgumentsInDirectory (string directoryPath, string program, string arguments).....	1306
XPathNodeIterator ExecuteProgramWithArgumentsInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string program, string arguments)..	1307
XPathNodeIterator ExecuteProgramWithArgumentsWithTimeout (long timeoutInSeconds, string program, string arguments).....	1308

XPathNodeIterator ExecuteWithTimeout (long timeoutInSeconds, string commandLine).....	1309
ExecPython.....	1310
XPathNodeIterator ExecutePyScriptInCommandLine (string commandLine)...	1311
XPathNodeIterator ExecPyScript (string pyScriptPath [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19]).....	1312
XPathNodeIterator ExecutePyScriptInDirectoryInCommandLine (string directoryPath, string commandLine).....	1313
XPathNodeIterator ExecPyScriptInDirectory (string directoryPath, string pyScriptPath [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19])...	1314
Sass.....	1315
XPathNodeIterator SassToCss (string inputSassFilePath, string outputCssFilePath).....	1315
void ReplaceAllVariablesInFile (string inputSassFilePath, object replacements).....	1316
ExslDocument.....	1317
void Document (object input, string path [, string encoding, string method, string version, string indent, string omit_xml_declaration, string standalone, string doctype_public, string doctype_system, string cdata_section_elements, string media_type]).....	1318
XPathNodeIterator LoadXMLWithoutResolver (string uriAsString [, bool preserveSpace]).....	1319
XPathNodeIterator LoadXMLWithResolver (string uriAsString [, bool preserveSpace]).....	1321
XPathNodeIterator MakeEmptyElement (object input).....	1322
Files.....	1323
bool UpToDate (string path, string projectChecksum, string groupID, string documentID, string actionChecksum).....	1324
FileSystem.....	1326
bool AppendFileWithFile (string targetPath, string sourcePath).....	1329
bool ChecksumUpToDate (string path, string checksum).....	1331
string Combine (string path, string component1 [, string component2, string component3, string component4, string component5, string component6, string component7, string component8, string component9, string component10]).	1333

XPathNodeIterator CopyDirectoryFiles (string sourceDirectoryPath, string destinationDirectoryPath).....	1334
XPathNodeIterator CopyFile (string sourcePath, string destinationPath).....	1336
bool CreateDirectory (string path).....	1338
void DeleteDirectory (string path).....	1340
void DeleteFile (string path).....	1342
bool DirectoryExists (string path).....	1344
bool Exists (string path).....	1346
bool FileExists (string path).....	1348
bool FilesEqual (string alphaPath, string betaPath).....	1350
string GetAbsoluteFrom (string relativePath, string referencePath).....	1352
string GetBaseName (string path).....	1354
string GetChecksum (string path).....	1355
string GetDirectoryName (string path).....	1357
string GetExtension (string path).....	1359
string GetFileName (string path).....	1361
string GetFileNameWithoutExtension (string path).....	1363
XPathNodeIterator GetFiles (string path).....	1365
string GetLongPathName (string path).....	1367
XPathNodeIterator GetRelativeFiles (string path).....	1369
string GetRelativeTo (string path, string anchorPath).....	1371
string GetShortPathName (string path).....	1373
string GetTempFileName ().....	1375
string GetTempPath ().....	1375
string GetWithExtensionReplaced (string path, string extension).....	1375
string MakeValidFileName (string fileNameSeed).....	1377
void TranslateFileToEncoding (string sourceFilePath, string sourceFileEncodingName, string destinationFilePath, string destinationFileEncodingName).....	1378
Fonts.....	1380
bool UnicodeFont (string fontFamily).....	1380
Imaging.....	1381
XPathNodeIterator GetInfo (string imageFilePath).....	1382
void MapPDFLinks (object fileTable, string fileToFix, string fileToWrite, string originalFilePath, string outputFilePath, bool useAbsPath).....	1384
bool MergePDFs (object sourceFileList, string targetFilePath).....	1386
bool MergePDFs (object sourceFileList, object fileTable, string targetFilePath).....	1388
bool PostScriptToPDF (string postScriptFilePath, string pdfJobSettings, string pdfFilePath).....	1390

XPathNodeIterator RasterizePostScript (string postScriptFilePath, int renderHorizontalDPI, int renderVerticalDPI, int renderWidth, int renderHeight, string targetImageFormatAsString, int targetImageColorDepth, bool targetImageGrayscale, bool targetImageTransparent, bool targetImageInterlaced, int targetImageQuality, string targetFilePath).....	1392
XPathNodeIterator Transform (string inputImageFilePath, string outputImageFormat, int outputImageWidth, int outputImageHeight, string outputImageFilePath).....	1393
XPathNodeIterator Transform (string inputImageFilePath, string outputImageFormatAsString, int Choice_outputImageQuality_outputImageWidth, int Choice_outputImageWidth_outputImageHeight, string Choice_outputImageHeight_outputImageFilePath, string Choice_outputImageFilePath_outputResolution).....	1394
XPathNodeIterator Transform (string inputImageFilePath, string outputImageFormatAsString, int outputImageQuality, int outputImageWidth, int outputImageHeight, string outputImageFilePath, int outputResolution).....	1395
Log.....	1397
void Error (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10]).....	1398
void Message (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10]).....	1399
void Warning (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10]).....	1400
MultiSearchReplaceExtension.....	1401
void ReplaceAllInFile (string inputEncodingAsString, string inputFilePath, string outputFilePath, object replacements).....	1401
void ReplaceAllInFile (string inputEncodingAsString, string inputFilePath, string outputEncodingAsString, string outputFilePath, object replacements).....	1403
string ReplaceAllInString (string input, object replacements).....	1404
NodeSet.....	1405
XPathNodeIterator FirstUnique (object input, string attributeLocalName).....	1405
XPathNodeIterator FirstUniqueWithNamespace (object input, string attributeLocalName, string attributeNamespaceURI).....	1408
XPathNodeIterator LastUnique (object input, string attributeLocalName).....	1410
XPathNodeIterator LastUniqueWithNamespace (object input, string attributeLocalName, string attributeNamespaceURI).....	1412
Progress.....	1414
bool Abort ().....	1415
void Cancel ().....	1415
void End ().....	1415



void QueueAlert (string message).....	1415
void Retry ().....	1416
void SetStatus (string message).....	1416
void Start (int totalSubSteps).....	1417
Project.....	1418
bool DocumentExtension (string extension).....	1421
bool GetConditionIsPassThrough (string conditionName).....	1422
string GetConfigurationChangeID ().....	1423
XPathNodeIterator GetContextRule (string ruleTypeAsString, string ruleName, string documentID, string uniqueID).....	1423
string GetDocumentDataDirectoryPath (string documentID).....	1424
string GetDocumentGroupPath (string documentID).....	1425
string GetDocumentID (string documentPath [, string groupID]).....	1426
string GetDocumentPath (string documentID).....	1427
string GetDocumentsToGenerateChecksum ().....	1428
string GetFormatID ().....	1428
string GetFormatName ().....	1428
string GetFormatSetting (string name).....	1428
string GetFormatSetting (string name, string defaultValue).....	1429
string GetGroupDataDirectoryPath (string groupID).....	1430
string GetGroupName (string groupID).....	1431
XPathNodeIterator GetOverrideRule (string ruleTypeAsString, string ruleName, string documentID, string uniqueID).....	1432
string GetProjectDataDirectoryPath ().....	1433
string GetProjectDirectoryPath ().....	1433
long GetProjectDocumentsCount ().....	1433
string GetProjectFilesDirectoryPath ().....	1433
string GetProjectFormatDirectoryPath ().....	1434
string GetProjectName ().....	1434
string GetProjectReportsDirectoryPath ().....	1434
string GetProjectTargetName ().....	1434
string GetProjectTargetOverrideDirectoryPath ().....	1434
XPathNodeIterator GetRule (string ruleTypeAsString, string ruleName).....	1435
string GetTargetDataDirectoryPath ().....	1436
string GetTargetFilesInfoPath (string targetIDAsString).....	1436
string GetTargetOutputDirectoryPath ().....	1437
string GetTargetReportsDirectoryPath ().....	1437
StageInfo.....	1437

string Get (string param_key).....	1438
void Set (string param_key, string param_value).....	1439
StringUtilities.....	1440
string CSSClassName (string styleName).....	1442
string DecodeURI (string value).....	1443
string DecodeURIComponent (string value).....	1444
string EclipseId (string identifier).....	1445
string EncodeURI (string value).....	1446
string EncodeURIComponent (string value).....	1447
string EscapeForXMLAttribute (string value).....	1448
string Format (string format, string argument1 [, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10]).....	1449
string FromFile (string sourceFilePath, string sourceFileEncodingName).....	1450
string JavaScriptEncoding (string value).....	1451
bool MatchExpression (string input, string matchExpressionAsString).....	1452
string MatchExpressionValue (string input, string matchExpressionAsString).	1454
string MD5Checksum (string value).....	1456
string NCNAME (string identifier).....	1457
string NormalizeQuotes (string value).....	1458
string OEBCClassName (string styleName).....	1459
string Replace (string input, string search, string replacement).....	1460
string ReplaceWithExpression (string input, string searchExpressionAsString, string replacement).....	1461
string ReplaceWithExpressionForCount (string input, string searchExpressionAsString, string replacement, int count).....	1462
string SHA1Checksum (string value).....	1463
string ToLower (string value).....	1464
string ToUpper (string value).....	1465
string ToCamel (string value).....	1466
string ToPascal (string value).....	1467
bool EndsWith (string input, string suffix).....	1468
string WebWorksHelpContextOrTopic (string key).....	1469
Units.....	1470
double Convert (double sourceValue, string sourceUnits, string targetUnits).	1471
string CSSRGBColor (string htmlColor).....	1472
string EncodingFromCodePage (int codePage).....	1474
string NumericPrefix (string value).....	1476
string RTFColor (string htmlColor).....	1477

string UnitsSuffix (string value).....	1479
URI.....	1480
string AsFilePath (string uriAsString).....	1481
string AsURI (string filePathAsString).....	1483
string EscapeData (string unescapedString).....	1485
string EscapeUri (string unescapedUri).....	1486
string GetRelativeTo (string uriAsString, string anchorUriAsString).....	1487
bool IsFile (string uriAsString).....	1489
string MakeAbsolute (string absoluteUriAsString, string uriAsString).....	1491
XPathNodeIterator PossibleResolvedUris (string uriAsString).....	1493
string Unescape (string escapedString).....	1495
ZipExtension.....	1497
void Zip (string zipFilePath, object nodes).....	1498
void ZipAdd (string zipFilePath, object nodes).....	1499
void ZipAddWithoutCompression (string zipFilePath, object nodes).....	1500
void ZipDirectory (string zipFilePath, string directoryPath).....	1501
void ZipDirectoryWithoutCompression (string zipFilePath, string directoryPath).....	1502
void ZipExtract (string zipFilePath, string targetDirectory).....	1503
void ZipWithoutCompression (string zipFilePath, object nodes).....	1504
<b>Frequently Asked Questions.....</b>	<b>1506</b>
What changed in Font Awesome version 5.15.4 from 4.7.x that impacts Reverb 2.0?.....	1506
How do I disable automatic Preview generation in Designer?.....	1507
How do I capture source file meta data in the published HTML output?.....	1508
How do I use a Project Variable's value in the published output?.....	1509
How do I upgrade an existing WebWorks Reverb 2.0 project to a newer version?.....	1510
How to add an image icon and stylize your DITA Hazard statement elements?..	1510
How to supplement or replace Reverb Toolbar Group Tabs.....	1511
Invalid File Path Characters Filtered from Generated File Names.....	1513
How to enable 'WebWorks Menu' (Transit) Add-in for Microsoft Word.....	1514
How do I set image horizontal alignment in ePublisher?.....	1516
Why is the PDF format skipping Markdown files?.....	1517
How can a markdown file include another?.....	1517
How can you insert an Index Marker into a Markdown file?.....	1518
How to use alternate DITA-OT installation with ePublisher.....	1518
How do I embed HTML within DITA source content?.....	1518

How to use Context Links in Reverb 2.0.....	1520
What happens if a Reverb link is no longer valid?.....	1521
How do I know what baggage files are in my Reverb 2.0 output?.....	1521
What user interactions can Reverb track in Google Analytics?.....	1522
How can I modify the Reverb search result summary?.....	1523
How to add Keywords meta data to generated HTML pages.....	1523
Why does HTML content within my source file not publish to PDF?.....	1524

# ePublisher Platform Documentation

[What's New in ePublisher 2024.1](#)

Welcome to the ePublisher Platform Documentation! Here you'll find everything you need to effectively create, customize, and publish your content using ePublisher's powerful tools. Whether you're a first-time user or a seasoned pro, our documentation is designed to guide you through every step.

## Chat With WebWorks Wizard on ChatGPT

For an even faster way to get your questions answered, try out [WebWorks Wizard](#). This custom GPT assistant is trained on the same documentation you're reading, making it a perfect companion to help you find what you need quickly and easily. Usage of WebWorks Wizard is free, and only requires an OpenAI account to access.

## Main Sections

Explore the main sections using the links below, search for specific topics with the search bar, or browse through the Table of Contents.

### Latest Release

- [What's New](#)
- [Release Notes](#)

### Authoring Content

- [Markdown++](#)
- [Adobe FrameMaker](#)
- [Microsoft Word](#)
- [DITA XML](#)

## Using ePublisher

- [ePublisher Interface](#)
- [Automap Interface](#)

## Generating Output

- [Output Formats](#)

## Customizing Output

- [Designing Stationery](#)
- [Advanced Customizations](#)

## Frequently Asked Questions

- [FAQs](#)

# What's New in ePublisher 2024.1

Read on to learn about the latest features available in this ePublisher release.

## Adapters

### FrameMaker 2022 Update 4/5 Support

The FrameMaker adapter has been updated to support update 4 and 5 of version 2022, enabling improved compatibility with the latest authoring tools. Users can now leverage the newest features and capabilities available in FrameMaker 2022 within ePublisher.

## Markdown++ Enhancements

- Added support for parsing multiline tables in Markdown++, improving flexibility when handling complex tables.
- Implemented multiline table output in Markdown to enhance content display consistency.
- Plain-text URLs now properly render in output, ensuring that Markdown files are displayed as expected.
- Index markers with multiple entries no longer have leading whitespace, ensuring better structured and clearer output.

## Formats

### Markdown++ Output Format

- Introduced a new Markdown++ output format, allowing for more advanced Markdown features and improved compatibility with existing workflows.
- Locale strings for Danish and other languages updated, providing better localization support.
- Renamed "Quote WIF" element to "Block" for greater clarity and consistency.

## XML+XSL Output Deprecation

- XML+XSL output format has been deprecated in this release to streamline supported formats and reduce maintenance complexity.

## Reverb 2.0

### Popups Support

- Full support for creating Popup content.
- Enhanced popup behavior for hyperlinks to generated content, providing a more responsive user experience.

## Performance Enhancements

- Improved load performance and eliminated duplicate files to streamline Reverb 2.0 operations.
- Enhanced efficiency of ASP template file dependency copying, resulting in faster build times.
- Improved search results and result highlighting.

## PDF - XSL-FO Integration

- Project-wide PDFs are now supported using the page toolbar PDF buttons if it is the only "Result" specified in the `Copy PDFs from target` target used for PDF generation.
- Added PDF buttons to the Splash and "File Not Found" pages.

## PDF - XSL-FO

- Added Paragraph Style Option: `Start new page sequence` which allows users to change page layouts within the same generated PDF. In prior releases this functionality was achieved using the `Page break priority` which was less flexible and could cause excessive page breaking between paragraphs.
- Improved macro support within the title page, headers, and footers. Including the ability to use ePublisher project variables (`projvars`) and source content meta data via style variables (`wwvars`).
- Improved default experience with page headers and footers.

## User Interface

### Style Designer Inheritance

- Style target properties, options, and target options can now inherit their values the same way properties have always been able to inherit their values.
- Style properties and options display inherited values when either an "Explicit" value has been set or if a default value exists for that property/option somewhere in the inheritance hierarchy.

### Updated Release Artwork

- Updated the release artwork for version 2024.1, providing a refreshed visual experience for users.

### Script Error Dialog Fix

- Fixed the script error dialog that appeared when accessing help or trial guide, ensuring a smoother user experience.



# **Evaluation Materials**

## **Updated Evaluation Package**

- Updated ePublisher evaluation materials for 2024.1, ensuring new users have access to the most recent information and guidance.

# **Build Improvements**

## **Version and Signtool Updates**

- Version updated for the 2024.1 release.

# **Stationery**

## **Saving Stationery Enhancements**

- Resolved issues with saving as Stationery on read-only project folders with customizations, allowing users to maintain their workflow more effectively.

# **General Improvements**

## **Windows Long File Path Awareness**

- Windows filenames and paths are no longer limited to 249/260 characters.

## **Memory Management and Conversion Times**

- Improved conversion times by optimizing memory management, resulting in faster and more efficient project processing.

# ePublisher 2024.1 Release Notes

Improvements  
Fixed Issues  
Summary

---

## Improvements

### Adapters

- Updated FrameMaker support to include version 2022, ensuring compatibility with the latest authoring tools (EPUB2516).
- Parse multiline tables in Markdown++ for more flexibility when handling complex content (EPUB2535).

### Core

- Style properties, target properties, options, and target options now all support inheritance.
- Style properties and options display inherited value and source when available.
- Improved conversion times affected by memory management, resulting in more efficient project handling (EPUB2524).
- Enhanced support for long filename paths, addressing limitations with paths beyond 249/260 characters (EPUB2522).

### Formats

- Renamed "Quote" WIF element to "Block" for increased clarity in formatting (EPUB2528).
- Locale strings for Danish and other languages have been added, expanding the language support (EPUB2450).
- Added the ability to generate output in Markdown++ format, enhancing compatibility with Markdown workflows (EPUB2519).
- Deprecated XML+XSL output format to streamline supported formats and simplify maintenance (EPUB2537).

### Licensing

- Upgraded License Generator to .NET 4.7.2, with improved TLS handling to meet modern security standards (EPUB2540).
- Fixed misleading license error messages, changing 'GroupID' to 'Contract ID' for better clarity (EPUB2531).

### User Interface

- Fixed script error dialog when accessing help or trial guide, resulting in a smoother user experience (EPUB2529).
- Updated release artwork for version 2024.1, refreshing the visual presentation (EPUB2532).

## **Reverb 2.0**

- Fixed issues with phrase searches and incorrect search result highlighting and navigation (EPUB2598).
- Enhanced popup behavior for hyperlinks to generated content, providing a more intuitive user experience (EPUB2387).
- Improved load performance and eliminated duplicate files for better efficiency (EPUB2511).
- Improved efficiency of ASP template file dependency copying, reducing build times (EPUB2533).
- Addressed issues with combined index letter groups being out of order, improving the indexing experience (EPUB2509).
- Added support for project-wide PDF button links (EPUB2584), now also available in the Splash and NotFound pages.

## **Markdown Adapter**

- Implemented multiline table output to improve content display consistency (EPUB2534).
- Fixed an issue where plain-text URLs did not render correctly in output, ensuring all links are visible (EPUB2514).
- Addressed footnotes not publishing as documented, improving Markdown compliance (EPUB2518).
- Index markers with multiple entries now correctly handle leading whitespace (EPUB2525).
- Fixed an issue where code fences with style tags were ignored, ensuring proper code formatting (EPUB2527).
- Resolved an issue with stylized quotes or apostrophes causing generation to fail, enhancing content stability (EPUB2515).

## **Build**

- Version updated for 2024.1 release, reflecting the latest updates and enhancements (EPUB2536).
- Signtool emergency bypass implemented to address specific build scenarios (EPUB2520).
- License key updated for the 2024.1 version (EPUB2538).

## **Evaluation Materials**

- Updated ePublisher evaluation materials to provide prospective users with the latest information and guidance (EPUB2542).

## Stationery

- Resolved issues with saving as Stationery on read-only project folders with customizations, ensuring smoother workflows (EPUB2513).

## Documentation

- Improved online help packaging to ensure accurate and easily accessible information (EPUB2539).
- Matched product integration URL links to the new website for better navigation and usability (EPUB2543).

## PDF Generation

- Title page, header, and footer macros now support project variables and ePublisher style option variables.
- `RunningTitle` and `ChapterTitle` variables work more consistently across page breaks.
- Eliminated page breaking from FrameMaker and Word source documents as these are redundant and confusing.
- Added `Start new page sequence` paragraph style option.
- Improved master page default region handling for easier initial setup.
- Generating selected documents no longer tries to compile other unselected documents.
- Fixed an issue with the XSL-FO bullet property that was not rendering image files correctly (EPUB2545).
- Eliminated empty-value FO properties to improve PDF output quality (EPUB2549).
- Style group label `HTML` now labeled as `Layout`.
- Title page headers and footers are supported but disabled in the `Title.asp` file (EPUB2578).
- Title page publish date can now be styled using an ePublisher Paragraph Style (EPUB2587)
- Title page Styles can use absolute positioning (Top, Left) (EPUB2588)

---

## Fixed Issues

### List of Issues Fixed in This Release

Issue ID	Summary
EPUB1657	WebWorks Help - Infinite reload loading <code>blank.html</code> in Chrome
EPUB1874	Reverb 2.0 - Index entries should have option to group by letter not group
EPUB2114	Formats - Bullet and note paragraph styles need a better way to handle vertical alignment
EPUB2116	HTML Formats - meta description tag in generated files and marker type
EPUB2146	PDF deploy incorrectly copies the Files folder contents
EPUB2377	Reverb 2.0 - Scroll-to elements too close to top of page
EPUB2387	Reverb 2.0 - Popup behavior for hyperlinks to generated content
EPUB2416	Windows Protected Folders cause ePublisher to crash instead of graceful failure
EPUB2450	Formats - Locale strings for Danish and other languages
EPUB2478	Reverb 2.0 - Search highlighting does not filter stop words
EPUB2482	Reverb 2.0 - HTML Baggage file dependences are not copied to output folder
EPUB2484	Reverb - url_maps.xml Page element href attributes are incorrectly escaping underscore characters
EPUB2485	Build - Version for 2023.1
EPUB2486	DITA - Hazardstatement paragraph style name for the icon has incorrect 'Type' suffix

Issue ID	Summary
EPUB2488	FrameMaker - Native PDF generation fails without Distiller with 32bit FrameMaker
EPUB2489	Word - Improve Digital Signature for WebWorks Menu and other addins
EPUB2491	DITA - Upgraded DITA OT 4.0 support to use version 4.0.2
EPUB2492	DITA - Support for DITA OT version 4.1 using release 4.1.2
EPUB2493	Update HTML release notes
EPUB2494	Documentation - Update ePubublisher online help
EPUB2495	DITA - ant pipeline failure calling <code>fn:concat()</code> with multiple item sequence
EPUB2496	Reverb 2.0 - #context links to different help sets on same server are incorrectly treated as internal links
EPUB2502	Reverb 2.0 - Make TOC Icon Configurable to Left or Right
EPUB2503	ePublisher - Update Release Artwork for 2023.1
EPUB2505	Build - License Key for 2023.1
EPUB2506	Update Evaluation Materials to match latest release
EPUB2507	Update ePubublisher Express Evaluation Guide
EPUB2509	Reverb 2.0 - Combined Index letter groups out of order sometimes
EPUB2511	Reverb 2.0 - Improve load performance and eliminate duplicate files

Issue ID	Summary
EPUB2513	Saving as Stationery on Read-only Project folder with customizations can fail
EPUB2514	Markdown - Plain-text URLs do not render in Output
EPUB2516	FrameMaker - Update FDK for FrameMaker 2022 Update 4
EPUB2519	Markdown - Generate output in Markdown++ format
EPUB2520	Build - Signtool emergency bypass
EPUB2521	Reverb 2.0 - No way to redirect 404 Not Found links to the First page
EPUB2522	Long filename paths beyond 249/260 character limit not working
EPUB2524	Improve conversion times affected by memory management
EPUB2525	Markdown Adapter - Index markers with multiple entries may have leading white space
EPUB2528	Formats - Rename Quote WIF element to Block
EPUB2529	User Interface - Script error dialog appears when accessing help or trial guide
EPUB2530	Reverb 1.0 - Generation error when Search Implementation target setting set to None
EPUB2531	License error message says 'GroupID' instead of 'Contract ID'
EPUB2532	ePublisher - Update Release Artwork for 2024.1

Issue ID	Summary
EPUB2533	Reverb 2.0 - Improve efficiency of ASP template file dependency copy operations
EPUB2534	Markdown - Implement multiline table output
EPUB2535	Adapters - Parse multiline tables in Markdown++
EPUB2536	Build - Version for 2024.1
EPUB2537	Build - Deprecate XML+XSL output format
EPUB2538	Build - License Key for 2024.1
EPUB2539	Documentation - Online Help packaging
EPUB2543	Match website product integration URL links to new website
EPUB2544	Task - Update Release Notes
EPUB2545	PDF - XSL-FO Bullet property using image file not rendering
EPUB2549	PDF - XSL-FO Eliminate empty-value FO properties
EPUB2550	PDF - XSL-FO Invalid property values containing <code>NaNpt</code> for <code>text-indent</code> and <code>margin-left</code>
EPUB2554	Reverb 2.0 - Corporate skin doesn't ever finish loading
EPUB2555	PDF - XSL-FO break-before and other layout properties ignored for UList/OList styles
EPUB2556	Reverb - Google Translate language selector is not fully visible
EPUB2557	Improve security by removing all usage of Windows <code>System.IO.Path.GetTempFileName()</code>



Issue ID	Summary
EPUB2558	Word Transit help menu does not display online help
EPUB2559	Update DockableHelp to not use WebWorks Help API
EPUB2560	Update WebWorks Help API to match WebWorks Help SDK v. 4.0.11
EPUB2561	PDF - XSL-FO TOC, Index style prefix target setting should not be a dropdown
EPUB2562	PDF - XSL-FO Word source documents create excessive PDF page breaks
EPUB2563	PDF - XSL-FO Replace style option: 'Page break priority' with 'Start new page sequence'
EPUB2564	User Interface - Launch links in user's default browser instead of WebBrowser form
EPUB2565	Style Designer inheritance does not work for target properties or options
EPUB2566	PDF - XSL-FO Improve default master page region handling
EPUB2567	User Interface - PDF - XSL-FO not displaying properties consistently
EPUB2568	Reverb 2.0 - Deployment fails to copy analytics.js and connect.js to the deploy folder
EPUB2569	User Interface - In PDF - XSL-FO, rename HTML property group to Layout
EPUB2570	User Interface - Style Designer property source assignment not working consistently
EPUB2571	Installer - Remove empty folder for Designer > Helpers > WebWorks

Issue ID	Summary
EPUB2572	PDF - XSL-FO Generate selected can cause FOP failure of non-selected documents
EPUB2575	PDF - XSL-FO Title page, headers, footers can use variables and source meta data
EPUB2577	PDF - XSL-FO Running titles used in headers/footers are not consistently set
EPUB2578	PDF - XSL-FO Title page headers/footers can be enabled in <code>Title.asp</code>
EPUB2579	PDF - XSL-FO <code>float</code> property emitted in images and causes RenderX XEP exception
EPUB2580	User Interface - Deleting Style with set properties and options sometimes causes exception
EPUB2581	PDF - XSL-FO Set first body page-sequence <code>initial-page-number</code> property to 1 when not set
EPUB2582	Reverb 2.0 - Use project-wide generated PDF for PDF icon link
EPUB2583	PDF - XSL-FO Line breaks create excessive white-space and can cause page breaks
EPUB2584	PDF - XSL-FO Project-wide PDF filename should use Merge Settings Title
EPUB2585	Markdown - files in "Files" folder should not be copied to output
EPUB2586	PDF - XSL-FO Bookmarks are disabled when Table of Contents is disabled
EPUB2587	PDF - XSL-FO Unable to set style name for Title page Publish Date

Issue ID	Summary
EPUB2588	PDF - XSL-FO Use <code>block-container</code> with <code>block</code> for Title Page Title/Subtitle/Publish Date Styles
EPUB2589	PDF - XSL-FO Disable Layout Type and Layout Clip properties
EPUB2590	PDF - XSL-FO Disable <code>Generate per Document result</code> as a default setting
EPUB2591	User Interface - Missing localization resource strings discovered in <code>de</code> , <code>fr</code> , and <code>ja</code> locales
EPUB2595	FrameMaker - Update wwFDK.dll to support Windows 11
EPUB2596	Reverb 2.0 - Search highlighting after selecting search result does not work
EPUB2598	Reverb 2.0 - Phrases with stop words do not always highlight
EPUB2600	Reverb 2.0 - Index terms with "&" characters display as "&"

---

## Summary

The ePublisher 2024.1 release introduces significant improvements in support for the latest software versions, enhances memory management and load performance, and resolves multiple issues across Markdown, Reverb, and other formats. Key upgrades to licensing, PDF generation, and evaluation materials ensure better usability and compliance. This release also optimizes the user experience through several performance enhancements and issue fixes.

---

Copyright © 2001-2023 Quadralay® Corporation. All rights reserved.

Quadralay and WebWorks are registered trademarks of Quadralay Corporation. FrameMaker is a registered trademark of Adobe Systems, Inc. Microsoft and Windows are registered trademarks of Microsoft Corporation. Oracle is a registered trademark of Oracle Corporation. Sun, Java and JavaHelp are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. . Doc-To-Help is a registered trademark of ComponentOne LLC. All other trademarks used herein are the properties of their respective owners and are used for identification purposes only.

Portions of this software are copyrighted by others, as follows:

### **.NET Framework Checker NSIS plugin**

All NSIS source code, plug-ins, documentation, examples, header files and graphics, with the exception of the compression modules and where otherwise noted, are licensed under the zlib/libpng license.

### **Apache**

Copyright © 1999-2006 The Apache Software Foundation

### **DITA-OT**

The DITA Open Toolkit is licensed for use, at the user's election, under the Common Public License v1.0 or the Apache Software Foundation License v2.0. If, at the time of use, the Project Management Committee has designated another version of these licenses or another license as being applicable to the DITA Open Toolkit, user may select to have its subsequent use of the DITA Open Toolkit governed by such other designated license.

Copyright © 1999-2006 The Apache Software Foundation

### **Flvplayer**

The flvplayer is licensed for use under the Mozilla Public License (MPL) version 1.1.

### **Ghostscript API wrapper**

Copyright © 2004, Pelagon Limited. All rights reserved.

### **HTML Tidy**

Copyright (c) 1998-2016 World Wide Web Consortium (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All Rights Reserved.

Additional contributions (c) 2001-2016 University of Toronto, Terry Teague, @geoffmcl, HTACG, and others.

## **IKVM**

Copyright © 2002-2011 Jeroen Frijters All rights reserved.

## **ImageMagick**

Copyright © 2002 ImageMagick Studio, a non-profit organization dedicated to making software imaging solutions freely available.

## **International Components for Unicode (ICU)**

Copyright © 1995-2003 International Business Machines Corporation and others. All rights reserved.

## **iText**

Copyright © 1999-2005 by Bruno Lowagie and Paulo Soares. All rights reserved.

## **jQuery**

Copyright © jQuery Foundation.

## **Java and JavaHelp**

Copyright © 2003 Sun Microsystems, Inc. All rights reserved. Use is subject to license terms. Sun, Sun Microsystems, the Sun Logo, Solaris, Java, the Java Coffee Cup Logo, JDK, Java Foundation Classes (J.F.C.), Java Plug-in and JavaHelp are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

## **JGL - The Generic Collection Library for Java™**

Copyright © 1997 ObjectSpace, Inc. All Rights Reserved.

## **Microsoft DotNetZip**

Copyright © 2006, 2007 Microsoft Corporation, One Microsoft Way, Redmond, Washington 98052-6399 U.S.A. All rights reserved.

## **Microsoft Help Workshop**

Copyright © 2004 Microsoft Corporation, One Microsoft Way, Redmond, Washington 98052-6399 U.S.A. All rights reserved.

## **Mistune**

Copyright (c) 2014 - 2015, Hsiaoming Yang. All rights reserved.

## **NetAdvantage**

Copyright ©1992-2005 Infragistics, Inc., Windsor Corporate Park, 50 Millstone Road, Building 200 - Suite 150, East Windsor, NJ 08520. All rights reserved.

## **Newtonsoft.Json**

Copyright (c) 2007 James Newton-King.

## **NSIS (Nullsoft Scriptable Install System)**

Copyright (C) 1999-2017 Contributors.

**All NSIS source code, plug-ins, documentation, examples, header files and graphics, with the exception of the compression modules and where otherwise noted, are licensed under the zlib/libpng license.**

## **nsisDDE plugin**

Copyright (c) 2005-2010 Olivier Marcoux.

## **Oracle Help**

Copyright © 2002 Oracle Corporation. All Rights Reserved.

## **Python**

The Python programming language by Guido van Rossum, PythonLabs, and many contributors -- Python Software Foundation License version 3

The Python Software Foundation License version 3 is an OSI Approved Open Source license. It consists of a stack of licenses that also include other licenses that apply to older parts of the Python code base. All of these are included in the OSI Approved license: PSF License, BeOpen Python License, CNRI Python License, and CWI Python License. The intellectual property rights for Python are managed by the Python Software Foundation.

## **Saxon**

Copyright © 1999-2003 Intalio, Inc. All Rights Reserved.

## **Various .Net Utilities**

Portions copyright © 2002-2004 The Genghis Group (<http://www.genghisgroup.com/>)

## **Xerces**

Copyright © 1999-2004 The Apache Software Foundation. All rights reserved.



# Contacting Quadralay

Please contact us with your questions and comments. We look forward to hearing from you. If you need assistance with an issue, please contact Technical Support. The Support Web site allows you to review the support policy and create a case to track an issue.



<b>Telephone:</b>	877.893.2967 512.719.3399
<b>Email:</b>	<a href="mailto:info@webworks.com">info@webworks.com</a>
<b>Support</b>	<a href="http://www.webworks.com/Support/">www.webworks.com/Support/</a>
<b>Web Site:</b>	<a href="http://www.webworks.com">www.webworks.com</a>

## Conventions

WebWorks ePublisher documentation uses consistent conventions to help you identify items. The following table summarizes these conventions.

Convention	Use
<b>Bold</b>	<ul style="list-style-type: none"> <li>• Window and menu items</li> <li>• Technical terms, when introduced</li> </ul>
<i>Italics</i>	<ul style="list-style-type: none"> <li>• Book titles</li> <li>• Variable names</li> <li>• Emphasized words</li> </ul>
Fixed Font	<ul style="list-style-type: none"> <li>• File and folder names</li> <li>• Commands and code examples</li> <li>• Text you must type</li> <li>• Text (output) displayed in the command-line interface</li> </ul>
Orange	<ul style="list-style-type: none"> <li>• Links</li> </ul>
>	<ul style="list-style-type: none"> <li>• Submenu selections, such as <b>Target &gt; Target Settings</b></li> </ul>
Brackets, such as [value]	<ul style="list-style-type: none"> <li>• Optional parameters of a command</li> </ul>
Braces, such as {value}	<ul style="list-style-type: none"> <li>• Required parameters of a command</li> </ul>
Logical OR, such as value1   value2	<ul style="list-style-type: none"> <li>• Exclusive parameters. Choose one parameter.</li> </ul>

# Introduction to the WebWorks ePublisher Platform

[What Is ePublisher?](#)

[How ePublisher Helps You](#)

[How Organizations Use ePublisher](#)

Content development, publication, and maintenance are complex, time-consuming processes. Each day, companies spend numerous hours writing, formatting, and publishing information needed by internal and external users. At the same time, these users search among vast amounts of content to find the information they want and need. Companies need to streamline content production processes while delivering the content to users when, where, and how they need it.

Adding to this complexity, teams across the company use multiple content authoring tools, such as Markdown++, Adobe FrameMaker, Microsoft Word, and XML editors, to create the content. These teams must be able to use the authoring tools that best meet their needs. However, mastering these tools for content development is only half the battle. Content developers also need tools to publish content consistently in multiple formats, such as print, HTML, and PDF. This requirement is difficult to meet and often leads to increased production costs or an inconsistent corporate image. In addition, corporate branding standards change over time, and implementing these changes across all content adds to production and maintenance costs.

With all these variations in content creation, publication, and maintenance, delivering the right information to the right people in the right format and at the right time is an endless and costly struggle that consumes enormous time and resources across organizations.

## What Is ePublisher?

The WebWorks ePublisher Platform (ePublisher) is a powerful, comprehensive solution that delivers cost-effective processes for efficiently publishing and maintaining online and print information. ePublisher gives you the flexibility to deliver content from multiple types of source documents, such as Markdown++, Adobe FrameMaker, Microsoft Word, and DITA, in virtually any output format you need without incurring training or software deployment expenses. The open, standards-based architecture provides a powerful engine that does not lock your content in a proprietary format that can become outdated as tools and standards change.

With the robust combination of input and output formats supported by ePublisher, you can develop the content using your preferred content authoring tools, and then produce and maintain all your deliverables within a single publishing environment. You can implement a consistent look and feel across all deliverables and quickly

modify and deploy that branding if and when needed. ePublisher integrates seamlessly with your content management or version control systems, so you can automatically generate and deploy the deliverables you need and reduce the time demands on your teams.

## Workflow

WebWorks ePublisher Platform components provide a workflow that ensures you can deliver your content, your way, every time. A successful online content delivery workflow includes the following items:

- Creation and automation of consistent, reusable online content designs
- Packaging of online content designs for seamless, consistent reuse
- Application of online content designs to new and existing projects to ensure consistent content delivery and deployment

ePublisher supports this workflow by allowing ePublisher users to perform the following tasks:

- Stationery designers use ePublisher Designer to create and manage online content designs, then package the designs into Stationery for writers to use
- Writers use ePublisher Express and Stationery to create and deploy consistent online content
- ePublisher AutoMap can be configured to automatically generate and deploy online content

For more information about the WebWorks ePublisher Platform components that support this workflow, see “WebWorks ePublisher Platform Components”. For more information about the ePublisher workflow, see “Understanding the ePublisher Workflow”.

## WebWorks ePublisher Platform Components

With ePublisher, you can write your content in your preferred authoring tools, then use ePublisher components to design and deliver your content. The ePublisher components allow you to design all the content output formats you need, and then automate the publication process and integrate it with your company-wide processes, such as product builds and Web site updates.

ePublisher includes the following components:

### **ePublisher Designer**

The design tool for creating and designing Stationery. **Stationery** defines the appearance and functionality of all the output formats you need. ePublisher provides several default formats that you can use as a basis for your Stationery, and then you can customize that standard and save it as your Stationery for your deliverables produced using the other ePublisher components.

### **ePublisher Express**

The on-demand publishing tool that transforms your content based on your Stationery and converts your source documents into the desired output formats. This component is installed on the desktop and integrates with your existing authoring tools to support the features you require, such as related topics and expand/collapse sections within your deliverable. With this component, you can quickly prepare your source documents and generate your final deliverables.

### **ePublisher AutoMap**

The automation tool that enables you to automate the content conversion process, batch processing, and integration with content management or version control systems. This component lets you schedule conversion projects to occur at times when you are not using your computer. For example, you can schedule the conversion to occur overnight. Then, when you arrive the next morning, your transformed content is ready for you. You can also automatically generate and deploy deliverables to meet your specific needs, such as updating Web site content based on updated source documents.

### **ePublisher Legacy Formats**

Supports projects that use output format versions of ePublisher that are no longer supported. These are available in a separate installer. Contact support for details.

## **Supported Input Formats**

ePublisher provides a single-sourcing environment that works with the following input formats.

- Markdown++
- Adobe FrameMaker
- Microsoft Word
- DITA XML Files

## Supported Output Formats

ePublisher lets you define your **output formats**, such as XHTML and WebWorks Help, so content developers can focus on developing quality content without worrying about tedious conversion requirements for various deliverables. You can manage your content as you want and produce deliverables in the following formats:

- WebWorks Reverb 2.0
- WebWorks Reverb
- WebWorks Help 5.0
- Dynamic HTML
- Microsoft HTML Help
- Eclipse Help
- Sun JavaHelp 2.0
- Oracle Help
- PDF
- PDF - XSL-FO
- eBook - ePUB 2.0
- XML+XSL

## How ePublisher Helps You

With ePublisher and its agile enterprise publishing capabilities, you have unparalleled design flexibility with the ability to deliver your information, regardless of input format, in multiple output formats. This solution enables both large and small organizations to implement the publishing environment that works best for them.

## Streamline and Automate the Content Publishing Process

In a traditional content authoring environment, a content author produces content designed for a single output format. This environment typically has the following limitations:

- Content is often duplicated across multiple content-producing teams.
- Content is not maintained consistently across the multiple teams.
- Production is expensive with multiple tools and technologies.

Using ePublisher, you can quickly publish content from your source documents. You can develop the content using your preferred content authoring environment, such as Markdown++, Microsoft Word, Adobe FrameMaker, or DITA. You do not have to spend time and resources learning how to format content for each and every output format in which your content will be delivered. You can focus on the content, and then use your Stationery to quickly and easily deliver information in the multiple formats that meet the requirements for your organization.

ePublisher reduces wasted time and expenses that occur when multiple groups in your organization unknowingly produce the same information at the same time. This publishing environment allows your organization to produce content one time. This content can then be shared in varying input formats across your organization.

## **Produce High Quality Deliverables with Fewer Individual Dependencies**

Instead of requiring team members to understand the entire process to produce content in multiple formats, team members can focus on their areas of expertise. Content developers can create informative content to add value to the products and services they document. You can also expand the skills of individual team members into important new areas, which strengthens your team as a whole.

ePublisher lets you concentrate on producing high-quality content within the authoring environment that works for you. You spend much less time on designing, implementing, and delivering multiple output formats. With ePublisher, you quickly generate complete, ready-to-deploy publications.

ePublisher also allows you to preview, proof, and review your content before you publish and deploy it. The comprehensive reports and on-demand reporting help you identify and correct any issues that may effect your online content, such as invalid styles, missing links, and compliance with Web accessibility standards. In this way, ePublisher ensures that the content you produce is of the highest quality and consistency.

## **Reduce Support Costs and Increase Customer Satisfaction**

When you consider the many steps content goes through to get from the content developer to its final destination, publishing content can be a tedious, costly

process. When an organization does not commit its attention and resources to product information, the negative results impact many aspects of the business.

Customers want to find their solution by reading as little as possible. Consistent content helps customers skim the content and find the information they need. The time content developers spend formatting content for various output formats reduces the time they have to review and improve the content. Customers can become frustrated when they spend time sorting through inconsistent and potentially inaccurate or incomplete information.

Frustrated customers quickly give up and contact customer support, often with a negative impression of the company. Increased customer support calls, especially for basic concepts and product usage, can waste valuable company resources.

ePublisher provides a workflow designed to make the publishing process as non-intrusive as possible. By allowing you to choose your preferred authoring environment and having role-focused software components, training time and production costs are reduced. The published content is consistent and delivers more value to your customers.

## Quickly Update and Deliver Content More Often

If your style or online content requirements change in your organization, you need to make changes throughout your content to implement these new requirements. Extended production times increase the difficulty and complexity of these changes. ePublisher streamlines the production process to enable you to quickly implement and deploy updated content. With these streamlined processes in place, you can deliver updated content more often.

ePublisher allows you to define and deploy centralized Stationery that all projects use. When corporate standards change, such as logos and branding, you can quickly update the Stationery to incorporate the new standards. Then, content developers can import the updated Stationery into their projects and publish their updated deliverables using the new standards without changing or redesigning their source documents.

## Reduce Content Management Life Cycle Costs

Due to the limitations of the traditional content model, many organizations want to move to a single-sourcing environment. **Single sourcing** allows the same content to be used multiple times and delivered in different formats. Organizations use single sourcing to eliminate duplicate content, reduce content translation and maintenance costs, improve content consistency, and minimize errors. Single sourcing also allows organizations to produce information in various formats using the same source.



Many single-sourcing solutions require all content authors to use the same authoring tool. ePublisher allows you to develop the content using your preferred content authoring environments, such as Markdown++, Microsoft Word, Adobe FrameMaker, or DITA. Each department can standardize on the authoring tool that is right for them, and ePublisher ties all the input formats together with a single, unified, reliable publishing process. ePublisher allows you to create integrated deliverables with source documents from multiple authoring tools.

With ePublisher, you can use your existing authoring tools and content management systems to meet organization-wide publishing needs without incurring training or software deployment expenses. The open architecture, based on industry-standard XSL, provides a flexible solution that you can customize to meet your needs without locking you into a proprietary format that could result in expensive future migration costs.

## **How Organizations Use ePublisher**

Companies use ePublisher to meet many of their content development, delivery, and maintenance needs:

- Update Web sites automatically with thousands of HTML pages every day
- Merge content across functional boundaries and deliver consistent content on corporate intranets and extranets
- Deliver integrated, context-sensitive help systems with products
- Single-source and deliver content in online and print formats
- Deploy content for multiple platforms and devices

The following sections highlight several ways you can use ePublisher to deliver consistent, comprehensive information.

## **Automatically Update Content on Web Sites**

Corporate Web sites have evolved into far more than just flashy advertising with contact information for your business. In addition to attention grabbing marketing about products and features, many company Web sites feature tutorials, product demos, specific product requirements and details, and Web 2.0 resources such as community forums where customers can share information.

With ePublisher, you can consistently update the content on your Web site to maintain the latest information and make sure it is available to your customers. You can schedule and automate content processing and deployment to deliver up to date information each and every day.

ePublisher also allows you to easily maintain corporate intranets and publish source documents from multiple organizations across your company. You can define a standard Stationery and templates for teams to use. You can then define an ePublisher job and schedule it to search a drop-box folder on a regular basis and publish the content from the source documents in that folder using your standard Stationery. This scenario ensures your team members have the latest information they need and reduces the expenses associated with publishing and maintaining this content on your intranet.

## **Deliver Full-Featured, Context-Sensitive Help Systems**

Products need to provide comprehensive help systems that meet the needs of many potential audiences. Content design and delivery must ensure that users get the information they need when, where, and how they need it. Some products need to deliver different content to different audiences. Other products are sold by multiple companies and require distinct product branding.

ePublisher provides comprehensive support for many advanced features used in online content design and delivery, including the following elements:

- Customizable browse navigation and breadcrumbs
- Customizable table of contents and mini-TOCs
- Expandable/collapsible text sections
- Related topics
- Images, image maps, and multiple forms of multimedia
- Context-sensitive help topics
- Merged help systems (multi-volume help)
- Variables and conditions
- Includes (one source file includes another)
- Accessibility features, such as alternate text and long descriptions
- Field-level help

## **Produce Single-Sourced Print and Online Optimized Content**

Customers have different needs and expectations for product content. In many cases, producing information in multiple formats for users involves extensive conversion and customization work to develop and deliver the various formats. Content authors must shift their attention to manipulating and converting the content into the many different user formats, often for both print and online, instead of focusing their time and efforts on developing quality information for users.

With ePublisher, you can quickly and efficiently produce consistent, effective print and online content in multiple formats. ePublisher provides XML/XSL processing and intelligent caching to process your source documents faster than ever before.

ePublisher produces the formatting code for you, whether it is HTML, XML, Formatting Objects (PDF generation), or a completely custom format. You do not need to know how to tag files for various output formats. With ePublisher, content developers can produce a printable PDF manual and a comprehensive online help deliverable immediately after finishing their content using the Stationery defined separately from their content.

# Planning and Installing ePublisher

[Licensing Considerations](#)  
[Components and Supported Configurations](#)  
[Requirements](#)  
[Downloading ePublisher Installers](#)  
[Microsoft Windows Requirements](#)  
[Downloading and Installing the Microsoft .NET 4.7.2 Framework](#)  
[Installing ePublisher](#)  
[Working with Contract IDs](#)  
[Upgrading from Previous Versions](#)  
[Uninstalling ePublisher](#)  
[Troubleshooting Installation, License Keys, and Uninstallation](#)

This section helps you plan your ePublisher installation and install ePublisher components. This section provides information about ePublisher components and supported configurations and ePublisher requirements. This section also explains how to download and install ePublisher components, use your **contract identifier** (Contract ID), work with license keys, upgrade ePublisher, and troubleshoot installation and licensing issues.

## Licensing Considerations

Before you can generate output using ePublisher, you must have a valid Contract ID. ePublisher uses your Contract ID to automatically handle the licensing of all ePublisher components and features. Your Contract ID is valid for your ePublisher use. Your contract ID may also be valid for other users, as long as the other users were included in the contract associated with the contract ID at the time ePublisher was purchased or the other users have been added to the same contract.

Currently ePublisher is licensed based on component and source document input format. ePublisher components include ePublisher Express, ePublisher Designer, and ePublisher AutoMap. ePublisher input formats include Adobe FrameMaker, Microsoft Word, and DITA-XML. Based on the input format of the files you use to author content, you may have access to one or more input formats. For more information about each ePublisher component, see “WebWorks ePublisher Platform Components”. For more information about Contract IDs, see “Working with Contract IDs”.

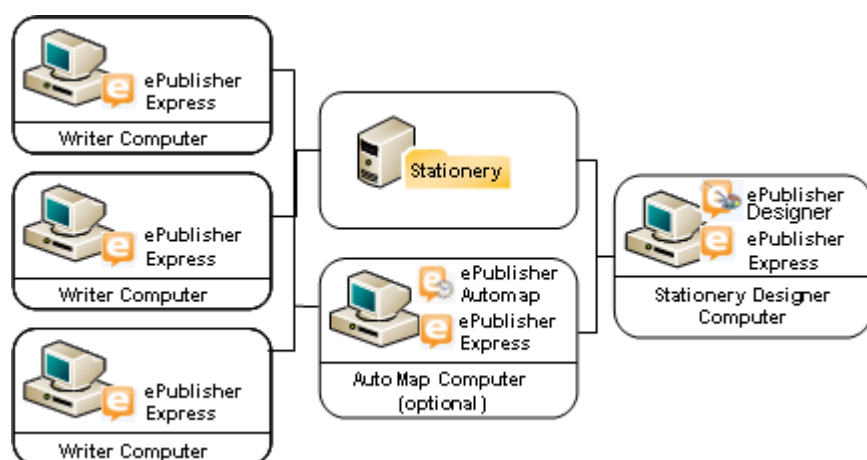
## Components and Supported Configurations

Stationery designers must install ePublisher Designer and ePublisher Express on their computers. Stationery designers use ePublisher Designer to design Stationery and ePublisher Express to test Stationery.

Writers install ePublisher Express on their computers. Writers use ePublisher Express to generate output using Stationery created by a Stationery designer.

If you want to use AutoMap to automate output generation and integrate your output generation with content management or version control systems, install ePublisher AutoMap and ePublisher Express on the computer where you want to use ePublisher AutoMap. ePublisher AutoMap requires ePublisher Express. You can install ePublisher AutoMap on its own separate computer, or you can install ePublisher AutoMap on a Stationery designer or writer computer where ePublisher Express is already installed.

The following figure shows a sample ePublisher configuration.



## Requirements

This section lists requirements for ePublisher components and input and output formats.

### ePublisher Express, ePublisher Designer, and ePublisher AutoMap Requirements

The following table lists the minimum and recommended system requirements for ePublisher Express, ePublisher Designer, and ePublisher AutoMap.

**Note:** Memory requirements can vary with the size of the job, including number of files to generate, size of each file, number of images and tables, and more. Generally, performance increases with available memory. The following values provide good performance for an average job.

	<b>Minimum</b>	<b>Recommended</b>
<b>Processor</b>	Intel i3	3.0 GHz Xeon Dual Core
<b>Memory</b>	1 GB RAM	16 GB RAM
<b>Available Disk Space</b>	1 GB available hard disk space	2 GB available hard disk space
<b>Operating System</b>	Microsoft Windows 10	<ul style="list-style-type: none"> <li>• Microsoft Windows 10 or 11</li> <li>• ePublisher AutoMap is also supported on Windows Server 2012 - 2022.</li> </ul>
<b>Additional Software</b>	<ul style="list-style-type: none"> <li>• Java 11+</li> <li>• Ghostscript (installed from Internet if missing, more info at: <a href="http://www.ghostscript.com">www.ghostscript.com</a>)</li> <li>• Microsoft .NET Framework 4.7.2 (installed from Internet if missing)</li> </ul>	<ul style="list-style-type: none"> <li>• Java 17 LTS (installed from Internet if missing)</li> <li>• Ghostscript (installed from Internet if missing, more info at: <a href="http://www.ghostscript.com">www.ghostscript.com</a>)</li> <li>• Microsoft .NET Framework 4.7.2 (installed from Internet if missing)</li> </ul>
<b>Display</b>	800 x 600 display screen resolution	1280 x 1024 display screen resolution (dual monitors supported)

## Additional Source Document Requirements

The following table lists the minimum and recommended system requirements for and Markdown++, Adobe FrameMaker, Microsoft Word, DITA Open Toolkit.

<b>Input Format</b>	<b>Minimum</b>	<b>Recommended</b>
<b>Markdown++</b>	N/A	N/A
<b>Adobe FrameMaker</b>	FrameMaker 2017	FrameMaker 2022
<b>Microsoft Word</b>	Word 2013	Word 2021
<b>DITA Open Toolkit</b>	Oracle JRE or JDK (Java), version 11+	Oracle JDK (or compatible), version 17 LTS

For additional information, please refer to the following page for up-to-date input requirements: [https://www.webworks.com/Support/ePublisher/System\\_Requirements/](https://www.webworks.com/Support/ePublisher/System_Requirements/)

## Additional Output Format Requirements

You can use ePublisher to produce output in several different formats. This section provides output format requirements for each output format ePublisher supports.

### WebWorks Reverb 2.0

WebWorks Reverb 2.0 can now be viewed directly from your computer's file system or from a running web server. This means that you can use this format to deliver online help as part of a non-networked help system. However, in order to provide Reverb's social-media capabilities (i.e. commenting, likes) to your end-users you must deploy the output to a system that is running a web server. Then your end-users must access the content via an **http** or **https** url. If you do not have a web server, you can configure IIS on Windows or any other available web server software. For more information on IIS, consult the following [resource](#).

When you are using ePublisher to generate WebWorks Reverb, ePublisher includes a viewer that you can use on the computer where you installed ePublisher to view the WebWorks Reverb output you generated using ePublisher.

### *Configuring web server for Reverb*

If deploying Reverb output to a public web server that has X-FRAME-OPTIONS configured, make sure that for HTML pages this header is not configured as follows:

```
X-Frame-Options: DENY
```

Instead it should be configured as:

```
X-Frame-Options: SAMEORIGIN
```

## ***Reverb browser requirements***

WebWorks Reverb has been tested on the following platforms:

- Internet Explorer 10+

**Note:** Internet Explorer has been discontinued by Microsoft

- Microsoft Edge
- Mozilla Firefox
- Safari
- Chromium browsers (Brave, Google Chrome, etc.)

### **Local Deployment Limitations**

Google Translate and Disqus Commenting are not functional when content is deployed locally (i.e. not hosted on a web server). However, all other capabilities of the help system will still be functional.

### **Web Server Deployment Requirements**

End users are not required to enable **DOM storage**. However, if not enabled, the **Thumbs Up, Thumbs Down** features will not record analytic events.

## **WebWorks Reverb 1.0 Limitation**

WebWorks Reverb 1.0 is the predecessor version of Reverb and requires that you deploy your output to a web server (not directly from your computer's file system).

When you are using ePublisher to generate WebWorks Reverb 1.0, ePublisher includes a viewer that you can use on the computer where you installed ePublisher to view the output you generated using ePublisher.

## **Dynamic HTML**

To view Dynamic HTML, users must have a browser that supports HTML 4.0 installed. HTML 4.0 was published in late 1997, and the major browsers, such as Internet Explorer, Firefox, and Safari support HTML 4.0. For more information about the HTML version a browser supports, see the documentation for the browser. If you choose to implement online features that require JavaScript, such as popups, users may also need JavaScript enabled. Most browsers have JavaScript enabled by default.



## PDF -XSL-FO

For generating PDF output, the PDF - XSL-FO output format is the best choice if you want to have complete control of your output's styling using ePublisher. If you are using Adobe FrameMaker or Microsoft Word, you can consider using the PDF output format, which relies upon the print engine of those authoring environments for styling.

To generate PDF - XSL-FO files, you must have the Java2 Platform SDK version 1.2.2 or later installed. You can download the Java2 Platform SDK for free from the Sun Microsystems Web site at <http://java.sun.com/javase/index.jsp>.

**Note:** Make sure to use Java version 8, 64-bit, if possible.

## eBook - ePUB 2.0

To generate output in this format, there are no external tools required. However, you will need a compatible ePUB reader in order to view the generated output. For development purposes, it is common practice to use an ePUB reader on your computer desktop, for example, you can use Adobe Digital Editions available at: <https://www.adobe.com/solutions/ebook/digital-editions.html>.

## Eclipse Help

To generate Eclipse Help, you must have the Java2 Platform SDK version 1.2.2 or later installed. You can download the Java2 Platform SDK for free from the Sun Microsystems Web site at <http://java.sun.com/javase/index.jsp>.

If you are using ePublisher to generate Eclipse Help, ePublisher includes a viewer that you can use on the computer where you installed ePublisher to view the Eclipse Help you generated using ePublisher.

To view Eclipse Help when you include Eclipse Help with an application, users must have the Eclipse integrated development environment (IDE) installed. Typically, application developers configure their applications to install the Eclipse IDE with the Eclipse Help content to ensure users can view the Eclipse Help while using the application. To view Eclipse Help, users must also have Microsoft Internet Explorer 6.0 or later or a Mozilla-based browser 1.7 or later installed.

## Microsoft HTML Help 1.x

To generate Microsoft HTML Help, you must have Microsoft HTML Help Workshop 1.x installed. If you do not have Microsoft HTML Help Workshop installed, ePublisher will ask you if you want to install Microsoft HTML Help Workshop during the ePublisher installation process. You can also download the Microsoft HTML Help Workshop for free from the Microsoft Developer Network Web site at <http://msdn.microsoft.com/en-us/library/ms669985.aspx>.

To view Microsoft HTML Help, users must have the Microsoft HTML Help viewer installed. The Microsoft HTML Help viewer is installed with most Windows operating systems in use today. Users must also have Internet Explorer 4.0 or later installed. Microsoft HTML Help does not require that users use Internet Explorer as their default browser. Microsoft

**Note:** Due to the legacy nature of this help run time, if you are generating your help from a networked location, you must map your help drive to a mapped letter such as z:\. UNC drives such as \\server.example.com\directory will not work as output locations for this help format. For more information on this issue, please refer to [Microsoft's Support website](#) and search for your version of Windows.

## Oracle Help

To generate Oracle Help, you must have the Java2 Platform SDK version 1.2.2 or later installed on your computer. You can download the Java2 Platform SDK for free from the Sun Microsystems Web site at <http://java.sun.com/javase/index.jsp>. The Java 2 Platform is also known as the Java Platform, Standard Edition (Java SE).

To view Oracle Help, users must have the Java Runtime Environment (JRE) installed on their computer. Typically application developers configure their applications to install the JRE with the Oracle Help content to ensure users can view the Oracle Help while using the application. Oracle Help components must be installed and viewed on the local computer.

## PDF

Most modern browsers such as Chrome and Microsoft Edge have the ability to read PDF files by default, though the user may also install Adobe Reader if a desktop application is needed. You can download Adobe Reader for free from the Adobe Web Site at [http://www.adobe.com/products/acrobat/readstep2\\_allversions.html](http://www.adobe.com/products/acrobat/readstep2_allversions.html).

## Sun JavaHelp 2.0

To generate Sun JavaHelp 2.0, you must have the Java2 Platform SDK version 1.2.2 or later installed on your computer. You can download the Java2 Platform SDK for free from the Sun Microsystems Web site at <http://java.sun.com/javase/index.jsp>. The Java 2 Platform is also known as the Java Platform, Standard Edition (Java SE).

To view Sun JavaHelp, users must have the Java Runtime Environment (JRE) installed on their computer. Typically, application developers configure their applications to install the JRE with the Sun JavaHelp content to ensure users can view the Sun JavaHelp while using the application. Sun JavaHelp components must be installed and viewed on the local computer.

## WebWorks Help 5.0

To view WebWorks Help, users must have JavaScript enabled in the browser. If JavaScript is not enabled, then the help system does not display in its entirety. For more information about determining whether JavaScript is enabled in your browser, see your browser options.

WebWorks Help has been tested on the following platforms:

**Note:** Due to modern browser security constraints, WebWorks Help 5.0, does not support local file system deployments (i.e. `file://` URLs).

- Microsoft Internet Explorer

**Note:** Internet Explorer has been discontinued by Microsoft.

- Microsoft Edge
- Mozilla Firefox
- Safari
- Chromium browsers (such as Brave, Google Chrome)
- Opera

Please refer to the WebWorks wiki for an up-to-date list of supported browsers:  
<http://wiki.webworks.com/Permalinks/BrowserSupport>

## Downloading ePublisher Installers

ePublisher installers are available for download as `.exe` files on a secure area on the WebWorks Web site. You can obtain ePublisher installers through one of the following methods:

- ***If you are evaluating ePublisher***, the WebWorks customer service team will send you an email that contains a link to the location where you can download the ePublisher installer.
- ***If you are a new ePublisher customer***, the WebWorks customer service team will send you an email that contains a link to the location where you can download the ePublisher installer when you purchase ePublisher.
- ***If you are an existing ePublisher customer with an active maintenance agreement***, the WebWorks customer service team will automatically send you an email that contains a link to the location where you can download the ePublisher installer each time a new version of ePublisher releases. If you have a My Cases login for the WebWorks technical support Web site, you can also obtain the ePublisher installer in the My Cases area when you log in to the WebWorks technical support site.

- ***If you are an existing ePublisher customer without an active maintenance agreement***, contact the WebWorks account management team for more information.

The link you receive to the download location for the ePublisher installer is typically active for only one to two weeks. ePublisher installer download locations are changed often for security reasons. If you need the latest link to an ePublisher download kit, you can request a link by submitting a support request on the WebWorks Web site at <http://www.webworks.com/Support/>. WebWorks technical support will verify that you purchased an ePublisher license for the requested component and then provide a link where you can download the requested installer.

## To download an ePublisher installer

1. Click the download link in the email from WebWorks.
2. On the WebWorks download page, click the link for the ePublisher component you want to install.
3. Click **Save**.
4. Browse to a location on your local computer where you want to save the installer, and then click **Save**.
5. Click **Close** when the download completes.
6. Browse to the location on your local computer where you saved the `.exe` file for the ePublisher component.
7. Run the `.exe` file.

# Microsoft Windows Requirements

ePublisher Requirements for running on Microsoft Windows.

## Downloading and Installing the Microsoft .NET 4.7.2 Framework

ePublisher components require the Microsoft .NET 4.7.2 Framework.

**Note:** The ePublisher Express installer will handle the download and installation of this component if it is not present on the user's machine. If the user cannot connect to the Internet while installing, then this component will need to be downloaded and installed manually before proceeding with the ePublisher Express installation. This component can be found for free at: <https://dotnet.microsoft.com/download/dotnet-framework/net472>.

In any case, after the installation of the Microsoft .NET 4.7.2, the user may need to restart the computer before installing ePublisher Express.

## Installing ePublisher

This section explains how to install ePublisher components. Read this section before you install ePublisher components.

# Installation Order for ePublisher Components

Ensure you install ePublisher components in the correct order. Review the following installation options before you install ePublisher components:

- ***If you want to generate output using Stationery created by a Stationery designer***, install only ePublisher Express on the computer.
- ***If you want to design Stationery and generate output***, install ePublisher Express and ePublisher Designer on the computer.

**Note:** ePublisher Designer requires ePublisher Express. Ensure you install ePublisher Express on the computer before you install ePublisher Designer.

- ***If you want to schedule and automate output generation***, install ePublisher Express and ePublisher AutoMap on the computer where you want to use ePublisher AutoMap.

ePublisher AutoMap requires ePublisher Express. Ensure you install ePublisher Express on the computer before you install ePublisher AutoMap. You can install ePublisher AutoMap using one of the following configurations:

- \_ On its own separate computer where ePublisher Express is already installed
- \_ On a Stationery design computer where ePublisher Express and ePublisher Designer are already installed
- \_ On a writer computer where ePublisher Express is already installed

## Installing ePublisher Components

This section provides instructions for installing ePublisher components, including ePublisher Express, ePublisher Designer, and ePublisher AutoMap.

**Note:** You must install ePublisher Express first. Then, you can install ePublisher Designer and ePublisher AutoMap. Both ePublisher Designer and ePublisher AutoMap require ePublisher Express.

***If you are upgrading from a previous version of ePublisher***, review the upgrade instructions. For more information, see “Upgrading from Previous Versions”.

## To install ePublisher components

1. Log on as a user using an account that is a member of the Administrators group on the local computer.
2. Close all instances of Microsoft Office applications running on the local computer, including instances of Microsoft Word and Microsoft Outlook. Close all instances of Adobe FrameMaker running on the computer.
3. Run the `WebWorks ePublisher <Product> (<32|64>-bit).<Version>.<Build_Number>.exe` file for the ePublisher component you want to install.
4. Review the welcome message, and then click **Next**.
5. Review the license agreement. If you agree to the terms of the agreement, click **I Agree**.
6. Select the application shortcuts you want to create, and then click **Next**.
7. Specify the location of the installation directory, and then click **Next**. The default installation directory is `C:\Program Files (x86)\WebWorks` or `C:\Program Files\WebWorks`.
8. Click **Next** to confirm your selections and to begin installing the ePublisher component.
9. In the WebWorks Licensing Info window, complete the following steps:
  - a. Enter your Contract ID. If you previously installed ePublisher on the computer using a valid Contract ID, ePublisher will automatically detect the Contract ID and display your Contract ID information. For more information about contract IDs and obtaining a Contract ID, see "Working with Contract IDs" "Obtaining Contract IDs".
  - b. Enter your email address. If you have an email address that you use as your WebWorks support login, enter that email address.
  - c. Enter the name of your computer.
  - d. Click **Confirm**.
10. **If the installer displays the HTML Help Workshop 1.3 Setup window**, you can install Microsoft HTML Help Workshop 1.3 as part of your ePublisher installation. Install Microsoft HTML Help Workshop if you plan to generate Microsoft HTML Help output.

- ***If you want to install Microsoft HTML Help Workshop 1.3***, click **Yes**, and then follow the instructions to install Microsoft HTML Help Workshop.
- ***If you do not want to install Microsoft HTML Help Workshop 1.3***, click **No**.

**11.** Click **Close** when the installation completes. ePublisher also opens a new browser window and displays a page on the [www.webworks.com](http://www.webworks.com) web site when the installation completes.

## Installing Ghostscript

Beginning with the 2019.2 release of ePublisher, the Ghostscript postscript processing tool will no longer be bundled with the ePublisher Express installer. Instead, the ePublisher Express installer attempts to detect if Ghostscript is already installed and if not, will attempt to automatically download and install it. If the installer is unable to install Ghostscript, then you can follow the steps below to manually install it on your system.

Ghostscript can be installed either before or after you install ePublisher. However, if you run conversions that require postscript processing, ePublisher will generate error messages in its generation log and the output may be missing image files.



## To download and install Ghostscript

1. Access the Ghostscript download page at: <https://github.com/ArtifexSoftware/ghostpdl-downloads/releases>.
2. Click on the link labeled `gs<VERSION>w64.exe` or `gs<VERSION>w32.exe` depending on if you are using the 64-bit or 32-bit version of ePublisher. For example: `gs9550w64.exe` for the 64-bit version of ePublisher.
3. Run the installer, it should have a filename similar to: `gs9550w64.exe`.
4. Check **Generate cidfmap for Windows CJK TrueType fonts**.
5. If ePublisher was running, make sure to restart it before continuing to use it.

## Ghostscript not Installed Warnings

If Ghostscript is not installed ePublisher will generate warnings of the following form:

```
[Warning] Attempt to render PostScript to 'jpeg' returned
'Unable to load DLL 'gsdll32.dll': The specified module
could not be found. (Exception from HRESULT: 0x8007007E)    at
Pelagon.GhostScript.gsapi_new_instance(IntPtr& pInstance, IntPtr
pCallerHandle)
```

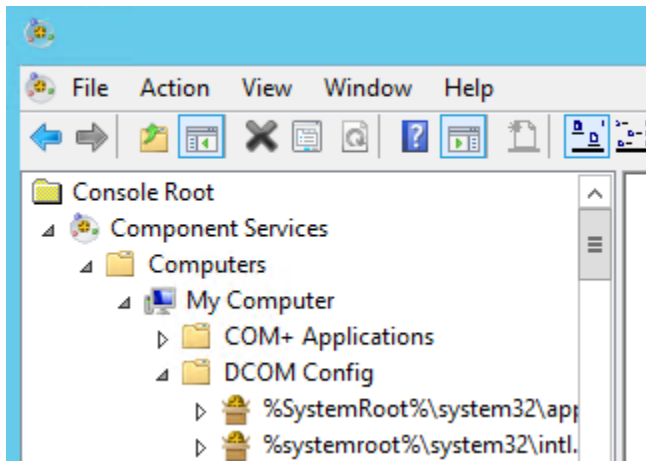
Once Ghostscript is installed, restart ePublisher and generate the project again.

## Configuring AutoMap for Microsoft Source Document Inputs

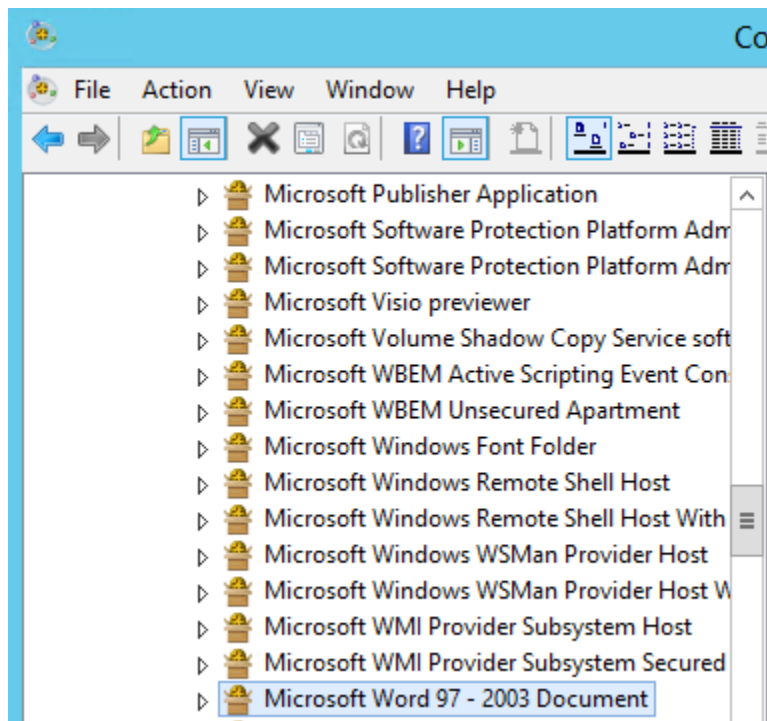
When installing ePublisher AutoMap and if publishing Microsoft Word documents, it may be necessary to configure the DCOM **Login Identify** for the **Microsoft Word 97 - 2003 Document** configuration. Depending on how you or your team will be running ePublisher AutoMap the user account that Microsoft Word will be launched as should be set accordingly.

## To configure the DCOM account Identify of Microsoft Word

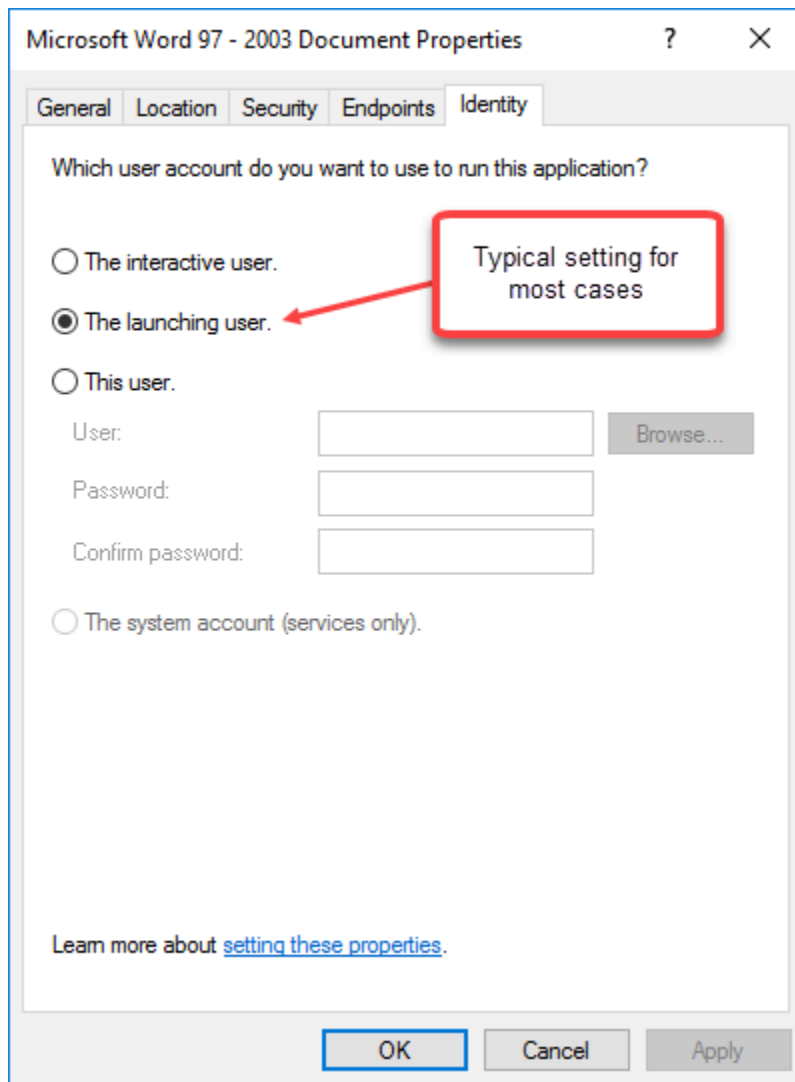
1. **If working with a 32-bit version of Microsoft Word**, from the **Start** menu, type: `mmc comexp.msc /32`
2. **If working with a 64-bit version of Microsoft Word**, from the **Start** menu, type: `mmc comexp.msc`
3. In the Component Services dialog, select **Component Services > Computers > My Computer > DCOM Config**.



4. Underneath **DCOM Config**, right-click **Microsoft Word 97 - 2003 Document** and select **Properties**.



5. In the **Microsoft Word 97 - 2003 Document Properties** dialog, select the **Identity** tab, then specify the user account to use to run Microsoft Word, which may or may not be a specific user depending on how you are planning to use ePublisher AutoMap.



## Understanding Installed Sample Projects and Stationery

ePublisher Express and ePublisher Designer install sample projects and Stationery. You can use these sample projects and Stationery to see some examples of how you can use ePublisher to generate output. For more information about using these sample projects and Stationery to generate output, see "Customizing Your ePublisher Workspace".

## Working with Contract IDs

ePublisher no longer requires you to manually enter license keys. ePublisher now uses Contract IDs to enable product functionality, which simplifies the ePublisher licensing process.

A Contract ID is a unique identifier that identifies the number of users and type of functionality enabled for your ePublisher installation. WebWorks generates an appropriate Contract ID for your ePublisher installation when you purchase ePublisher or request an evaluation copy of ePublisher. A Contract ID enables functionality based on the items and time frame specified in the purchase contract between your company and WebWorks.

If you have a valid contract ID for one version of the ePublisher product, when a new version of ePublisher releases, you can continue to use your same Contract ID when you upgrade to the new version of the product. You can also continue to use your same Contract ID if you have to uninstall and then re-install a version of ePublisher.

ePublisher licensing is flexible, and the WebWorks team can work with you ensure that you have the licensing that is right for you. Contact WebWorks Sales at [sales@webworks.com](mailto:sales@webworks.com) or WebWorks Customer Service at [customerservice@webworks.com](mailto:customerservice@webworks.com) to discuss any special licensing needs you may have.

## Viewing Licensing and Contract ID Information

You can view licensing information in the License Information window in ePublisher. ePublisher uses adapter license keys, or activation codes, to enable ePublisher functionality. Adapter licensing information is specified in your Contract ID. ePublisher uses an Internet connection to connect to the ePublisher licensing server and periodically retrieve and update adapter activation codes as needed based on your Contract ID.

**Note:** If you need to install ePublisher in an environment without Internet connectivity, WebWorks can provide Contract IDs that support this environment. For more information, see “Managing Licensing in Environments without Internet Connectivity”.

ePublisher licenses, or activation codes, do not display in the ePublisher user interface, but you can view the adapters for which you are licensed and your Contract ID number in the ePublisher user interface.

## To view ePublisher licensing and Contract ID information

1. On the **Help** menu, click **License Keys**. ePublisher displays the input formats for which the component is licensed in the License Information window.
2. *If you want to view your Contract ID number*, click **Register**.

## Obtaining Contract IDs

ePublisher now uses Contract IDs instead of license keys to enable ePublisher functionality.

***If you are evaluating ePublisher***, the WebWorks customer service team will send you an email that contains a Contract ID you can use when you install an evaluation copy of ePublisher. If you have not received an evaluation Contract ID or are having problems with your evaluation license, send an email to [customerservice@webworks.com](mailto:customerservice@webworks.com).

***If you are a new ePublisher customer***, the WebWorks customer service team will send you an email that contains your Contract ID when you purchase ePublisher. If you have not received a Contract ID or are having problems with your licensing, send an email to [customerservice@webworks.com](mailto:customerservice@webworks.com).

***If you are an existing ePublisher customer with an active maintenance agreement***, the WebWorks customer service team will automatically send you an email that contains a link to the location where you can download the ePublisher installer. ePublisher will automatically detect and use your existing Contract ID each time you install a new version of the ePublisher product. If you have not received a Contract ID or are having problems with licensing, send an email to [customerservice@webworks.com](mailto:customerservice@webworks.com) or submit a support request.

***If you are an existing ePublisher customer without an active maintenance agreement***, contact the WebWorks account management team for more information about obtaining your Contract ID by sending an email to [sales@webworks.com](mailto:sales@webworks.com).

For more information about Contract IDs, see “Working with Contract IDs” and “Entering Contract IDs”.

## Entering Contract IDs

ePublisher now uses Contract IDs instead of license keys to enable ePublisher functionality. You must enter your Contract ID, email address, and computer name before you can use ePublisher components. The Contract ID enables the ePublisher product components and ePublisher input formats for which you are licensed. For more information about Contract IDs, see “Working with Contract IDs”.

## To enter a Contract ID

1. On the **Help** menu, click **License Keys**. ePublisher displays the input formats for which the component is licensed in the License Information window.
2. Click **Register**.
3. In the **Contract** field, enter your Contract ID.
4. In the **Email** field, enter your email address. If you have an email address that you use as your WebWorks support login, enter that email address.
5. In the **Computer Name** field, enter the name of the computer where you are installing ePublisher.
6. Click **Confirm**.

## Managing Licensing in Environments without Internet Connectivity

ePublisher uses an Internet connection to connect to the ePublisher licensing server and retrieve or update adapter activation codes as needed based on your Contract ID. If you need to install ePublisher in a restricted environment where ePublisher computers do not have Internet access, contact WebWorks Sales at [sales@webworks.com](mailto:sales@webworks.com) or WebWorks Customer Service at [customerservice@webworks.com](mailto:customerservice@webworks.com) to request a non-network Contract ID. The ePublisher licensing model is flexible, and WebWorks can work with you to provide non-network Contract IDs or other licensing solutions appropriate for your environment.

## Updating Licensing

ePublisher automatically contacts the ePublisher licensing server as needed to obtain updated activation codes. ePublisher communicates with the ePublisher licensing server using an Internet connection. ePublisher obtains updated activation codes as appropriate based on the licensing specified in your Contract ID.

**Note:** If your ePublisher is installed in an environment without Internet connectivity, WebWorks can provide Contract IDs that support this environment. For more information, see “Managing Licensing in Environments without Internet Connectivity”

Typically, you will not need to request updated activation codes, as ePublisher obtains updated codes for your automatically. However, you can manually request updated activations codes in the ePublisher interface. For example, you may want to manually request updated activation codes if you know that the computer where you installed ePublisher will not have Internet access for a long period of time.

When you request updated activations codes in the ePublisher interface, ePublisher immediately establishes an Internet connection to the ePublisher licensing server and automatically obtains updated activation codes for the ePublisher adapters for which you are licensed.



### To update ePublisher licensing

1. On the **Help** menu, click **License Keys**. ePublisher displays the input formats for which the component is licensed in the License Information window.
2. Click **Refresh keys**. ePublisher retrieves updated activation codes from the ePublisher licensing server.

## Deactivating Licensing

You can deactivate ePublisher licensing in the ePublisher user interface. Deactivating licensing for the current ePublisher installation allows you to install ePublisher on a different computer without affecting the number of available seats allowed by your contract.

**Note:** The terms of the ePublisher end-user license agreement (EULA) allow you to install ePublisher Express or ePublisher Designer on one office computer and on one home or travelling computer for each assigned ePublisher user seat. ePublisher AutoMap licensing terms can vary based on whether ePublisher AutoMap was purchased on a per writer or per server basis, or in conjunction with a Content Management System (CMS).

### To deactivate ePublisher licensing:

1. On the **Help** menu, click **License Keys**.
2. Click **Unregister**.

## Upgrading from Previous Versions

In most cases, upgrading from a previous version to a new version of ePublisher can be accomplished in just a few steps. This section explains how to prepare for an upgrade, how to upgrade a typical ePublisher installation, and how to upgrade an ePublisher implementation with advanced customizations.

## Updating ePublisher installation

**Note:** If you are transitioning from 32-bit ePublisher to 64-bit ePublisher or vice-versa, be sure to use the installer from the older product or Windows' Add/Remove Programs to Uninstall the previous version of ePublisher before installing the new version of ePublisher. Versions of ePublisher on a different bit platform should be uninstalled before installing a new version of ePublisher on a different bit platform.

If you are installing an ePublisher component that is of version 2018.2 or higher and your currently installed component is version 2014.1 or higher, then your component will automatically be uninstalled for you when you choose the option to update. Otherwise, before installing a new version of an ePublisher component, you must uninstall any previous versions of the component. Uninstalling an ePublisher component removes the installation folder and registry entries for the component from the computer. Since Windows has different registries for 32-bit and 64-bit applications, this only applies to installers of the same platform.

## To update/repair an ePublisher component with an ePublisher executable installer

1. Close all ePublisher user interfaces.
2. Close all instances of Microsoft Office applications running on the local computer, including instances of Microsoft Word and Microsoft Outlook. Close all instances of Adobe FrameMaker running on the computer.
3. Double click the executable installer.
4. Select whether you want to Update/Repair, and then click **Next**.
5. Follow the instructions in the consecutive pages.

## Preparing existing projects for ePublisher Upgrade

To ensure smooth migration of existing projects and stationeries perform the following steps:

- Save your Stationery, Stationery design projects, and any projects you currently use to generate output to a secure location. **Stationery** defines the appearance and functionality of all the output formats you need. **Stationery design projects** are the ePublisher Designer projects used to create Stationery. For more information about Stationery and Stationery design projects, see “Understanding Stationery” and “Creating a Stationery Design Project”. For more information about Stationery and Stationery design projects, see “Understanding Stationery” and “Creating a Stationery Design Project”.
- ***If you implemented overrides when designing Stationery***, then you will need to make sure and migrate each overridden file so that it is compatible with the latest format files of the new release, or decide to keep the base format version of the file unchanged. Examples of overrides include the following items:
  - \_ Modifications to the `Page.asp` file
  - \_ Custom `.css` files
  - \_ Custom `.scss` files
  - \_ Modifications to image files

- Any advanced overrides such as modifications to `.xsl` or `.fti` files or files in the `Formats` or `Targets` folder.

When the Stationery designer creates and saves Stationery, ePublisher creates the following folders:

- `StationeryName\Formats\OutputFormat`
- `StationeryName\Formats\OutputFormat.base`

where *StationeryName* is the name the Stationery designer specified for the Stationery, and *OutputFormat* is the type of output format the Stationery Designer specified for a target in the Stationery.

The `StationeryName\Formats\OutputFormat` folder contains any customizations or overrides the Stationery designer specified when designing the Stationery. ePublisher Express synchronizes with the files in the *OutputFormat* folder and uses the information about customizations and overrides contained in files in the *OutputFormat* folder to generate output.

**Note:** The Stationery may have one or more *OutputFormat* folders, based on the settings the Stationery designer specified.

The `StationeryName\Formats\OutputFormat.base` folder contains copies of all the files located in the `\Program Files\WebWorks\ePublisher\release_number\Formats\OutputFormat` folder. These files define the default output format and transforms and are installed by default when you install ePublisher.

Stationery designers can do a compare, or **diff**, between the files located in these folders to quickly see any customizations or overrides specified for the Stationery. Stationery designers can use this information to help them reapply customizations and overrides as needed when designing a newer version of the Stationery in ePublisher Designer.

For more information about overrides, see “Stationery, Projects, and Overrides”.

## Upgrading Typical ePublisher Implementations

After you save your existing Stationery design projects, Stationery, any projects you currently use to generate output, and any copies of override files to a secure location, perform the following steps:

- On the Stationery designer computer, uninstall all existing versions of ePublisher components, such as ePublisher Express, ePublisher Designer, and ePublisher AutoMap. The **Stationery design computer** is the computer the

Stationery designer uses to create and update Stationery. ePublisher Express and ePublisher Designer are installed on the Stationery design computer. Based on your configuration, ePublisher AutoMap may also be installed on the Stationery design computer.

- Install the new version of ePublisher Express and ePublisher Designer on the Stationery designer computer. Also install the new version of ePublisher AutoMap if you run ePublisher AutoMap on the Stationery designer computer.
- Open your existing Stationery design projects using the new version of ePublisher Designer.
- Generate output and verify that your output generates as expected. Make any adjustments as needed.

- ***If you have implemented typical overrides in a Stationery design project***, such as overrides to `Page.asp` files, custom `.css` files, or `image` files, you can continue to use your overrides to these files, and the new version of ePublisher will recognize and use these existing modifications when generating output.

- ***If you have implemented advanced overrides***, such as overrides to `.xsl` or `.fti` files, or overrides to files in the `Formats` folder, update these files in your new ePublisher installation to include your advanced overrides. For more information, see “Upgrading Implementations with Advanced Customizations”.

**Note:** When ePublisher Designer detects overrides, by default it will not update to the latest version of the format. This means that no modifications will be necessary in order to continue using your Stationery. However, in this default mode, you will not get any of the format improvements built into the latest release. If you want these improvements, then you will have to configure the **Project Settings** to use the latest version of ePublisher’s formats.

- Create new Stationery for each Stationery design project.
- Deploy the updated Stationery to an appropriate location.
- On each writer computer, update ePublisher Express installation.
- The next time writers generate output, they open their existing projects using the new version of ePublisher Express. Writers can choose to synchronize their projects immediately to obtain the latest Stationery and then generate output, or writers can continue using their existing Stationery until they are ready to move to the latest version of the Stationery.

# Upgrading Implementations with Advanced Customizations

If you have implemented advanced overrides in the Stationery design, such as overrides to `.xsl` or `.fti` files, or overrides to files in the `Formats` folder, ensure you save a copy of the following items to a secure location before uninstalling a previous version of ePublisher and installing a new version:

- Overrides currently used in the Stationery design project
- A copy of the original files from which the overrides were created

If you want to continue to use your advanced customizations with the new version of ePublisher, first uninstall your previous ePublisher version and then install a new ePublisher version. Then identify and include your overrides in the new versions of the ePublisher files as appropriate by performing a three-way merge of the following items:

- A copy of the existing override file used in the Stationery design project, located in the `StationeryName\Formats\OutputFormat` folder, where `StationeryName` is the name the Stationery designer specified for the Stationery, and `OutputFormat` is the type of output format the Stationery Designer specified for a target in the Stationery.
- A copy of the original file from which the override was created, available in the `StationeryName\Formats\OutputFormat.base` folder, where `StationeryName` is the name the Stationery designer specified for the Stationery, and `OutputFormat` is the type of output format the Stationery Designer specified for a target in the Stationery.
- A copy of the new file from the new version of ePublisher

Performing a three-way merge allows you to identify the code you changed when you created the override, and also allows you to quickly and easily create the override again in the new ePublisher files. You may find tools such as Araxis Merge Pro, available at <http://www.araxis.com/merge>, or KDiff3, available at <http://kdiff3.sourceforge.net>, helpful as you compare and merge override files.

After you perform your three-way merge and update the files you want to override in the new version of ePublisher with the overrides you specified in the previous version, test your overrides by generating output using the new version of ePublisher Designer and the Stationery design project to confirm your output generates appropriately. After you verify the output generated correctly using your advanced customizations, you can create new Stationery using the Stationery design project and then deploy the updated Stationery that includes your advanced customizations to writers to use to generate output.

# Upgrading Advanced Customizations of WebWorks Reverb 2.0

Most of the time, you will only need to customize the `*.scss` files to achieve all of your styling requirements for your WebWorks Reverb 2.0 output. However, the Reverb 2.0 format is highly customizable and if necessary you can make advanced customizations to this format. If you are upgrading from a prior release, then you will want to understand what files are most likely to be customized and how this is affected if you change the Skin **Target Setting**.

If you have or plan to make advanced customizations to the WebWorks Reverb 2.0 layout or look-and-feel, then most likely you will have to modify one or more of the following files.

Table 5: Advanced Reverb 2.0 Files that are typically customized

Filename	The display area or items affected by this file
<code>webworks.scss</code>	Content panel styling only. Includes the styling of the MiniTOC, RelatedTopics, Social Buttons.
<code>skin.scss</code>	Styling of TOC, Index, Toolbar, and Breadcrumbs. All icons used in the <code>skin.png</code> sprite file are managed here. Styling of content that appears above the Toolbar, such as the company information.
<code>search.scss</code>	Styling of the search results page.
<code>skin.png</code> (derived from <code>skin.Fireworks.png</code> )	PNG file with alpha channel that stores all of the Reverb icons.
<code>connect.asp</code>	Used to manage the button placement in the toolbar. Also manages the TOC/Index/Search panel title for the <i>Corporate</i> skin.
<code>connect.scss</code>	Manages basic structure of the entry-point file generated from the <code>connect.asp</code> template file.

When working with alternate skins, you need to be aware of which files are most likely affected as a result of changing the skin type. If you have Advanced Customizations in any of these files, then you need to re-examine the *diffs* of these files after you switch the skin type. Most likely you will have minimal differences. Here are some basic steps you can follow to make sure you translate those changes to the new skin properly.



## Basic steps for setting an alternate skin type when Advanced customizations are present

1. Check your **Advanced Customizations** for files listed in “Advanced Reverb 2.0 Files that are typically customized”.
2. Make sure any of these commonly customized files are implemented as **Target Overrides** as opposed to **Format Overrides**. Setting an alternate skin type will create an implicit target override that will have priority over any format overrides of the same name.
3. Before changing the skin type you will need to record any existing file differences. On the **Advanced** menu click **Manage Target Customizations**. Now use the procedure discussed in “Format and Target Overrides” to record these file differences. These file differences will be used later after the skin type has been changed.
4. On the **Target** menu, click **Target Settings**.
5. In the **WebWorks Reverb** category, select the right column of the **Skin** entry to display the file picker button.
6. Click the file picker button to bring up an **Open** file dialog which will display a list of skin plugin files. Each skin plugin file is identifiable by a `.weplugin` extension.
7. Browse to the plugin file that you wish to use and double-click it to set the skin to that value.
8. At this point, you need to consider either removing your existing customizations and then re-implementing them using the information from your previously recorded file differences. Or managing the differences directly by comparing the differences using the procedure discussed in “Format and Target Overrides”. Either method will work.

## Uninstalling ePublisher

Uninstalling an ePublisher component removes the installation folder and registry entries for the component from the computer.

Before uninstalling ePublisher, consider unregistering the license key(s) using the menu: **Help > License Keys...** and then selecting the **Unregister** button. Unregistering will make the license available for use on a different system.

**Note:** When uninstalling ePublisher Express, all other components should also be uninstalled. Doing this ensures that all components are compatible with each other and get installed with the same version and build number.

***If ePublisher installed the WebWorks Transit menu for Microsoft Word on the computer***, ePublisher removes the WebWorks Transit menu and WebWorks Transit registry entries when you uninstall the last ePublisher component on the computer.

## **To uninstall an ePublisher component with an ePublisher executable installer**

- 1.** Close all ePublisher user interfaces.
- 2.** Close all instances of Microsoft Office applications running on the local computer, including instances of Microsoft Word and Microsoft Outlook. Close all instances of Adobe FrameMaker running on the computer.
- 3.** Double click the executable installer.
- 4.** Select the Uninstall option, and then click **Next**.
- 5.** Follow the instructions in the consecutive pages.

## To uninstall an ePublisher component using Windows Control Panel

1. Close all ePublisher user interfaces.
2. Close all instances of Microsoft Office applications running on the local computer, including instances of Microsoft Word and Microsoft Outlook. Close all instances of Adobe FrameMaker running on the computer.
3. Open Control Panel.
4. Open Add or Remove Programs.
5. Select the ePublisher component you want to uninstall.
6. Click **Remove**.
7. Click **Yes** to confirm you want to remove the ePublisher component from your computer. ePublisher removes the selected ePublisher component.

# Troubleshooting Installation, License Keys, and Uninstallation

This section helps you troubleshoot issues related to the following ePublisher issues:

- Installing ePublisher. For more information, see “Problems Installing ePublisher”.
- Obtaining, adding, and removing Contract IDs and working with licensing. For more information, see “Problems with FrameMaker or Microsoft Word”.

## Problems Installing ePublisher

This section helps you troubleshoot issues related to installing ePublisher.

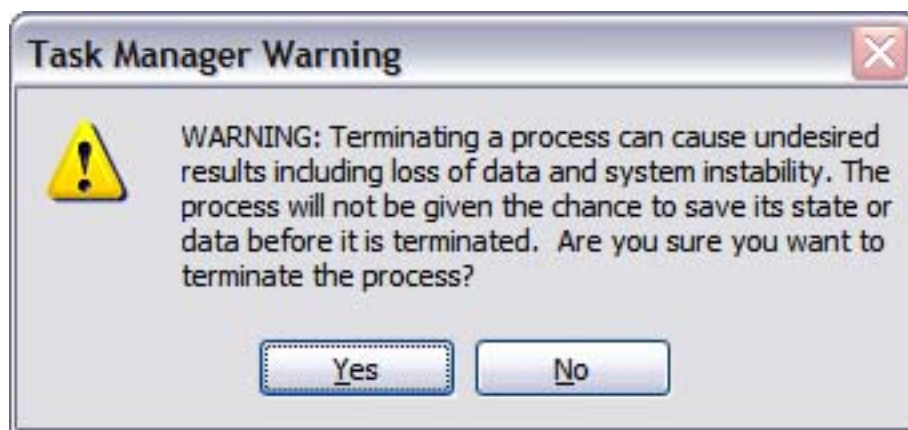
### Error: Please Close all Running Sessions of Microsoft Word

If you have any Microsoft Office processes running when installing ePublisher, including instances of Microsoft Word and Microsoft Outlook, ePublisher displays the following error message.



### To resolve this issue

1. Close any running instances of Microsoft Word.
2. Close Microsoft Outlook.
3. Open Task Manager.
4. Click on the **Processes** tab.
5. Search for `WINWORD.EXE`. You can click on the **Image Name** column to sort the processes alphabetically.
6. ***If there is a WINWORD.EXE process running***, complete the following steps:
  - a. Select `WINWORD.EXE`.
  - b. Click **End Process** to close all running Word processes. Task Manager displays the following warning.



- c. Click **Yes**.
  - d. Close Task Manager, and then proceed with your ePublisher installation.
7. ***If there are no WINWORD.EXE processes running***, proceed with your ePublisher installation.

## Problems with FrameMaker or Microsoft Word

This section helps you troubleshoot issues related to ePublisher interacting with FrameMaker and Microsoft Word.

## **Error: Error Communicating with Adobe FrameMaker**

If you install FrameMaker after installing ePublisher, the ePublisher add-ins for FrameMaker might not be installed.

**To resolve this issue**

1. Make sure all FrameMaker instances are shutdown and no background instances are running.
2. Run a 'Update/Repair' install using the ePublisher Express installer.

**Error: Cannot Duplicate Document**

This error will occur if you install FrameMaker or Microsoft Word after installing ePublisher, the ePublisher add-ins for Word and FrameMaker might not be installed.



### To resolve this issue

1. Make sure all FrameMaker or Word instances are shutdown and no background instances are running.
2. Run a 'Update/Repair' install using the ePublisher Express installer.

## Problems with Contract IDs and Licensing

This section helps you troubleshoot issues related to obtaining, adding, and removing Contract IDs. For more information about Contract IDs, see “Working with Contract IDs”.

### No Contract ID Received

After you purchase ePublisher components, your WebWorks customer service team will e-mail your Contract ID that enables licensing for the products you purchased. If you have not received a Contract ID, send an email to [sales@webworks.com](mailto:sales@webworks.com). For more information about Contract IDs, see “Working with Contract IDs”.

### Error: No Valid License Key Found

You must enter a Contract ID before you can generate output. If you have not entered your Contract ID information, ePublisher displays an error stating that no valid license key was found to enable support for your content authoring tool.

If you have entered a Contract ID but still receive this error, verify you entered your Contract ID information correctly. For more information about Contract IDs, see “Working with Contract IDs”.

### Other Contract ID and Licensing Problems

If you have received your Contract ID and entered your Contract ID into ePublisher but you are having problems with licensing, send an email to [customerservice@webworks.com](mailto:customerservice@webworks.com). For more information about Contract IDs, see “Working with Contract IDs”.

**Note:** ePublisher licensing is flexible, and the WebWorks team can work with you ensure that you have the licensing that is right for you.

# Exploring ePublisher

Understanding the ePublisher Workflow  
Exploring the ePublisher User Interfaces  
Customizing Your ePublisher Workspace  
Specifying General ePublisher Preferences

This section helps you understand the ePublisher workflow, from the tasks that the Stationery designer performs, such as preparing source document templates and creating Stationery, through the tasks that writers perform, such as preparing their source documents for output generation, generating output, and validating their generated output.

This section also provides an overview of each ePublisher user interface and, through the use of sample Exploring ePublisher source documents, projects, and Stationery, helps you understand how you can quickly and easily produce online content that meets required styles and standards

## Understanding the ePublisher Workflow

ePublisher allows your organization to quickly and easily produce online content that meets your organization's styles and standards. With ePublisher, you use the following workflow to quickly and easily generate online content:

- Stationery designers create new or modify existing source document templates and then use ePublisher Designer to create Stationery for writers to use to generate output. For more information, see "Stationery Designers and ePublisher Designer".
- Writers use ePublisher Express and the Stationery created by a Stationery designer to generate and validate online content. For more information, see "Writers and ePublisher Express".
- ePublisher AutoMap can automatically generate and deploy online content using the Stationery created by a Stationery designer and source documents created by writers. For more information, see "Automating Output Generation with ePublisher AutoMap".

## Stationery Designers and ePublisher Designer

If you are a Stationery designer, your first step when you work with ePublisher will be to identify which input formats will be used to author source documents, which types of output need to be generated, and which online features should be included in generated output.

After you identify your input and output formats and the features you want to include in your online content, your next step is to either create source document templates or prepare your existing source document templates for output generation. Stationery designers use source document templates to configure settings and create Stationery for writers to use when generating output. Writers use the source document templates to create content and prepare source documents for output generation. If writers already use source document templates, Stationery designers can simply prepare the existing source document templates for output generation. If writers are not currently using source document templates, the Stationery designer creates a source document template for each content authoring tool used by the authors in the organization. Writers then use the styles and standards defined in the template to prepare their source files for output generation.

For example, if all writers use Adobe FrameMaker and Adobe FrameMaker templates when authoring content, the Stationery designer can take the existing Adobe FrameMaker templates, modify the templates as needed to support online content, and then use the existing templates to create Stationery. If writers use both Adobe FrameMaker and Microsoft Word but are not yet consistently using templates, the Stationery designer creates a new standard set of both Adobe FrameMaker and Microsoft Word templates for writers to use when authoring content.

After the Stationery designer prepares source document templates for the content authoring tools writers use, the Stationery designer uses ePublisher Designer to perform the following tasks:

- Create a Stationery design project using the source document templates.
- Configure settings and options in the Stationery design project that define the look, feel, and behavior for each output target.

For example, the Stationery designer can specify if ePublisher should use the existing styles and formatting in the source documents when generating output to ensure the source documents and online content share the same look and feel. The Stationery designer can also specify that online content have a completely different look and feel than the source documents based on online output design goals and business needs.

- Create Stationery using the settings defined in the Stationery design project. The Stationery defines the style and behavior of generated output, and writers use the settings in the Stationery when they generate output from their source documents.

After the Stationery designer creates the Stationery, the Stationery designer places the ePublisher Stationery and source document template files on a shared network folder for writers to use as they author their content, prepare their source documents for output generation, and generate output.

# Writers and ePublisher Express

With ePublisher, writers use their preferred content authoring tool and the source document templates provided by the Stationery designer to create content and prepare their source documents for output generation. Writers can use Markdown ++, Microsoft Word, structured or unstructured Adobe FrameMaker, and DITA-XML content authoring tools to author content. Writers format, or tag, their source documents by applying styles and formats, and then ePublisher uses this information to generate the appropriate output based on the settings the Stationery designer specified in the Stationery. Writers do not need to worry about output design. Instead, writers can focus on creating the content users need.

When writers are ready to generate online content, writers use ePublisher Express to perform the following tasks:

- Create an ePublisher Express project based on Stationery created by a Stationery designer.
- Add the source documents they want to use to generate output to the ePublisher Express project.
- Generate output. The output that writers generate adheres to the styles and standards defined by the Stationery designer in the Stationery. Writers can generate output on demand. ePublisher can also automatically generate output based on a schedule if ePublisher AutoMap is implemented.
- Deploy generated output. Writers can deploy their generation to a Web site or to a central location from which a product build can obtain the files. ePublisher can automatically deploy output if ePublisher AutoMap is implemented.
- Check the generated output into a version control system or copy the generated output to a central archive location.

Although writers typically use the Stationery provided by the Stationery designer without modification, writers can use ePublisher Express to perform some customizations of output target settings if they have target setting modification permissions. For example, writers can customize company information such as company name, phone number, and Web site information in their generated output if they have target setting modification permissions. If writers need to customize any target or project settings, they should first ensure they have appropriate target setting modification permissions. After they make any target setting customizations, they regenerate their output and verify their target setting customizations before deploying their final output.

Most writers prefer to use ePublisher Express to perform an initial output generation early in their project cycle. This allows writers to quickly and easily verify that they are formatting their source documents correctly and also confirm that their generated output has the appearance and features they want. After an initial

verification of the generated output early in the project cycle, writers continue to add content to their source documents, then regenerate output on a periodic basis, such as once a week, as the project progresses.

If ePublisher AutoMap is implemented, ePublisher AutoMap can be configured to automatically generate output for writers on a regular schedule. For example, ePublisher AutoMap can be configured to automatically generate output for a project every night. When writers arrive at work the next day, writers can quickly and easily verify that the latest content they added to their source documents is formatted correctly and displays appropriately in their generated output. Periodic output generation using the latest version of source documents and Stationery provides the following benefits:

- Allows writers to quickly and easily confirm that they are formatting their source documents correctly and feel confident that their generated output always conforms to the standards and styles defined in their project Stationery by the Stationery designer
- Reduces the amount of time writers spend doing a quality assurance review for their generated output at the end of a project cycle
- Helps eliminate post-generation editing and processing, which save organizations time and money
- Ensures that the end of the project cycle is smooth and hassle-free

ePublisher allows writers to quickly and easily produce high-quality output according to specifications each time they generate output.

## **Automating Output Generation with ePublisher AutoMap**

If you implement ePublisher AutoMap, you can configure ePublisher AutoMap to automate your output generation using Stationery created by a Stationery designer and source documents created by writers.

## **Exploring the ePublisher User Interfaces**

The WebWorks ePublisher Platform provides the following user interfaces:

- ePublisher Express user interface, which writers use when generating output
- ePublisher Designer user interface, which Stationery designers use when creating and updating Stationery

- ePublisher AutoMap user interface, used to automate output generation and integrate the output generation process with source control systems and product build systems

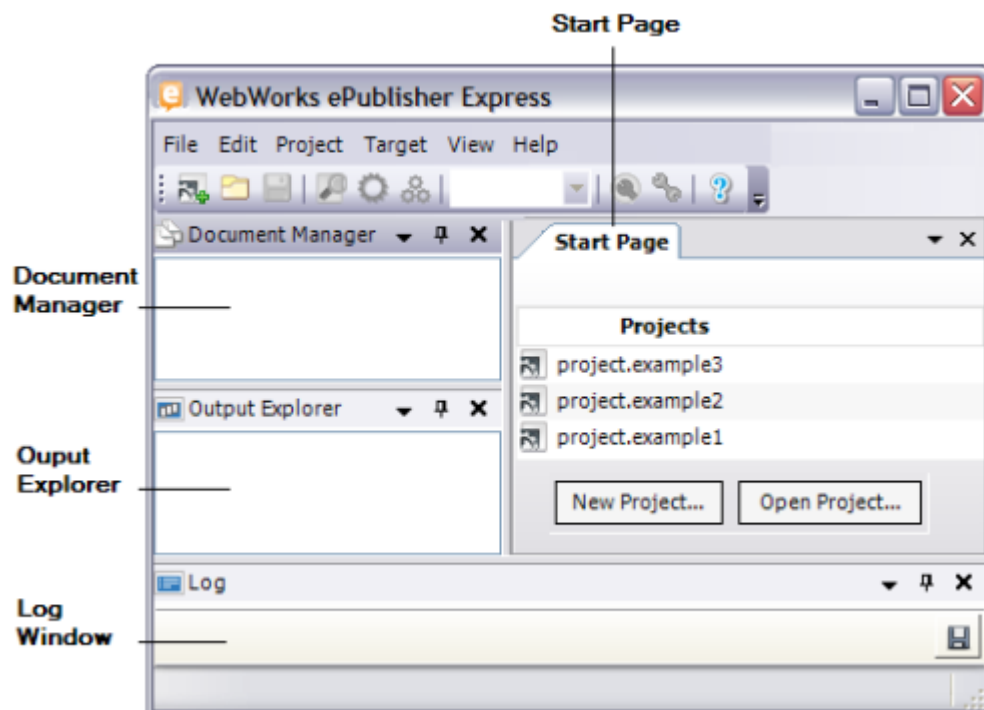
This section provides an overview of each of these ePublisher user interfaces.

## Exploring the ePublisher Express User Interface

The ePublisher Express user interface includes the following windows:

- Start page. For more information, see “Understanding the Start Page”.
- Document Manager. For more information, see “Understanding Document Manager”.
- Output Explorer. For more information, see “Understanding Output Explorer”.
- Log Window. For more information, see “Understanding the Log Window”.

The following figure shows the ePublisher Express user interface.

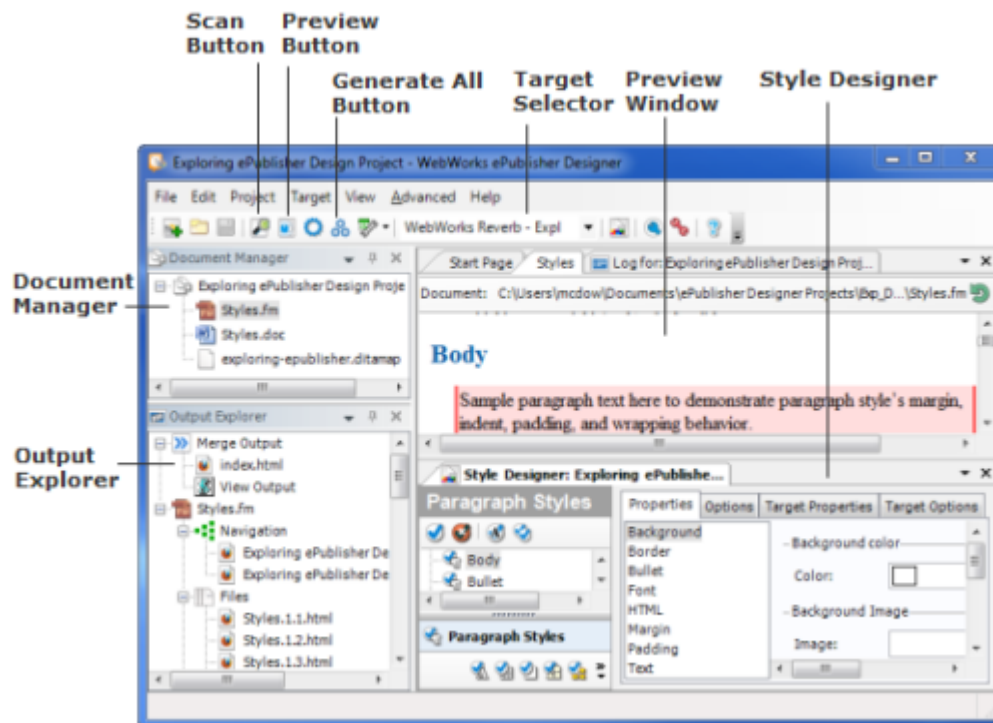


# Exploring the ePubisher Designer User Interface

The ePubisher Designer user interface includes the following windows:

- Start page. For more information, see “Understanding the Start Page”.
- Document Manager. For more information, see “Understanding Document Manager”.
- Output Explorer. For more information, see “Understanding Output Explorer”.
- Log Window. For more information, see “Understanding the Log Window”.
- Style Designer. For more information, see “Understanding Style Designer”.
- Preview window. For more information, see “Understanding the Preview Window”.

The following figure shows the ePubisher Designer user interface.



## Understanding the Start Page

The Start page is available in both ePubisher Express and ePubisher Designer. The Start page lists the most recently opened ePubisher projects. You click on a project

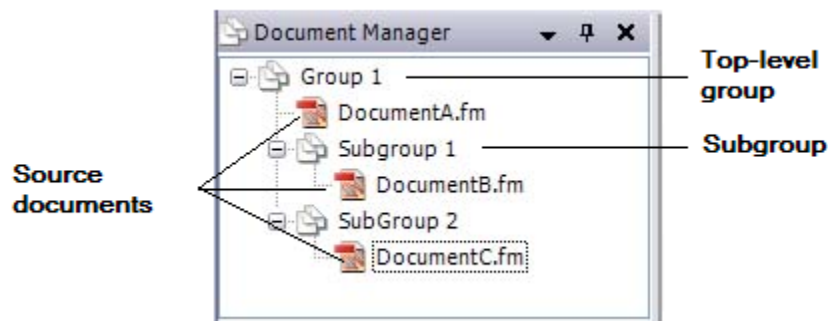
name on the Start page to open the associated project file. You can also specify the number of projects you want to display on the Start page. For more information about specifying the number of projects you want to display on the Start page, see “Specifying General ePublisher Preferences”.

The following figure shows the Start page.



## Understanding Document Manager

Document Manager is available in both ePublisher Express and ePublisher Designer. Document Manager allows you to organize the source documents in your project. Within Document Manager, you can add, remove, and rearrange the groups and source documents in your project. The following figure shows Document Manager.



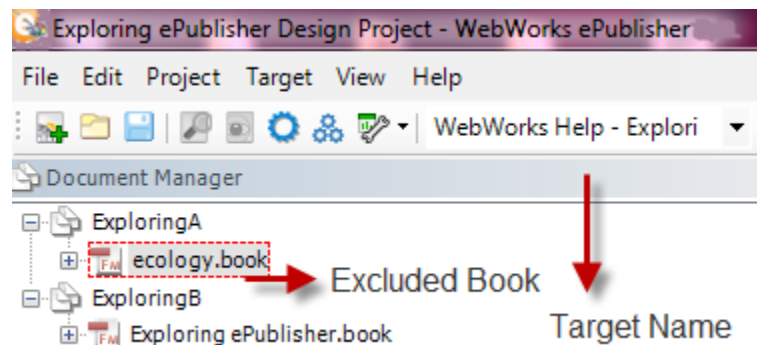
## Including or Excluding Files



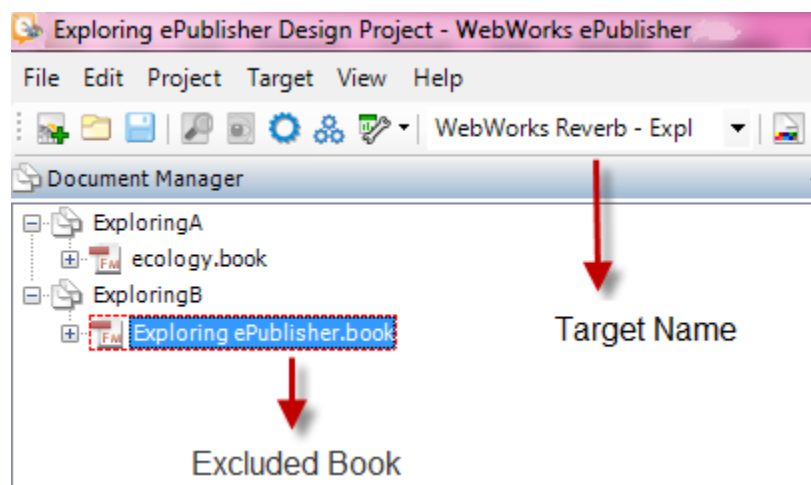
ePublisher allows you to include/exclude documents from processing on a target by target basis. This capability is available to all ePublisher users, regardless of their source authoring format. Users working with Adobe FrameMaker book files will see that ePublisher scans default include/exclude values from their source documents.

To include or exclude a document from processing, right-click on any file or book. The context menu will appear with the Include/Exclude option. Clicking this item will reveal three choices: **Include**, **Exclude** and **Use Document Value**. Clicking Exclude will create a red dotted line around the file that will indicate that this source file will not be created in the output file. Clicking include will show as normal and if it is not changed in the FM source, will be included. The Use Document Value will take whatever is set in FrameMaker and if already set to Exclude in the source will show the dotted red lines around the source documents.

**Note:** Include/exclude settings are configured on a per Target. For example in this project's WebWorks Help target, `ecology.book` (along with all child files) is excluded:



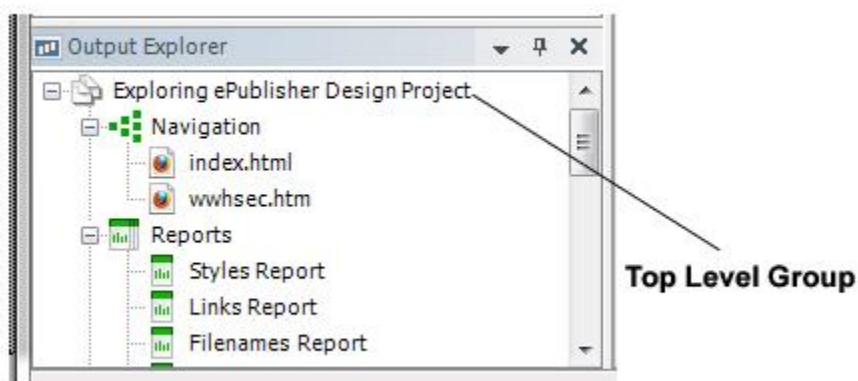
In the same project's WebWorks Reverb target, `ecology.book` book is included for processing while `Exploring ePublisher.book` (and child files) is excluded.



## Understanding Output Explorer

Output Explorer is available in both ePublisher Express and ePublisher Designer. Output Explorer displays all of the files generated by ePublisher.

The following figure shows Output Explorer.



The output files displayed in the Output Explorer depend on the item you select in Document Manager. Output Explorer displays items as follows:

- ***If you have a source document selected in Document Manager,*** ePublisher displays the source document group in Output Explorer. The source document group displayed in Output Explorer contains the Files group, which lists all of the generated topic files, the Images group, which lists all of the generated images for your output, and the Reports group, which lists Styles, Links, Accessibility, Filenames, and Topics reports.
- ***If you have a top-level group selected in Document Manager,*** ePublisher displays the top-level group in Output Explorer. The top-level group displayed in Output Explorer contains the Navigation group, which lists the entry-point file for the generated output based on the active target selected in the project, and the Reports group, which lists Styles, Links, Accessibility, Filenames, and Topics reports.
- ***If you have a subgroup selected in Document Manager,*** ePublisher displays the subgroup in Output Explorer. The subgroup contains the entry-point file for the generated output.

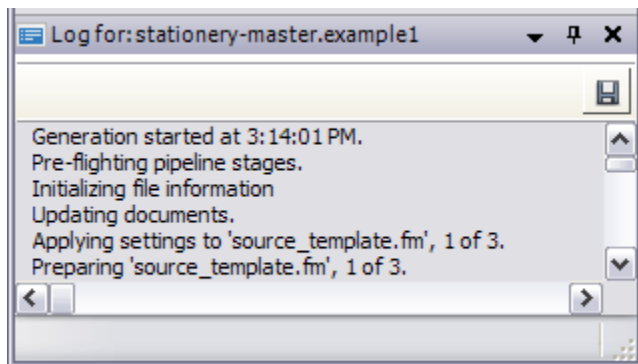
***If you have more than one top-level group in Document Manager,*** in addition to displaying the top-level group you select in Document Manager in Output Explorer, ePublisher also displays a Merge Output group in Output Explorer. The Merge Output group displays the merged entry-point file created from each top-level group entry point file. You can use the merged entry-point

file ePubublisher automatically creates when you have more than one top-level group in Document Manager to create merged help systems.

For more information about merged help systems, see “Merging Top-level Groups (Multivolume Help)”.

## Understanding the Log Window

The Log Window is available in both ePubublisher Express and ePubublisher Designer. The Log Window displays the log ePubublisher creates when generating output. When ePubublisher creates a log during output generation, you can see the status of the output generation process and any errors generated. You can also quickly and easily see which pipelines ePubublisher processed, which settings ePubublisher applied, and which files ePubublisher parsed. For more information about viewing or working with logs, see the “Working with Output Log Files”. The following figure shows the Log Window.

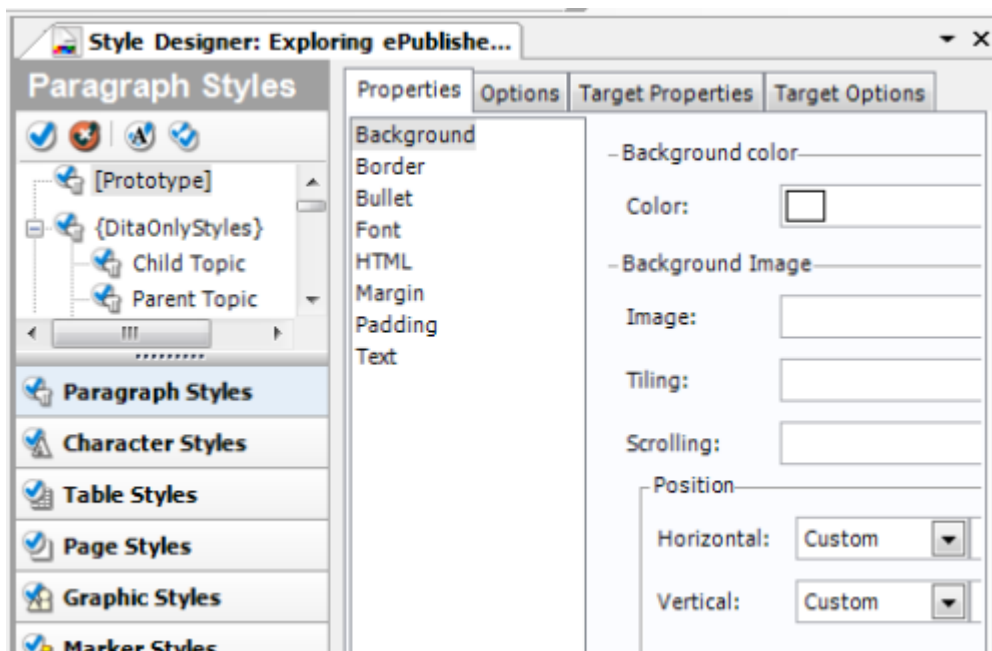


## Understanding Style Designer

Style Designer is available in only ePubublisher Designer. ePubublisher intelligently discovers the styles in the source documents and presents a list of these styles in Style Designer. Stationery designers then define how output should be generated by specifying properties and options for each style. Stationery designers use Style Designer to define how paragraphs, characters, tables, and images display in generated output, including the color or font of a paragraph style, the style of a table border, the layout of a page, and the file format of converted images. Stationery designers can also specify other aspects of generated output, such as page layout and when topic pages are created.

ePubublisher uses the styles in source documents along with the settings the Stationery designer defines in the Stationery to generate output. Using styles in source documents and Stationery settings allows precise control over the appearance and behavior of generated output.

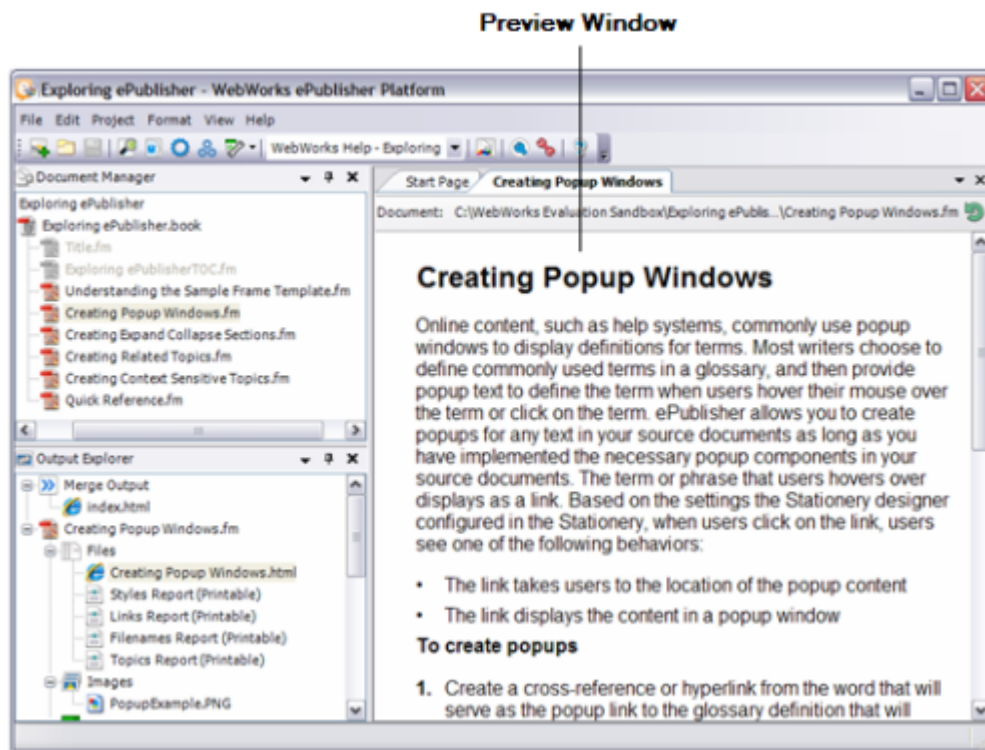
The following figure shows Style Designer.



## Understanding the Preview Window

The Preview window is available in only ePublisher Designer. When designing Stationery, Stationery designers can use the Preview window to quickly see how modifications made to styles and project settings affect the appearance of generated output. You can generate a preview of output from a source file in ePublisher Designer when you select a source document in Document Manager. However, some online content features, such as popup windows, links, and conditions, are not displayed or active in the Preview window.

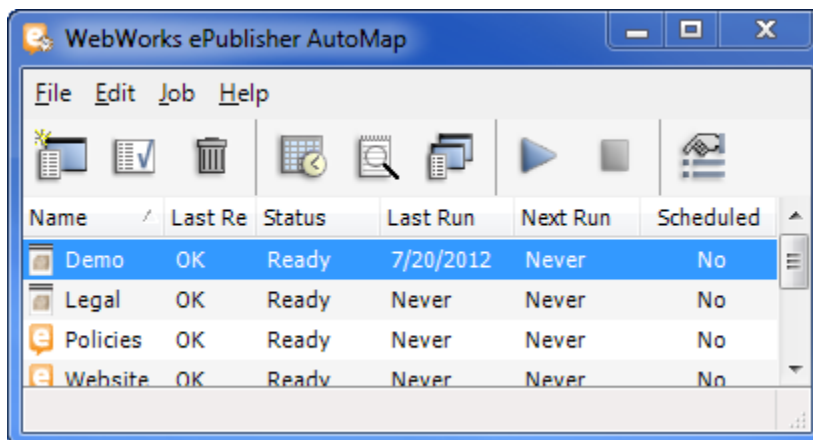
The following figure shows the Preview window.



For more information about using the preview window in ePublisher Designer when designing Stationery, see "Previewing the Output from a Source File".

## Exploring the ePublisher AutoMap User Interface

The ePublisher AutoMap user interface allows you to create, edit, and schedule ePublisher AutoMap jobs. The following figure shows the ePublisher AutoMap user interface.



For more information about using ePublisher AutoMap, see “Scheduling and Integrating Processes with AutoMap”.

## Customizing Your ePublisher Workspace

By default, the ePublisher user interface comes preset with certain toolbar icons and window settings. ePublisher gives you extensive control over the appearance of the user interface by allowing you to customize the display of windows and toolbars.

When you open ePublisher for the first time, Document Manager and Output Explorer are docked and the Log Window is undocked. When a window is undocked, it displays as a tab in the sidebar, and it will auto-hide unless you hover over the tab in the sidebar. When you hover over an undocked window, the window displays.

You can customize the display of windows in the user interface to suit your needs. In ePublisher, you can move Document Manager, Output Explorer, and the Log Window to different locations within the user interface or make them into floating windows. When you dock a window, it becomes stationary within the user interface and is always visible.

You can rearrange docked windows by moving the window to a new location within the user interface. For example, you can move the Output Explorer into a new window pane next to the Start page. However, the Start page cannot be moved. The Start page serves as the central point from which all the windows are arranged. To move a docked window within the user interface, click on the title bar of the window and then drag the window to a new location. To undock a window, click on the pin icon in the upper right corner of the window.

In addition to rearranging docked windows, you can add, remove, customize, or create buttons on tool bars. You can also create your own custom toolbars. By customizing your toolbars and buttons, you can create a workspace that fits your preferences and work style.

Whenever you make changes to the user interface by moving windows, changing window dock settings, or customizing toolbars, the changes take effect for each subsequent project you create or open. For example, if you dock Document Manager, Output Explorer, and the Log Window and add customized buttons to the toolbar, these settings will become the default settings for each project you open.

## Specifying General ePublisher Preferences

You use the **General** tab of the Preferences window to specify ePublisher preferences, such as the number of recent projects to display on the Start page,

whether to automatically scan source documents when you add them to Document Manager, and where to store user-created formats. You can also reset the user interface toolbar and window dock positions.

## To specify general ePublisher preferences

1. On the **Edit** menu, click **Preferences**.
2. Specify preferences for each setting. For more information about settings and options, click **Help**.



# Miscellaneous ePublisher Windows

Add New Target Window  
Conditions Window  
Cross Reference Rules Window  
Deployment Configuration (Name) Window  
Deployment Editor Window  
Documents Window  
Edit Target Window  
File Mapping Editor Window  
Folder Deployment Editor Window  
Target Settings Window  
Job Info Window  
License Information Window  
Main ePublisher AutoMap Window  
Main ePublisher Window  
Manage Targets Window  
Merge Settings Window  
New ePublisher AutoMap Job Window  
New Project Wizard  
Preferences Window  
Project Settings Window  
Save As Stationery Window  
Script Editor Window  
Target Configuration Window  
Target Selection Window  
User Information Window  
Variables Window  
WebWorks ePublisher Preferences Window  
WebWorks Licensing Info Window

The topics in this section identify the windows in ePublisher Express, ePublisher Designer, and ePublisher AutoMap. Each topic provides information about a window or tab and the related fields.

## Add New Target Window

This window allows you to create a new target in your project. A **target** is based on an output format, such as WebWorks Reverb 2.0, PDF - XSL-FO, Microsoft HTML Help, or Sun JavaHelp. Each target also has target settings to customize that target. A project can have multiple targets based on the same output format, such as WebWorks Reverb 2.0. You may add multiple targets in projects that generate multiple versions of the online content based on OEM partner agreements or multiple versions of the product. Each target can have different variable values and conditions set to generate unique output. The fields are defined as follows:

### Target Name

Specifies the name for the target. You can specify a target name that is different from the output format name. For example, you can name a target `OnlineHelp`, to identify the purpose of the generated output. You can also

name a target to match the output format name, such as `WebWorks Help`, as long as you have only one target in the project using that output format.

### Format Type

Specifies the output format for the target, such as WebWorks Reverb 2.0, PDF - XSL-FO, or Sun JavaHelp 2.0.

## Conditions Window

This window allows you to specify conditions to use when generating output.

**Conditions** allow you to show or hide information in your generated output.

**Conditional text** is any content that has a condition applied to it.

You apply conditions to the content in your source documents, and then you define the visibility for those conditions in your output. Your project lists the conditions in your source documents and allows you to specify whether to show or hide each condition. You can also use the condition settings specified in your source documents. By default, ePublisher uses the condition settings specified in your source documents. You can modify the settings for the conditions for each target in your project. This window provides the following tabs:

- “Classic Tab”
- “Expressions Tab (FrameMaker Only)”

## Classic Tab

This tab allows you to specify the visibility of conditions in your output. The columns are defined as follows:

### Name

Specifies the name of the condition.

### Visibility

Specifies whether the content with the condition applied to it is included in your output. The values for this setting are defined as follows:

Value	Description
<b>Visible</b>	Displays the content in your output.
<b>Hidden</b>	Excludes the content from your output.
<b>Use document value</b>	Uses the state of the condition specified in your source documents to determine whether the content is displayed in your output.

## Pass Through

Puts the content directly in your output without processing or transforming the output. This setting allows you to put HTML code in your source documents and have that code put directly in your output. The content with this type of condition applied is not transformed, so the special characters in HTML coding remain unchanged. For example, the `<` is not transformed to `&lt;` and the `>` is not transformed to `&gt;`.

## Expressions Tab (FrameMaker Only)

This window allows you to manage the expressions related to conditions in source documents authored in FrameMaker version 8.0 or higher. In these later versions of FrameMaker, FrameMaker allows you to define logical expressions to use when applying visibility settings to conditional text in a FrameMaker source document. Using FrameMaker conditional settings, you can create conditional tags and build complex Boolean expressions for defining output filters when authoring in both structured and unstructured modes. For more information about conditional expressions in FrameMaker, refer to the FrameMaker documentation.

**Note:** The use of Expressions is limited to FrameMaker input.

The fields are defined as follows:

### Use Document Value

Specifies whether to use the value specified in your source documents to determine whether the content is included in your generated output.

## Cross Reference Rules Window

This window allows you to add, edit, and delete cross-reference formats for your project. A **cross-reference format** is a combination of text and code that defines how ePublisher displays cross-references in your generated output.

Cross-references help users navigate through your content. ePublisher automatically converts cross-references to links in the online content. However, you often want cross-references in your online content to use a different format from your printed content. For example, you usually include page numbers only in your printed content. ePublisher enables you to add, edit, and delete cross-reference formats for your online output.

The columns are defined as follows:

### Document Type

Specifies the content type of the source document that contains the cross-reference formats to modify.

### Name

Specifies the name of the cross-reference format. ePublisher obtains the cross-reference formats displayed in this column from your source documents. The **Name** column shows either the cross-references formats used or a set of default cross-reference formats depending on the type of source document you are converting. These formats are hidden building blocks or codes that come directly from your source document. If you would like to find out how to interpret them, see the Help documentation for the application that was used to create the source document.

### Use Document Value

Specifies whether to use the value specified in your source documents instead of the specified value in the ePublisher project.

### Value

Specifies the replacement content and building block code that ePublisher uses when generating your output. For example, a FrameMaker cross reference format may be `<$paratext>` to provide the text of the linked heading paragraph as the cross reference. For more information about the building block of text and code that defines your cross-reference format, see the documentation for the authoring tool you used when you created your source documents. For example, if you created your source documents in Adobe FrameMaker, see the Adobe FrameMaker documentation.

To edit the cross-reference format, double-click the value to edit, and then edit the cross-reference format in the **Replacement** field in the Edit Cross Reference window. The **Value** column shows the corresponding online cross-reference formats. The first time you open this window for a project, the

formats in the **Value** column match the formats in the **Name** column so that ePublisher matches the content of your generated output to your source documents.

## Deployment Configuration (Name) Window

This window allows you to specify a name for an output destination when you configure an output destination for a target.

The fields are defined as follows:

### Name

Specifies the name for the output destination. Ensure you specify a descriptive name for the output destination. When you work with output destination, you can only see the name of the output destination. You will not be able to see the actual path you specified to the output destination. Type a descriptive name for the output destination that allows you to easily identify each output destination you specify.

## Deployment Editor Window

This window allows you to specify deployment locations for your generated output. A **deployment location** is a folder where you deploy generated output files. A deployment location can be a folder on your local computer or on a network share.

The columns are defined as follows:

### Name

Specifies the name of the deployment location.

### Deployment Type

Specifies the type of container to which the generated output files are deployed. ePublisher currently supports only folders as deployment locations.

## Documents Window

This window allows you to specify the source documents the ePublisher AutoMap job uses when generating output create groups and to create groups for source documents. This window also allows you to specify a script to use when obtaining source documents for a group. For example, to obtain source documents from a version control or content management system, you can specify a script that

obtains and prepares source documents based on the parameters you define in the script.

You must create a top-level group before you can add source documents to the ePublisher AutoMap job. A top-level group is the primary container for source documents, subgroups, and the entry-point file for the output. After you create a top-level group, you can add source documents to the job, and you can also create subgroups and add source documents to subgroups as needed.

The fields are defined as follows:

### Script to retrieve documents

Specifies the script to run to retrieve source documents. Type or paste the script to use to retrieve source documents into the text field or click **Edit Script** to use the Script Editor window to write your script. The script you specify runs before the ePublisher AutoMap job generates output, which ensures that the ePublisher AutoMap job always uses the most current version of the source documents.

## Edit Target Window

This window allows you to edit the name of a target. A **target** is the specific type of output you want to produce using your source files and project settings, and targets are based on an output format, such as WebWorks Reverb 2.0, PDF - XSL-FO, or JavaHelp. Targets include all of the project settings you specify for each output format included in your project when you configure your project.

### Target Name

Specifies the name of the target. You can specify a target name that is different from the output format name. For example, you can name a target based on the WebWorks Reverb 2.0 output format `OnlineHelpFormat`, or you can name a target based on the WebWorks Reverb 2.0 output format `WebWorks Reverb 2.0 format`.

### Format Type

Displays the output format for the target. For example, the format type may be WebWorks Reverb 2, PDF - XSL-FO, or WebWorks Help.

## File Mapping Editor Window

This window allows you to create a file mapping. A **file mapping** is an association between a file extension and an ePublisher adapter. An ePublisher **adapter** is an ePublisher component that bridges the gap between the application in which the source document was created and ePublisher. ePublisher currently provides adapters for Helper (Markdown++), Microsoft Word, Adobe FrameMaker, and XML.

The columns are defined as follows:

### **File extension**

Specifies the file extension to associate with an ePublisher adapter.

### **Adapter**

Specifies the ePublisher adapter to use for the file extension.

## **Folder Deployment Editor Window**

This window allows you to edit deployment locations. A **deployment location** is a folder where ePublisher deploys output files from a project. A deployment location can be a folder on your local computer or on a network share.

The fields are defined as follows:

### **Name**

Specifies the name you of the deployment location.

### **Directory**

Specifies the folder where to deploy your output files.

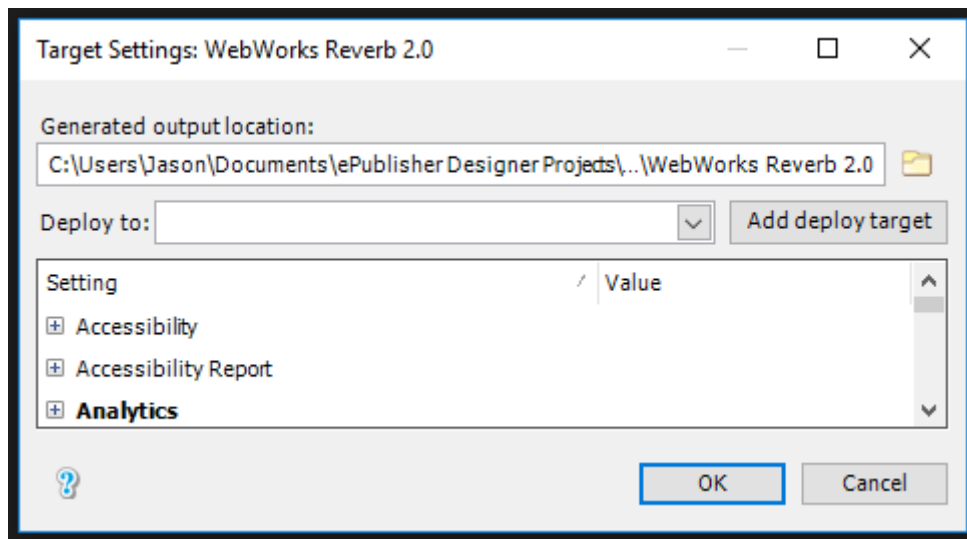
## **Target Settings Window**

This window allows you to specify the settings to use for a selected output target. You can reach this window by selecting the menu: **Target > Target Settings**.

The target settings are grouped according to function.

## **Generated output location**

Specifies the location where ePublisher saves the generated output for the selected target.



**Note:** We do not recommend changing the generated output location as it creates issues with maintaining project in the future.

## Deploy to

Specifies the name of the deployment location where ePublisher deploys the output generated for the target if you have created and selected a deployment location. Specify a deployment location by selecting a deployment location you created from the list.

## List of Target Settings

The main component of the **Target Settings Window** is the list of available target settings that can be configured for a given target in your project. See "Target Settings Reference" for a complete list of available settings.

## Job Info Window

This window allows you to specify information about an ePublisher AutoMap job. You can also specify if any scripts run before or after the job runs and generates output.

The fields are defined as follows:

### Job name

Specifies the name of the job.

### Choose ePublisher project or stationery



Specifies the ePublisher project or Stationery for the job. You specified the ePublisher Designer project or Stationery when you created the job. To change the ePublisher project or Stationery specified for the job, type a path to the new ePublisher project or Stationery for the job or click the **Folder** icon to browse to and select the new ePublisher project or Stationery for the job.

### **Pre-build**

Specifies the pre-build script to run before ePublisher AutoMap generates output. Type or paste the pre-build script into the text field or click **Edit Script** to use the Script Editor window to write your script.

### **Post-build**

Specifies the post-build script to run before ePublisher AutoMap generates output. Type or paste the post-build script into the text field or click **Edit Script** to use the Script Editor window to write your script.

## **License Information Window**

This window allows you to view the adapters for which you are licensed. Adapter licensing information is specified in your Contract ID. ePublisher uses adapter license keys, or activation codes, to enable ePublisher functionality, and ePublisher automatically retrieves and enables these codes based on your Contract ID. ePublisher licenses, or activation codes, do not display in the ePublisher user interface. ePublisher components automatically communicate with the ePublisher licensing server periodically to refresh and update ePublisher activation codes as needed.

If you have a valid Contract ID for one version of the ePublisher product, when a new version of ePublisher releases, you can continue to use your same Contract ID when you upgrade to the new version of the product. You can also continue to use your same Contract ID if you have to uninstall and then re-install a version of ePublisher.

The fields are defined as follows:

### **Product**

Specifies the name of the ePublisher adapters for which you are licensed. Your Contract ID specifies the ePublisher adapters for which you are licensed.

### **Expires**

Specifies the license expiration date for the licensed ePublisher adapters. ePublisher components automatically communicate with the ePublisher licensing server to renew activation codes as needed before they expire based on the terms of your current contract.

“Working with Contract IDs”

“Viewing Licensing and Contract ID Information”

“Obtaining Contract IDs”

“Entering Contract IDs”

“Managing Licensing in Environments without Internet Connectivity”

“Updating Licensing”

“Deactivating Licensing”

“Problems with FrameMaker or Microsoft Word”

## **Main ePublisher AutoMap Window**

This window allows you to view information about ePublisher AutoMap jobs.

The columns are defined as follows:

### **Name**

Specifies the name of the ePublisher AutoMap job.

### **Last Result**

Specifies the results from the last time the job ran. You can see if the job completed successfully, or if any errors occurred the last time the job ran.

### **Status**

Specifies the current state of the job, such as whether the job is currently running or ready to be run.

### **Last Run**

Specifies the date and time the job last ran.

### **Next Run**

Specifies the date and time the job is scheduled to run again.

### **Scheduled**

Specifies whether the job is currently scheduled to run.

# Main ePublisher Window

The WebWorks ePublisher Platform (ePublisher) is a powerful, comprehensive solution that delivers cost-effective processes for efficiently publishing and maintaining online and print information. ePublisher gives you the flexibility to deliver content from multiple types of source documents, such as Adobe FrameMaker, Microsoft Word, and DITA, in virtually any output format you need. The open, standards-based architecture provides a powerful engine that does not lock your content in a proprietary format that can become outdated as tools and standards change.

The main ePublisher window connects you to the many powerful features found in ePublisher. This window includes the following subwindows:





- "Document Manager"
- "Output Explorer"
- "Start Page"
- "Log Window"
- "Style Designer"
- "Document Designer"

## Document Manager

This window allows you to view and organize source documents in your ePublisher project. From the Document Manager window, you can add, remove, and rearrange groups, subgroups, and source documents.

In Document Manager, you can create an organizational structure for your source documents that includes groups and subgroups. A **group** is the highest level of organization of generated material. The group indicates how ePublisher outputs source documents. Each primary-level group created in Document Manager generates a separate Help system output. **Subgroups** organize your help project source documents into a logical and helpful system.









You can include multiple source documents of differing file types within Document Manager. Each file type has a unique and identifiable icon in the Document Manager window.

<b>File Type Icon</b>	<b>File Type Description</b>
	ePublisher Project file
	Adobe FrameMaker file
	DITA 1.0 or DITA 1.1 file
	Microsoft Word 98 to Microsoft Word 2007 file

“Understanding Document Manager”

## Output Explorer

This window allows you to view and open generated output files from ePublisher Designer. The Output Explorer window displays output files based on whether you select a source document or the project file in the Document Manager window. If you select the project file, Output Explorer displays the project file along with its navigation information and any project reports. If you select a source document, Output Explorer displays the individual files, images, and reports for the document.

File Type Icon	File Type Description
	ePublisher project directory
	Output files directory
	Images directory
	Output file that has been deleted
	Navigation directory
	Reports directory
	Single report
	Printable report

“Understanding Output Explorer”

“Understanding Document Manager”

## Reports

Once you have generated output or generated a report from ePublisher, you can double click each of the types of report to bring up a tab to the right of the Document Manager and Output Explorer that will list the various occurrences of filename markers, topic alias markers, and other components of your source document.

**Note:** If you do not have anything to report, the tab will be blank. For example, if you do not have any topic aliases, there will be nothing in the topics report.

## Reports - printable

Provides an xml list of the reports that are generated with the output.

# Start Page

This window lists all recently opened ePublisher projects. Click any of the project names listed to open the project file. Click **New Project** to create a new project from this window. Click **Open Project** to navigate and open an existing project not already listed on the **Start Page**.

To specify the maximum number of projects that ePublisher shows in **Start Page** or clear the currently displayed projects, select **Preferences** on the **Edit** menu. Enter the number of projects to display in the **Recent projects to remember** field. Click **Clear List** to remove the current list of projects shown on **Start Page**.

# Log Window

This window shows the status of project and report generation. The Log window identifies where errors occur when generating output files or reports and displays a detailed error message description. Links in this window are highlighted in blue and if clicked they will open that URL in the default browser.

When you generate output, ePublisher converts a source document to a target by breaking the process into a series of steps, also known as **stages**. Each stage performs a specific action in the process. ePublisher groups these steps into **pipelines** of related stages.

Each time you generate output or reports, ePublisher Designer creates an external log file, `generate.log` in the `Logs` directory of the selected project file. Send this log file to the ePublisher support team if you submit a support request. Click **Save As** in the Log window to save the current log contents as a text file in another directory.

The Log window automatically displays when you open the main ePublisher Designer window. If you close the Log window and need to reopen it, select **Log Window** on the **View** menu.

# Preview Window

This window shows a preview of how a source document will display when you generate output. You can modify output formatting in the Preview window from Document Designer. Document Designer allows you to make modifications only at the paragraph or table level, which means each modification you make applies only to the selected paragraph or table.

**Note:** The Preview window gives you an idea of how your project settings affect the appearance of generated output. However, this window does not display some output features or shows them as inactive, such as popups, links, and conditions.

After you modify properties in Document Designer, click **Refresh** at the top of the Preview window to view an updated preview. If you modify your original source documents, you must scan the document again to see changes in the Preview window

“Understanding the Preview Window”

“Customizing Your ePublisher Workspace”

## Document Designer

This window allows you to override the appearance of a paragraph or table in generated output. If there is an issue in the output and the designer does not have time to make the change in the Stationery, or if you did not set all the styles needed in the source documents and you do not have time to fix it, use Document Designer to put style overrides in place for specific paragraphs or tables. Select the **Allow user style overrides** project setting to allow style overrides.

**Note:** ePublisher does not keep changes made through Document Designer if you save a project as Stationery.

Document Designer works in conjunction with the Preview window. To preview a document, select a single document in the Document Manager, and then select **Generate Preview** from the **Project** menu. Select a paragraph or table in the Preview window and make modifications in Document Designer. Document Designer fields appear inactive until you open the Preview window and select text in the source document.

After modifying the selected content in Document Designer, click refresh in the Preview window to see how modifications look in the generated output.

**Note:** Changes made using Document Designer override any others modifications in ePublisher. For example, if you alter a paragraph in Document Designer to have a red background, but you later decide to change the style the paragraph uses in Style Designer to have a blue background, the Document Designer modifications take precedence.

To clear changes made in Document Designer, select **Clear Document Style Overrides** on the **Edit** menu.

### Style type

Indicates what style was applied to the paragraph or table selected in the Preview window. ePublisher populates this field with style name specified in your source document.

### Apply style

Allows you to select a different style to apply to the paragraph or table selected in the Preview window. ePublisher imports all styles in your original source document when you generate project files.

## Style Designer

This window allows you to specify the appearance and functionality of online content, including paragraphs, characters, tables, page layouts, images, table of contents levels, popups, and related topics. ePublisher Designer builds Stationery based on the settings you specify in Style Designer.

### Font Family Picker Window

This window allows you to assign a font or font family to a paragraph style in your project. From Style Designer, select the paragraph style to assign a font or font family and click the **Ellipses** button in the **Family** field. Select an installed font, a font family, or a custom font and click the **Arrow** button to move the font or font family to the **Selected** field. To remove a font or font family from the **Selected** field, select the font in that field and click "X" at the top of the window.

Consider specifying a generic font family, such as sans-serif, rather than a specific font. Many browsers and help systems use only fonts installed on the user's computer. If you indicate a specific font, a user who does not have that specified font installed does not see the help output correctly. By selecting a generic font family, the user does not have to have the exact font specified, only a font from that family.

If you specify multiple fonts, separated by commas, the user's browser displays the first available font in the list. For example, if you specify `Verdana, Arial, Helvetica, sans-serif` and the user does not have the Verdana font installed, the browser displays in the Arial font for the help output instead.

## Manage Targets Window

This window allows you to create and manage targets in your project. A **target** is based on an output format, such as WebWorks Reverb 2.0, PDF - XSL-FO, or Sun JavaHelp. Each target also has target settings to customize that target. A project can have multiple targets based on the same output format, such as Microsoft HTML Help. You may add multiple targets in projects that generate multiple versions of the online content based on OEM partner agreements or multiple versions of the product. Each target can have different variable values and conditions set to generate unique output. The columns are defined as follows:

### Target Name

Specifies the name for the target. You can specify a target name that is different from the output format name. For example, you can name a target



`OnlineHelp`, to identify the purpose of the generated output. You can also name a target to match the output format name, such as `WebWorks Reverb 2.0`, as long as you have only one target in the project using that output format.

## **Format Type**

Specifies the output format for the target, such as WebWorks Reverb 2.0, PDF - XSL-FO, or Sun JavaHelp 2.0.

# **Merge Settings Window**

This window allows you to specify custom settings for merged, or multivolume help. ePublisher only enables this window if you are working with an output format that supports merged, or multivolume help systems. Not all output formats support merged, or multivolume help. WebWorks Reverb 1 and 2, Eclipse Help, Microsoft HTML Help, and WebWorks Help are the output formats that currently support merged, or multivolume help systems. If you are working with an output format that does not support merged, or multivolume help, this window is disabled.

The fields are defined as follows:

## **Merge Title**

Specifies the name displayed in the title bar of the help.

## **Hierarchy**

Lists all the top-level groups in your project. This group hierarchy is used to define the hierarchy displayed in the table of contents for your multivolume help system. To reposition any of your top-level groups in the table of contents hierarchy, select the group to move and drag the group to a new location in the hierarchy.

## **Table of contents title**

Specifies the name displayed for the selected top-level group when that group is displayed in the table of contents for your merged, or multivolume help system. The group name specified in Document Manager defines the name of the folder that contains the generated output files in that group. Change the name displayed for the top-level group in your merged, or multivolume help system by selecting a top-level group and then typing a new name in this field. By default, the name displayed in the table of contents is the name of the top-level group specified in Document Manager.

## **Group context**

Specifies the help context to use when generating merged, or multivolume help that includes context-sensitive help. In WebWorks Help, you need to

include this context and the TopicAlias value in the help call to display the correct help topic.

**Note:** The group context must be unique so that when topic IDs are duplicated in different help sets, the context sensitive reference will refer to the correct help location. For more information, see "Opening Context-Sensitive Help in WebWorks Help using Standard URLs".

## New ePublisher AutoMap Job Window

This window allows you to create a new ePublisher AutoMap job. A **job** is a task that ePublisher AutoMap schedules that generates output based on an existing ePublisher project or existing ePublisher Stationery. Once you create an ePublisher AutoMap job, you can run the job immediately, or schedule the job to run at a later point in time.

The fields are defined as follows:

### ePublisher project

Creates a new ePublisher AutoMap job based on an existing ePublisher project.

### ePublisher stationery

Creates a new ePublisher AutoMap job based on existing ePublisher Stationery.

### Choose ePublisher project or stationery

Specifies the location of the project or Stationery that ePublisher AutoMap should use to generate output.

## New Project Wizard

This wizard allows you to create a new ePublisher project in ePublisher Express or ePublisher Designer. A **project** is all the necessary files and components needed to generate output from source documents. This wizard provides several windows:

- "New Project Window (New Project Wizard)"
- "Browse For Folder Window (New Project Wizard)"
- "Source Documents Window (New Project Wizard)"

## New Project Window (New Project Wizard)

This window allows you to specify a name for a new project and where to save the new project you are creating. If you are using ePublisher Express, this window allows you to specify the Stationery to use when creating your new project. If you are using ePublisher Designer, this window allows you to specify the output format for your Stationery design project.

The fields are defined as follows:

### **Project name**

Specifies the name to use for your project.

### **Location**

Specifies the folder in which to save your new project. Click **Browse** to open the Browse For Folder window where you can select the folder in which to save your new project.

### **Format**

Specifies the output format to use for your project. You must specify at least one output format when you create the project. You can also specify additional output formats for the project after you create the project.

## **Browse For Folder Window (New Project Wizard)**

This window allows you to specify the folder where you want to save the new project. Navigate to and select the folder where you want to save the project, or click **Make New Folder** to create a new folder in which to save the project.

## **Source Documents Window (New Project Wizard)**

This window allows you to specify the source documents to use in your project. To add source documents to your project, click **Add**, select the files to add, and then click **Open**. To remove a source document from the list, select the source document and then click **Remove**. You can also add and remove source documents after you have created the project.

When the **Source Documents** field lists all the source documents you want to initially add to the project, click **Finish**.

## **Preferences Window**

This window allows you to customize ePublisher AutoMap behavior for your specific needs. These preferences affect the behavior of the application, such as the language displayed in the console.

This window provides the following tabs:

- “General Tab (Preferences Window)”
- “File Mappings Tab (Preferences Window)”
- “Notification Tab (Preferences Window)”

## General Tab (Preferences Window)

This window allows you to specify a job folder, staging folder, and user formats folder for ePublisher AutoMap jobs. You can also specify the user interface language ePublisher AutoMap uses, whether you want ePublisher AutoMap jobs to always scan for variables and conditions, and if you want ePublisher AutoMap jobs to delete temporary files created in the staging folder after generating output.

The fields are defined as follows:

### Job folder

Specifies the folder where ePublisher AutoMap stores job files and log files.

A job file is a file that uses a proprietary XML format. ePublisher AutoMap stores all information that describes an ePublisher AutoMap job in the job file. ePublisher AutoMap uses the name of the job as the job file name, and ePublisher AutoMap uses WebWorks AutoMap Job (.waj) as the extension for the job file. ePublisher AutoMap creates the job file based on information you specify using the ePublisher AutoMap user interface. Do not edit the job file directly. If you edit the job file outside of the ePublisher AutoMap user interface, ePublisher AutoMap may no longer be able to read the job file.

A log file is a file that contains information about events that occurred the last time the ePublisher AutoMap job generated output. The log file is a plain text file that you can view using any text editor. Log files use the .txt extension. ePublisher AutoMap does not create and store a log file in the job folder until the job runs and generates output.

By default, ePublisher AutoMap creates the job folder in the following location:

`\Documents and Settings\UserName\My Documents\WebWorks Automap`

`\Jobs`, where `UserName` is the name of the user account under which the job runs.

## **Staging folder**

Specifies the folder where ePublisher AutoMap stores the job information and ePublisher AutoMap job needs to generate output. The staging folder serves as a working folder for the job. Information such as automatically generated ePublisher projects, intermediate data files, and output files are stored in the staging folder.

By default, ePublisher AutoMap creates the staging folder in the following location: `\Documents and Settings\UserName\My Documents\WebWorks Automap\Staging`, where `UserName` is the name of the user account under which the job runs.

ePublisher AutoMap creates the staging folder contents based on information you specify using the ePublisher AutoMap user interface and uses this information to generate output. Do not edit files in the staging folder directly. If you edit files in the staging folder directly, the ePublisher AutoMap job may no longer be able to run.

## **Always scan for variables and conditions**

Specifies whether ePublisher AutoMap always scans for new variables and conditions when adding a document to the Document Manager.

## **Delete temporary files after generating**

Specifies whether ePublisher AutoMap jobs delete the temporary files the job created in the staging folder after generating output. This option is disabled in ePublisher AutoMap by default so you can examine the actual project and intermediate files created and stored in the staging folder when the ePublisher AutoMap job generated output. Enable this option to reduce the amount of computer disk space ePublisher AutoMap uses and if you do not want to examine the actual project and intermediate files created and stored in the staging folder when the ePublisher AutoMap job generated output.

## **User interface language**

Specifies the language the ePublisher AutoMap user interface uses. ePublisher is currently available in English, French, German and Japanese. If you change the user interface language to use, you must close and then reopen the user interface before the language change you specified takes effect.

# **File Mappings Tab (Preferences Window)**

This window allows you to specify file mappings. A file mapping is an association between a file extension and an ePublisher adapter. An ePublisher adapter is

an ePublisher component that bridges the gap between the application in which the source document was created and ePublisher. ePublisher currently provides adapters for Markdown++ (helper), Microsoft Word, Adobe FrameMaker, and XML.

The columns are defined as follows:

### **File Extension**

Specifies the name of the file extension.

### **Adapter**

Specifies the ePublisher adapter associated with the file extension.

## **Notification Tab (Preferences Window)**

This window allows you to specify email notification options if you want ePublisher AutoMap to send an email notification after an ePublisher AutoMap job completes. You can configure ePublisher AutoMap to send an email notification that contains information about if the job completed successfully or if the job generated errors. You can also specify to include the log file generated by the ePublisher AutoMap job as an email text file (`.txt`) attachment.

The fields are defined as follows:

### **Enable email notification**

Specifies whether ePublisher AutoMap sends out an email notification when a job completes.

### **To address**

Specifies the email addresses to which ePublisher AutoMap sends email notifications. Enter an email address for each person to send an ePublisher AutoMap email notification to when a job completes. To enter multiple email addresses, separate email addresses using a comma (,) character. You can also send email notifications to group email aliases. To create a group email alias to send ePublisher AutoMap email notification to, contact your system administrator at your company. After your system administrator creates the appropriate group email aliases you want to use, enter the group alias in the field.

### **From address**

Specifies the email address from which the email notification is sent. If someone replies to an ePublisher AutoMap email notification, this is the email address to which the reply will be sent.

### **SMTP server**

Specifies the name of the SMTP server ePublisher AutoMap uses to send email notifications. ePublisher AutoMap email notifications require an SMTP email server in order to send emails. Contact your system administrator at your company to obtain the name of an SMTP email server ePublisher AutoMap can use to send email notifications.

### **Username**

Specifies the name of a user account with permissions to send emails using the SMTP server. Contact your system administrator at your company to obtain the user name and password for a user with permissions to send ePublisher AutoMap notifications using the SMTP server.

### **Password**

Specifies the password for the user account with permissions to send email using the SMTP server. Contact your system administrator at your company to obtain the user name and password for a user with permissions to send ePublisher AutoMap email notifications using the SMTP server. ePublisher encrypts and stores the password you specify.

### **Always attach log to email**

Specifies whether you want to include the log file generated by the job as an email text file (`.txt`) attachment in the ePublisher AutoMap email notification.

## **Project Settings Window**

This window allows you to specify file mappings for your project. If you are using ePublisher Designer, this window also allows you to specify if you want to allow user style overrides in document previews for the project and input configurations. Depending on which ePublisher component you are using, this window can provide the following tabs:

- “File Mappings Tab (Project Settings Window)”
- “Input Configurations Tab (Project Settings Window)”

## **File Mappings Tab (Project Settings Window)**

This window allows you to specify file mappings for your project. A file mapping is an association between a file extension and an ePublisher adapter. An ePublisher adapter is an ePublisher component that bridges the gap between the application in which the source document was created and ePublisher. ePublisher currently provides adapters for Microsoft Word, Adobe FrameMaker, and XML.

The columns are defined as follows:

### **File Extension**

Specifies the name of the file extension.

### **Adapter**

Specifies the name of the ePublisher adapter associated with the file extension.

## **General Tab (Project Settings Window)**

If you are using ePublisher Designer, this window allows you to specify if the project allows user style overrides in document previews by default. A style override is a change made with Document Designer to the preview of a single document. Overrides can be applied to a paragraph or a table, and an override supersedes any values or properties set in either the source document or in Style Designer. Overrides appear only in output. They do not display in or affect the source document. Overrides cannot be saved to Stationery.

The fields are defined as follows:

### **Allow user style overrides in document previews (ePublisher Designer only)**

Specifies whether to allow user style overrides. When you add overrides to the output in your project, ePublisher adds bookmarks to your source documents. These bookmarks contain unique IDs that store override information and allow ePublisher Designer to keep track of your overrides. These IDs are invisible and do not affect the functionality of your source documents.

If you clear this option, you cannot add overrides to your output, and you cannot use Document Designer. If you do not want ePublisher to add these unique ID bookmarks to your source documents, clear the check box.

If all paragraphs in your Microsoft Word documents use a style other than Normal and you do not plan on adding any overrides to your documents, you may want to clear this option.

### **Compatibility Configuration**

The Base format version determines which Format library will be used when generating output. You can select any ePublisher version back to 9.2.2. This option is available to ensure ease of use when upgrading should user contain XSL overrides which may be incompatible with newer format libraries. Any plugin used in the project will be updated to match format version.



Users will see a log message such as the following when a project is configured to operate in compatibility mode:

```
[Warning] Project configured to use legacy format version '2009.2'.
```

Keep in mind that a project in running in compatibility mode can only use formats defined for that version. You will be able to use the latest format versions once you have upgraded your customizations to be compatible with the newer release.

## Input Configurations Tab (Project Settings Window)

This window allows you to specify settings for ePublisher input formats, also known as ePublisher adapters.

The fields are defined as follows:

### **Preserve condition colors in PDF**

Specifies whether you want to preserve any colors assigned to conditions in your Adobe FrameMaker source documents when you generate output.

### **Preserve condition styles in PDF**

Specifies whether you want to preserve any condition styles assigned to conditions in your Adobe FrameMaker source documents when you generate output.

### **Version used for generation**

Specifies the version of Adobe FrameMaker to use when generating output. Use this setting if you have more than one version of Adobe FrameMaker installed on the computer used to generate output and you want ePublisher to use a specific version of Adobe FrameMaker when generating output.

### **Preserve change bars in PDF**

This allows the change bar feature in Microsoft Word to be added to the PDF output.

### **Preserve Index**

By default, ePublisher does not use the Index generated within Microsoft Word. In cases (usually when generating PDFs) where you want to preserve the Index generated by Microsoft Word, you can enable this feature.

### **Preserve Table of Contents**

By default, ePublisher does not use the table of contents generated within Microsoft Word. In cases (usually when generating PDFs) where you want to preserve the table of contents generated by Microsoft Word, you can enable this feature.

### **Preserve Table of Figures**

By default, ePublisher does not use the table of figures generated within Microsoft Word. In cases (usually when generating PDFs) where you want to preserve the table of figures generated by Microsoft Word, you can enable this feature.

### **RD field behavior**

Specifies how you want Referenced Document (RD) field code behavior used when you generate output using Microsoft Word source documents that contain RD field codes that reference other source documents. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Indicates Book document</b>	Processes source documents that contain RD field codes as book files. ePublisher processes all the source documents identified by RD field codes.
<b>Ignore</b>	Ignores all the source documents identified by RD field codes. ePublisher processes only the source documents included directly in the project.
<b>Auto-Detect</b>	Processes source documents that contain RD field codes based on the contents of the source document. ePublisher analyzes the contents of each source document to determine if the file should be processed as a book file, front matter, table of contents, or index. This analysis is based on the presence of RD field codes combined with the presence or absence of a TOC or INDEX field code, which indicate whether the document contains a table of contents, index, or table of figures..

### **All DITA topics must exist**

Specifies whether or not to generate an error instead of a warning whenever a missing topic is found during the processing of a ditamap.

### **Combine DITA character styles**

Starting with version 2014.1, ePublisher will use combined character styles to allow nested elements to be styled more precisely in the generated output. For new projects this setting will default to `enabled`, otherwise it will be disabled for backward compatibility.

### **Convert notes to tables**

Enable this setting to convert DITA `<note>` elements into simple tables with ePublisher paragraph styles for better control of the generated output.

### **DITA Abbreviated Form Scope**

Specifies what scope to use when controlling when an abbreviation should be used instead of the long form. The values for this setting are defined as follows:

Value	Description
Map	Use the long form once for the entire ditamap. All other instances after the first usage will use the abbreviated form.
Part	If you organize multiple chapters into a part, then only the first instance will use the long form within the part.
Chapter	Use the long form once per chapter.
Topic	Use the long form once per topic. This is the default configuration.

### DITA Open Toolkit version

Specifies the version of the DITA Open Toolkit you want to use when generating output using DITA source documents.

### Emit draft comments

Enable this setting when working with DITA and you want to emit draft comments.

### Ignore print attribute

Disable this setting if you want ePublisher to process the print attribute for DITA content. In the DITA 1.3 specification, the `@print` attribute was deprecated and a new attribute that obeys DITA filtering rules, `@deliveryTarget`, was introduced.

## Save As Stationery Window

This window allows you to create Stationery from an ePublisher Designer Stationery design project by creating and saving a Stationery file. Stationery contains all project configurations, including style information, target settings, variables, conditions, cross-reference definitions, target overrides, user files, merge settings, and output format information.

The fields and columns are defined as follows:

### Name

Specifies the name of the Stationery file.

### **Directory**

Specifies the location where ePublisher saves the Stationery.

### **Target Name**

Specifies the names of the targets included in the Stationery.

### **Format Type**

Specifies the output format of the each target.

## **Script Editor Window**

This window allows you to write ePublisher AutoMap job scripts using a simple script editor. The ePublisher AutoMap script editor provides a text area where you can paste or write a script and a list of useful ePublisher AutoMap variables you can use in your scripts or pass to other scripts and applications.

ePublisher AutoMap only recognizes text-based scripts. If you paste a script into the script editor, any formatting or additional information available in a third-party script editor is lost when you paste the script into the ePublisher AutoMap script window.

The columns are defined as follows:

### **Name**

Specifies the name of the ePublisher AutoMap variable.

### **Description**

Specifies a description for the ePublisher AutoMap variable.

## **Target Configuration Window**

This window allows you to specify output generation parameters for each output target in ePublisher AutoMap. Depending on which output target you select in the **Target Name** column, this window can provide the following tabs:

- "Info Tab (Target Configuration Window)"
- "Conditions Tab (Target Configuration Window)"
- "Variables Tab (Target Configuration Window)"

- “Target Settings Tab (Target Configuration Window)”
- “Merge Settings Tab (Target Configuration Window)”

For example, if you select an output target that uses the Dynamic HTML output format, the Target Configuration window displays the **Info**, **Conditions**, **Variables**, and **Target Settings** tabs, but does not display the **Merge Settings** tab, because merge settings is not supported for the Dynamic HTML output format. However, if you select an output target that uses the Microsoft HTML Help output format, the Target Configuration window displays the **Info**, **Conditions**, **Variables**, **Target Settings**, and **Merge Settings** tab.

## Info Tab (Target Configuration Window)

This window allows you to specify output target settings for the selected output target, such as where to deploy the output after generating the output, if you want ePublisher AutoMap to delete any files in the output location before generating output, and if you want to run a pre-build or post-build script before or after generating output for each output target.

The fields are defined as follows:

### Deploy to

Specifies the deployment location. The deployment locations listed are deployment location configured in the ePublisher Stationery or ePublisher project associated with the ePublisher AutoMap job. Click **Add deploy target** to add an output deployment location to the list.

### Delete files in output location before deployment

Specifies whether ePublisher AutoMap deletes any existing files in the deployment location before ePublisher AutoMap places new files in the deployment location.

### Pre-build

Specifies the pre-build script to run before ePublisher AutoMap generates output. Type or paste the pre-build script into the text field or click **Edit Script** to use the Script Editor window to write your script.

### Post-build

Specifies the post-build script to run before ePublisher AutoMap generates output. Type or paste the pre-build script into the text field or click **Edit Script** to use the Script Editor window to write your script.

# Conditions Tab (Target Configuration Window)

This window allows you to specify the visibility of conditions in your output. The columns are defined as follows:

## **Name**

Specifies the name of the condition.

## **Value**

Specifies whether the content with the condition applied to it is included in your output. The values for this setting are defined as follows:

Value	Description
<b>Visible</b>	Displays the content in your output.
<b>Hidden</b>	Excludes the content from your output.
<b>Use document value</b>	Uses the state of the condition specified in your source documents to determine whether the content is displayed in your output.

### Passthrough

Places the content directly in your output without processing or transforming the output. This setting allows you to put HTML code in your source documents and have that code put directly in your output. The content with this type of condition applied is not transformed, so the special characters in HTML coding remain unchanged. For example, the `<` is not transformed to `&lt;` and the `>` is not transformed to `&gt;`.

## Variables Tab (Target Configuration Window)

This window allows you to specify the variables to use when generating output.

The columns are defined as follows:

### Name

Specifies the name of the variable.

### Use document value

Specifies whether you want to use the variable value specified in your source documents when generating output.

### Value

Specifies the value of the variable that will be used in your output. The default ePublisher behavior is to use variable value specified in the source document in your project. However, you can change the value of the variable in your project by clicking in the **Value** field for the variable to change and then typing in a new value for the variable.



## Target Settings Tab (Target Configuration Window)

This window allows you to specify the settings to use for the selected output target. The target settings displayed in ePublisher AutoMap are the target settings specified in ePublisher projects and Stationery. Typically, the target settings are configured and managed by the Stationery Designer in ePublisher Designer. However, you can override the target settings specified in ePublisher projects and Stationery using ePublisher AutoMap for each output target in an ePublisher AutoMap job if needed.

The columns are defined as follows:

### Setting

Specifies the settings available for the selected target.

### Value

Specifies the value of the setting.

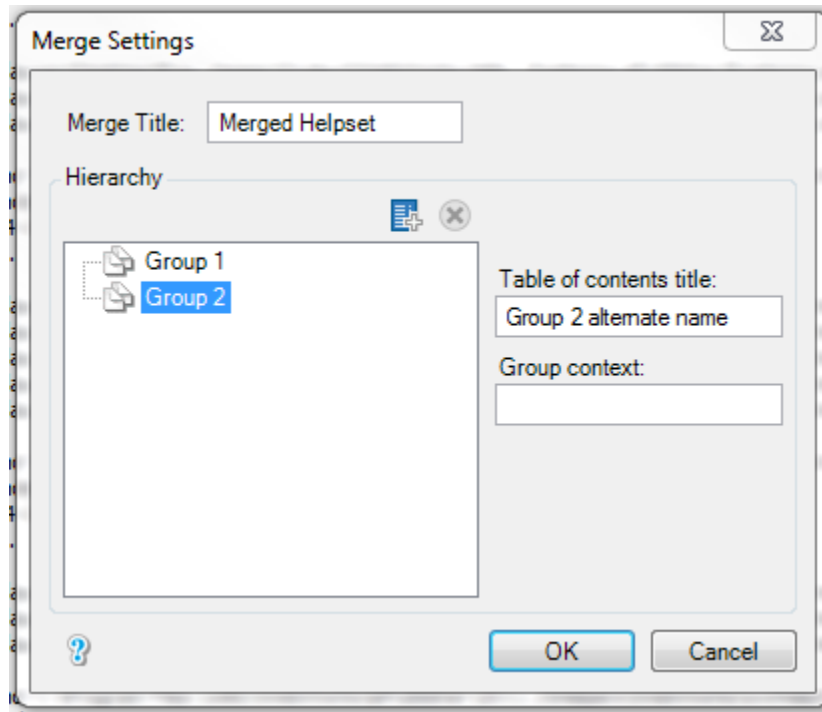
## Merge Settings Tab (Target Configuration Window)

This window allows you to specify custom settings for merged, or multivolume help. ePublisher only enables this window if you are working with an output format that supports merged, or multivolume help systems. Not all output formats support merged, or multivolume help. Eclipse Help, Microsoft HTML Help, and WebWorks Help are the output formats that currently support merged, or multivolume help systems. If you are working with an output format that does not support merged, or multivolume help, this tab is not displayed.

The fields are defined as follows:

### Merge Title

Specifies the name displayed in the title bar of the help system when the multivolume help system opens.



For example, if you have two or more groups, the Merge Title will reflect the title bar of the merged helpset and the top-level group.

## Hierarchy

Specifies all of the top-level groups in your project in a hierarchy. This group hierarchy is used to define the hierarchy displayed in the table of contents for your multivolume help system. To reposition any of your top-level groups in the table of contents hierarchy, select the group to move and drag the group to a new location in the hierarchy.

## Table of contents title

Specifies the name displayed for the top-level group when the top-level group is displayed in the table of contents for your merged, or multivolume help system. Change the name displayed for the top-level group in your merged, or multivolume help system by selecting a top-level group and then typing a new name in the field. By default, the name displayed is the name of the top-level group in your project.

## Group context

Specifies the name to use as the topic context when you generate merged, or multivolume help that includes context-sensitive help.

# Target Selection Window

This window allows you to specify the targets the ePublisher AutoMap job generates. A **target** is based on an output format, such as WebWorks Help, Microsoft HTML Help, or Sun JavaHelp, but targets also contain project-specific settings. Targets define how your output looks, and are based on output formats and settings specified in the ePublisher Stationery or ePublisher project associated with the ePublisher AutoMap job. An ePublisher AutoMap job can generate output for one or multiple targets.

The columns are defined as follows:

### **Target Name**

Specifies the list of targets in the ePublisher Stationery or ePublisher project associated with the ePublisher AutoMap job.

### **Format**

Specifies the type of output format associated with the target.

### **Build**

Specifies whether the target will be included in the ePublisher AutoMap job. Select this check box for the target to include the target in the ePublisher AutoMap job. Clear this check box if you do not want to include the target in the ePublisher AutoMap job.

## **User Information Window**

This window allows you to specify authentication information to schedule an ePublisher AutoMap job using Windows Task Scheduler. Windows Task Scheduler is a free tool that is included in Microsoft Windows operating systems.

With ePublisher AutoMap, you can specify when ePublisher AutoMap jobs run using Windows Task Scheduler. When you configure an ePublisher AutoMap job to run on a schedule, you must specify a user account and password for Windows Task Scheduler to use when running the job. When the ePublisher AutoMap job you scheduled using Windows Task Scheduler runs, Windows Task Scheduler executes the job using the security context of the credentials you specified when you created the job.

For more information about Windows Task Scheduler, refer to the Microsoft Windows operating system help.

The fields are defined as follows:

### **User name**

Specifies the user account name for Windows Task Scheduler to use when running the job. Enter your Windows user account name. Include the name of the domain to which the user account belongs to use the credentials of a user account that is a member of a Windows domain to run the job.

For example, type *DomainName\UserAccountName*, where *DomainName* is the name of the domain to which the user account belongs, and *UserAccountName* is the name of the Windows user account.

### **Password**

Specifies the password for the user account name. Type the password for the Windows user account you specified in the **User name** field.

## **Variables Window**

This window allows you to specify the values to use for variables when generating output. The columns are defined as follows:

### **Name**

Specifies the name of the variable.

### **Use document value**

Specifies whether you want to use the variable value specified in your source documents when generating output.

### **Value**

Specifies the value of the variable that will be used in your output. The default ePublisher behavior is to use variable value specified in the source document in your project. However, you can change the value of the variable in your project by clicking in the **Value** field for the variable to change and then typing in a new value for the variable.

## **WebWorks ePublisher Preferences Window**

This window allows you to customize ePublisher Designer and ePublisher Express behavior for your specific needs. These preferences affect the behavior of the application, such as the language displayed in the console.

This window provides the following tabs:

- “General Tab (WebWorks ePublisher Preferences Window)”

- “File Mappings Tab (WebWorks ePublisher Preferences Window)”

## General Tab (WebWorks ePublisher Preferences Window)

This window allows you to specify the number of projects listed on the Start Page, the user interface language ePublisher Express or ePublisher Designer uses, and source document scan options. You can also use this window to reset the toolbar, reset dock positions, and clear the **Do not ask again** option.

The fields are defined as follows:

### Scan added documents

Specifies when to scan source documents you add to a project. The scanning option you select will become the default selection for all existing projects, as well as all new projects you create. The default scan option is **Always**. When ePublisher scans source documents, it reads the style and formatting information, conditions, variables, and markers in your source documents. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Ask</b>	Specifies that ePublisher prompt you to scan source documents when you add them to a project.
<b>Always</b>	Specifies that ePublisher always scans source documents when you add them to a project. When you select this option, ePublisher automatically scans your source documents for style and formatting information when you add new source documents to your project.
<b>Never</b>	Specifies that ePublisher does not scan source documents when you add them to a project. If you select this option, ePublisher will never scan your source documents unless you manually scan your source documents using one of the scanning commands on the <b>Project</b> menu.

### **Preview Options (ePublisher Designer)**

Specifies when to create preview tabs of source documents added to the project or selected in the **Document Manager**.

#### **Automatically display preview for newly imported documents**

Checking this option causes ePublisher Designer to generate a new preview tab whenever a source document or file is added to the **Document Manager**.

#### **Display preview on document double-click**

Checking this option causes ePublisher Designer to generate a new preview tab whenever a source document or file in the **Document Manager** is double-clicked.

### **Recent projects to remember**

Specifies the number of projects to list on the Start Page. Click **Clear List** to clear the list of projects displayed on the Start Page.

### **Language**

Specifies the language the ePublisher Express or ePublisher Designer user interface uses from the drop-down list. If you change the user interface

language ePublisher Express or ePublisher Designer uses, you must close and then reopen the user interface before the language change you specified takes effect. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>English</b>	Specifies English as the user interface language.
<b>Deutsch</b>	Specifies German as the user interface language.
<b>Francais</b>	Specifies French as the user interface language.

## **Miscellaneous**

### **Reset Toolbar**

Reset toolbar interface to default position.

### **Reset Dock Positions**

Reset user interface dock positions to default.

### **Clear “Do not ask again”**

Reset automatic prompt settings to their default.

### **Reset Evaluation Materials**

Reset the trial projects and sample materials to their original state.

## **File Mappings Tab (WebWorks ePublisher Preferences Window)**

This window allows you to specify file mappings. A file mapping is an association between a file extension and an ePublisher adapter. An ePublisher adapter is an ePublisher component that bridges the gap between the application in which the source document was created and ePublisher. ePublisher currently provides adapters for Microsoft Word, Adobe FrameMaker, and XML.

The columns are defined as follows:

### **File Extension**

Specifies the name of the file extension.

### **Adapter**



Specifies the name of the ePublisher adapter associated with the file extension.

## Editor Preferences Tab (WebWorks ePublisher Preferences Window)

This window allows you to specify both a Text Editor as well as an Image Editor for use when modifying overridden files. The overridden files can be viewed and easily accessed from the **Advanced** menu.

## Diff Preferences Tab (WebWorks ePublisher Preferences Window)

This window allows you to specify the path to your preferred **Diff Program**. A `Diff` program allows you to manage your custom changes to overridden files. Overridden files can be easily accessed from the **Advanced** menu.

## Log Window Tab (WebWorks ePublisher Preferences Window)

This window allows you to edit the Log Window display settings. You can change the foreground, background, warning message, and error message colors. You can also set the font, and enable word wrap.

## WebWorks Licensing Info Window

This window allows you to specify your Contract ID. A **Contract ID** is a unique identifier that identifies the number of users and type of functionality enabled for your ePublisher installation. WebWorks generates an appropriate Contract ID for your ePublisher installation when you purchase ePublisher or request an evaluation copy of ePublisher.

You must enter your Contract ID, email address, and computer name before you can use ePublisher components. The Contract ID enables the ePublisher product components and ePublisher input formats for which you are licensed.

The fields are defined as follows:

### Contract

Specifies your Contract ID. Enter the Contract ID you received from WebWorks when you purchased ePublisher or requested an evaluation copy of ePublisher.

## **Email**

Specifies your email address. If you have an email address that you use as your WebWorks support login, enter that email address.

## **Computer**

Specifies the name of the computer where ePublisher is installed.

“Working with Contract IDs”

“Viewing Licensing and Contract ID Information”

“Obtaining Contract IDs”

“Entering Contract IDs”

“Managing Licensing in Environments without Internet Connectivity”

“Updating Licensing”

“Deactivating Licensing”

“Problems with FrameMaker or Microsoft Word”

# Producing Output from Stationery

- What Makes an ePublisher Project
- Source Documents
- Targets
- Stationery
- Creating Projects Based on Stationery
- Working with Source Documents
- Working with Targets
- Working with Projects
- Generating and Regenerating Output
- Viewing Output
- Validating Output Using Reports
- Merging Top-level Groups (Multivolume Help)
- Deploying Output
- Working with Target Settings
- Setting Variables in Projects
- Setting Conditions in Projects
- Setting Cross-References in Projects
- File Mappings for Source Documents

This section explains how writers can use their source documents and ePublisher projects and Stationery to produce output.

## What Makes an ePublisher Project

This section explains what a project is and the folder structure projects use.

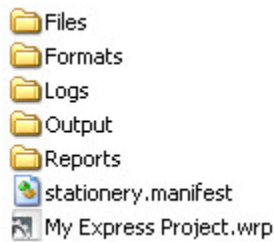
### ePublisher Projects

An ePublisher project consists of all the necessary pieces needed to convert your source documents into online output. It contains your source documents, images, project Stationery, and any settings or preferences you specify. After you create a project, you can modify your project settings and preferences, generate reports, and produce output.

ePublisher provides powerful single-sourcing capabilities that allow you to generate online output in multiple formats using a single project and a single set of source documents. For example, you can generate WebWorks Help, Microsoft HTML Help, Oracle Help, Dynamic HTML Help, and Sun JavaHelp using one project and one set of source files.

### Project Folder Structure

The following figure shows a sample project folder.



To view all of the files for your project, on the **View** menu, click **Project Directory**.

The project folder contains the following subfolders:

## Files

Contains any custom files you want your project to use. Typically, the `Files` folder contains logo images, custom `.css` files, custom bullet images, and custom background images. To view the files in the `Files` folder for your project, on the **View** menu, click **User Files**.

You can also place a `Files` directory in a target's override folder or within the target's format override folder.

- `Formats\<format name>\Files`
- `Targets\<target name>\Files`

These files will be copied for each target's sub-directory but will not be visible in the user interface to select, e.g. the company logo image in the Target Settings dialog.

## Formats

Contains output format overrides and all of the files required to generate output for an output format. Any time you want to override an output format file, place the override in the `Formats` folder. For example, if you want to override the standard table of contents icons for topics in WebWorks Help by specifying custom table of contents icons, place the custom table of contents icons you want to use in an `images` folder in the `Formats` folder. For more information about format overrides, see "Creating Format Overrides". To view the files in the `Formats` folder for your project, on the **View** menu, click **Format Override Directory**.

## Logs

Contains the `generate.log` file. The `generate.log` file contains information about the actions ePublisher performed when generating output, along with any warning or errors that occurred during output generation.

## Output

Contains the files ePublisher creates when generating output and provides a structure for generated output files. ePublisher creates a folder for each target in the project in the `Output` folder, and each target folder contains all topic pages, generated images, entry-point files, and merged help files generated for the target. The **entry-point file** is the file that opens the help system.

By default, ePublisher creates the `Output` folder in the following location:

- **If you use ePublisher Express**, by default ePublisher creates the `Output` folder in the `My Documents\ ePublisher Express Projects \ProjectName` folder, where `ProjectName` is the name of the project.
- **If you use ePublisher Designer**, by default ePublisher creates the `Output` folder in the `My Documents\ ePublisher Designer Projects \ProjectName` folder, where `ProjectName` is the name of the project.

You can view output files in the `Output` folder using Output Explorer or by clicking on and opening output files in the `Output` folder. To view the files in the `Output` folder for your project, on the **View** menu, click **Output Directory**. For more information about viewing output, see “Viewing Output in Output Explorer” and “Viewing Output in the Output Folder”.

## Reports

Contains all of your configured reports in an XML format that ePublisher can display.

## Targets

Contains target overrides. Any time you want to override a file in a target, place the override in the `Targets` folder. ePublisher looks for overrides in the `Targets` folder before it looks for overrides in the `Formats` folder. The `Targets` folder only appears in your project folder if your Stationery is configured to use target overrides. For more information about target overrides, see “Creating Target Overrides”. To view the files in the `Targets` folder for the selected target in your project, on the **View** menu, click **Target Override Directory**.

Your project also uses a `Data` folder. The `Data` folder contains information about how files in your project have been processed. The `.wif` files, which are located in the `Data` folder, contain style and content information from your source documents. ePublisher creates the Data folder in the following temporary folder location:

```
Documents and Settings\UserName\Local Settings\Temp\WebWorks
\ePublisherComponent\Data
```

where *UserName* is the name of the ePublisher user, and *ePublisherComponent* is the name of the ePublisher component used to generate output, such as ePublisher Express or ePublisher Designer.

To view the files in the `Data` folder for your project, on the **View** menu, click **Data Directory**.

## Source Documents

Source documents are documents you create your content in using a content authoring tool such as Notepad++, Microsoft Word, Adobe FrameMaker, or an authoring environment that supports DITA authoring. After you prepare your source documents, you use your source documents to generate output.

## Targets

A **target** is the specific type of output you want to produce using your source files and project settings. Targets are based the output formats you specify for your project, and include all of the project settings you specify for each output format included in your project when you configure your project.

For example, assume that you are a writer working at CompanyA, and you have the requirement to create a web-based help system using your source documents. In this scenario, you create a project using stationery that supports WebWorks Help output and then create a target called CompanyA WebWorks Help.

Next, assume that your documentation requirements change, and in addition to creating WebWorks Help for CompanyA, you must now also produce Microsoft HTML Help and PDF files for CompanyA using your same source documents. In this scenario, you update your project to now include the following targets:

- CompanyA WebWorks Help
- CompanyA Microsoft HTML Help
- CompanyA PDF Files

Finally, assume your documentation requirements change again, and now, based on an Original Equipment Manufacturer (OEM) agreement your company signed, in addition to creating WebWorks Help, Microsoft HTML Help, and PDF files for CompanyA, you must use your same set of source documents to create WebWorks Help, Microsoft HTML Help, and PDF files for CompanyB, using company information and variables and conditions specific to CompanyB. In this scenario, you update your project to now include the following targets:

- CompanyA WebWorks Help
- CompanyA Microsoft HTML Help
- CompanyA PDF Files

- CompanyB WebWorks Help
- CompanyB Microsoft HTML Help
- CompanyB PDF Files

When you have multiple targets included in a project, you choose an active target and then specify project settings for the active target. The active target is the target currently selected in your project. When you want to modify project settings for an target, if you have multiple targets included in your project, ensure you have the correct target selected in the project when you modify project settings. For more information about specifying an active target, see “Specifying Active Targets”.

## Stationery

The Stationery designer creates Stationery with ePublisher Designer using a Stationery design project. Stationery specifies the settings ePublisher uses to generate output. Stationery designers create Stationery by creating a Stationery design project in ePublisher Designer and then saving the processing rules, styles, and other information specified in the Stationery design project as Stationery. ePublisher Express and ePublisher AutoMap can then use the Stationery to generate output.

When the Stationery designer creates a project in ePublisher Designer and then saves a project using the **Save As Stationery** option, ePublisher Designer creates a Stationery file. A Stationery file is a file with the `.wxsp` file extension that contains formatting, project settings, project overrides, and style information. Source documents and document-specific information, such as Document Manager groups, are not saved in Stationery. After the Stationery designer creates the Stationery, writers use the Stationery provided by the Stationery designer when they create an ePublisher Express project. Writers use their ePublisher Express project and Stationery to generate output.

When the Stationery designer saves the Stationery, ePublisher creates the following folders:

- `StationeryName\Formats\OutputFormat`
- `StationeryName\Formats\OutputFormat.base`

where *StationeryName* is the name the Stationery designer specified for the Stationery, and *OutputFormat* is the type of output format the Stationery Designer specified for a target in the Stationery.

The `StationeryName\Formats\OutputFormat` folder contains any customizations or overrides the Stationery designer specified when designing the Stationery. ePublisher Express synchronizes with the files in the *OutputFormat* folder and

uses the information about customizations and overrides contained in files in the *OutputFormat* folder to generate output.

**Note:** The Stationery may have one or more *OutputFormat* folders, based on the settings the Stationery designer specified.

The *StationeryName*\Formats\OutputFormat.base folder contains copies of all the files located in the \Program Files\WebWorks\ePublisher\2024.1\Fformats\OutputFormat folder. These files define the default output format and transforms and are installed by default when you install ePublisher.

Stationery designers can do a compare, or **diff**, between the files located in these folders to quickly see any customizations or overrides specified for the Stationery. Stationery designers can use this information to help them reapply customizations and overrides as needed when designing a newer version of the Stationery in ePublisher Designer.

When the styles or features used in the generated output need to change, the Stationery designer uses the ePublisher Designer Stationery design project to update the styles and features specified in the Stationery, and then the Stationery designer saves the changes, creates updated Stationery, and deploys the Stationery. Once the new Stationery is available, writers synchronize their ePublisher Express project with the updated Stationery file and use the updated Stationery the next time they generate output.

**Note:** When you synchronize your project with Stationery, the synchronization process overwrites any target setting customizations you configured for the project.

For more information about synchronizing Stationery, see “Synchronizing Projects with Stationery”.

## Creating Projects Based on Stationery

Writers use ePublisher Express and Stationery created by a Stationery designer to generate output. When you use ePublisher Express to create a project based on Stationery, you specify the Stationery you want the project to use and the source documents you want to include in the project. The Stationery file uses a .wxsp file extension and contains information and settings for the project to use, such as style or format information, variable values, condition settings, cross-reference definitions, and more. The source documents contain the content for which you want to generate output. The project uses the settings specified in the Stationery file and the content and formatting in the source documents to generate output. A project file created with ePublisher Express uses the .wrp file extension.

**Note:** You cannot create a project based on Stationery using ePublisher Designer. You can only create projects based on Stationery using ePublisher Express. Stationery designers use ePublisher Designer to create Stationery using Stationery design projects.



## To create a project based on Stationery

1. In ePublisher Express, on the **File** menu, click **New Project**.
2. In the **Project Name** field, type a name for your project.
3. In the **Location** field, specify the location where you want to save your ePublisher project by clicking on the folder icon and browsing to the location where you want to save your project.

**Note:** Ensure you consider the length of the full path you specify for the project name and location. If you specify long names and paths for project, Windows may not be able to support the length of the full path.

By default, ePublisher stores projects in the `My Documents\ePublisher Express Projects` folder.

4. In the **Standalone stationery** field, specify the Stationery you want to use to create your project by clicking on the folder icon and browsing to the location of the Stationery file.
5. Select a Stationery file (`.wxsp` file), and then click **Open**.
6. Click **Next**.
7. Click **Add**.
8. Browse to the location of the source documents you want to include in your project, select the source documents, and then click **Open**.

**Note:** You can add source documents when you create your project or you can add source documents after you create your project. For more information about adding source documents to projects, see “Adding Source Documents to Projects”.

9. Click **Finish** to create the project. ePublisher creates the project and gathers information about the structure of your source documents.

After you create your project, add targets to your projects as needed and then generate output. For more information, see “Adding Targets to Projects Based on Stationery” and “Generating Output”.

## Working with Source Documents

This section explains how to work with source documents in Document Manger.

### Adding Source Documents to Projects

You can add source documents to your project when you create a project. You can also add source documents to your project after you create a project. When you add source documents to your project, ePublisher automatically adds the source documents to your project and creates a top-level group in Document Manager that contains your source document. For more information about top-level groups, see “Source Documents Groups”.

## To add a source document to your project

1. On the **Project** menu, click **Add Document**.
2. Browse to the folder that contains the source document you want to add to your project.
3. Select the source document you want to add to your project, and then click **Open**.
4. ***If you configured ePublisher to scan source documents when you add source documents to projects***, ePublisher adds the source documents to Document Manager and scans the source documents. For more information about scanning source documents and setting scanning options, see "Scanning Source Documents" and "Setting Scanning Options".
5. ***If you did not configure ePublisher to scan source documents when you add source documents to projects***, ePublisher adds the source documents to Document Manager but does not scan your documents. After ePublisher adds your source documents to Document Manager, scan your source documents. For more information about scanning source documents and setting scanning options, see "Scanning Source Documents" and "Setting Scanning Options".
6. ***If you are adding a FrameMaker book file to your project***, ePublisher adds the FrameMaker book file (`.bk` or `.book` files) and the source documents the FrameMaker book file contains (`.fm` files) to your project. Consider the following points when you add a FrameMaker book file to your project:
  - When you add a FrameMaker book to your project, by default ePublisher creates a group for the FrameMaker book in Document Manager, and any FrameMaker source documents contained within the FrameMaker book are always contained within the group in your project.
  - When you make changes to a FrameMaker book, such as adding or removing source documents from a FrameMaker book file, when you scan the FrameMaker book, ePublisher updates the project with the changes you made to the FrameMaker book file. If you add or remove FrameMaker source documents in a FrameMaker book, ensure you scan the FrameMaker book before you generate output. For more information about scanning source documents and setting scanning options, see "Scanning Source Documents" and "Setting Scanning Options".
  - ***If your FrameMaker book contains front matter files, table of contents files, or index files***, consider the following points:
    - ***If you are generating output for a target that uses any output format other than PDF***, by default ePublisher generates

output for source document front matter files included in a book, but does not generate output using the table of contents files and index files included in the FrameMaker book. ePublisher instead uses the headings and index entries in the source documents to generate a table of contents and an index for your online output.

- ***If you are generating output for a target that uses PDF as the output format***, by default ePublisher generates the PDF using the front matter, index, and table of contents files included in the FrameMaker book.
- The Stationery designer may modify these default file processing settings when designing Stationery. If you have target setting modification permissions, you can also customize these settings as needed. For more information about target setting customization permissions and customizing file processing settings, see “Working with Target Settings” and “Specifying File Processing Behavior for Front Matter, Index, and Table of Contents Files”. If you do not have target setting customization permissions, instead of adding an Adobe FrameMaker `.book` file that contains front matter, table of contents, and index files, you can instead add the individual Adobe FrameMaker chapter `.fm` files, and then use the individual chapter files to generate output.

After you add source documents to your ePublisher project, ePublisher displays your source documents in Document Manager. You can organize your source documents in Document Manager and perform the following tasks:

- Open and edit source documents from within Document Manager. For more information, see “Opening Source Documents from Document Manager”.
- Relink source documents. For more information, see “Relinking Source Documents”.
- Remove source documents from your project. For more information, see “Removing Source Documents from Projects”.
- Create an organizational structure for your online output using groups. For more information, see “Source Documents Groups” and “Organizing Source Documents Using Groups”.
- Rearrange the source document order in Document Manager. For more information, see “Rearranging Source Documents in Groups”.

## Opening Source Documents from Document Manager

If you want to edit the content of your source documents while working with a project, you can open the source documents from Document Manager.

## To open a source document from Document Manager

1. In Document Manager, double-click the source document you want to open. ePublisher opens the source document using the content authoring tool you used to create the source document.
2. ***If you want to edit the content in your source document***, edit the content using the content authoring tool you used to create the source document.
3. Save the source document.

## Scanning Source Documents

This section explains how scanning works, how to scan source documents, and source document scanning options.

### Scanning and Scanning Options

When ePublisher scans your source documents, it reads the style and formatting information, variables, conditions, and marker types in your source documents and then imports this information into your ePublisher project. Once ePublisher imports this information into your project, you can generate output. You can also modify target settings if you have permissions to modify target settings. For more information, see “Generating and Regenerating Output” and “Working with Target Settings”.

The scanning process can be time-consuming. You can reduce the amount of time it takes ePublisher to scan your documents by scanning only the source documents you select. Scan your source documents when you have made any of the following changes to your source documents:

- Added new content
- Added new style information
- Modified any existing styles
- Added new markers, variables, or conditions
- Modified existing markers, variables, or conditions

ePublisher provides the following options for scanning source documents in Document Manager:

#### Scan Selected

Scans only the selected source document in Document Manager.

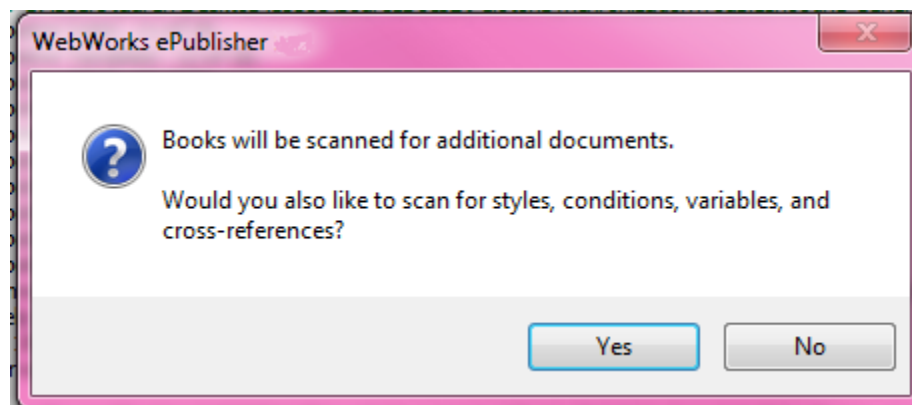
## Scan All Documents

Scans all of the source documents that you added to your project and that are displayed in Document Manager.

## Setting Scanning Options

By default, ePublisher will automatically scan your documents, thus making sure any new styles, conditions, variables, etc. are added to your project.

If you set this option to **Ask**, then adding a source document a dialog box will appear:



This indicates that files such as FrameMaker .book files will scan for additional .fm files linked from this source. Clicking Yes will scan the individual documents so that the document set styles, for example can be added to the Style Designer (the same goes for most of the document set customizations.)

However, you can specify that you want ePublisher to scan source documents when you add them to your project. For example, you can choose to have ePublisher prompt you to scan the source documents when you add source documents to your project, or you can choose to always have ePublisher scan source documents when you add them to a project. The scanning option you specify will become the default selection for all existing and subsequent projects.

If you choose to never have ePublisher scan source documents, when you add them to a project, you must remember to scan your source documents before you generate output.

### To set scanning options

1. On the **Edit** menu, click **Preferences**.
2. On the **General** tab, in the **Scan options** area, select the scan setting you want to specify. For more information about scanning options, click **Help**.
3. Click **OK**.

## Scanning Selected Documents

Sometimes you may make a change to content in a single source document. You can scan only the source document you changed. Scanning the selected document updates your project with the new information you specified in the selected source document.



### **To scan a selected source document**

1. In Document Manager, select the source document you want to scan.
2. On the **Project** menu, click **Scan Selected**. ePublisher scans the document you selected in Document Manager.

## **Scanning All Documents**

If you have made multiple changes to content in your source documents, you can scan all of the source documents included in your project at once. Scanning all source documents ensures that ePublisher includes any changes you made to any of the source documents in your project.

### **To scan all source documents in a project**

On the **Project** menu, click **Scan All Documents**. ePublisher scans all of the source documents displayed in Document Manager.

## **Relinking Source Documents**

Sometimes the link between Document Manager and the source document may become broken. For example, moving the source document to another folder location or deleting the source document from a folder may break the link between Document Manager and the source document. When ePublisher detects a broken link between Document Manager and the source document, ePublisher displays a **Broken Link** icon, or red question mark, next to the source document in Document Manager.

## To relink a source document

1. In Document Manager, double-click the **Broken Link** icon next to the name of the source document.
2. Browse to the location of the source document.
3. Select the source document, and then click **Open**. ePublisher recreates the link between the source document and Document Manager.

## Removing Source Documents from Projects

You can remove source documents from an ePublisher project. Remove source documents from your project when you no longer want to include the content in the source document in your project or in your generated output.

***If you are a Stationery designer using ePublisher Designer to design Stationery***, when you remove source documents from an ePublisher project, any styles or formats associated with the source document remain in Style Designer. For example, assume that the *UserManualTitle* style is a style that is specific to only one source document in your project. If you remove the source document that contains the *UserManualTitle* style from your project, ePublisher retains the *UserManualTitle* style name and style information in Style Designer. If you want to remove this style from Style Designer, you must manually delete it.

## To remove a source document from a project

1. In Document Manager, click the source document you want to remove from your project.
2. On the **Edit** menu, click **Remove**.
3. ***If you want to remove an Adobe FrameMaker source document (.fm file) that is a part of an Adobe FrameMaker book (.book or .bk file) you have added to a project***, you cannot remove the Adobe FrameMaker source document from the project using ePublisher. You must remove the Adobe FrameMaker source document from the Adobe FrameMaker book file and then scan the Adobe FrameMaker book file to remove the Adobe FrameMaker source document from your project. For more information about scanning source documents, see "Scanning Source Documents".
4. Click **Yes** to confirm that you want to remove the source document from your project.

## Source Documents Groups

Groups are containers in Document Manager that hold your source documents and allow you to create an organizational structure for your output. When you first create a new project, ePublisher automatically creates a new group in Document Manager using the project name. You can use ePublisher to create the following types of groups in Document Manager:

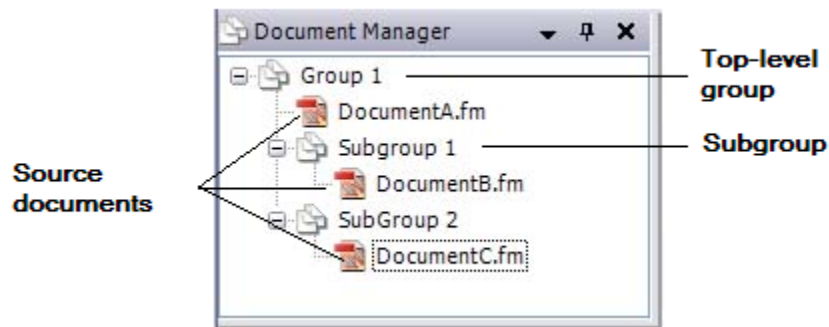
### Top-level groups

Contains source documents and subgroups. ePublisher creates an entry-point file for each top-level group in Document Manager. The entry-point file is the file that opens the generated output. All projects must contain at least one top-level group. You can create additional top-level groups to further organize your source documents in Document Manager or if you want to create merged, or multivolume, help. For more information about merged help systems, see "Merging Top-level Groups (Multivolume Help)".

### Subgroups

Used to organize source documents within top-level groups. Subgroups do not create entry-point files and do not represent an actual volume in a merged help system.

The following figure shows top-level groups and subgroups in Document Manager.



## Organizing Source Documents Using Groups

You can perform the following actions with source documents and groups in ePublisher:

- Create top-level groups. For more information, see "Creating Top-Level Groups".
- Create subgroups. For more information, see "Creating Subgroups".
- Rename groups. For more information, see "Renaming Groups".
- Rearrange source documents in groups. For more information, see "Rearranging Source Documents in Groups".
- Remove groups. For more information, see "Removing Groups".

### Creating Top-Level Groups

By default, ePublisher creates a top-level group based on the name of the project when you add your first source document to your project. There must always be at least one top-level group in Document Manager in order to add source documents to a project. You can create additional top-level groups if you want to further organize your source documents or create merged help systems, or multivolume help. For more information about creating merged help systems, see "Merging Top-level Groups (Multivolume Help)".

### To create a top-level group

1. On the **Project** menu, click **New Group**. ePublisher creates and displays a new top-level group in Document Manager.
2. Type a name for the new group.
3. Drag the new top-level group to its appropriate position above, below, or between an existing top-level group in Document Manager.

## Creating Subgroups

You can create subgroups in Document Manager to organize the source documents in a group. By organizing your source documents into subgroups, you can organize how you want to display your source documents in Document Manager and how you want content to display in your generated output.

### To create a subgroup

1. In Document Manager, select the group to which you want to add a subgroup. You can add a subgroup to a top-level group or to an existing subgroup.
2. On the **Project** menu, click **New Group**. ePublisher displays the new group in Document Manager.
3. Type a name for the new group.

## Renaming Groups

You can rename existing top-level groups and subgroups in Document Manager. For example, when you create a new project, by default ePublisher creates a new group based on the project name. However, you can change the default name of the group in Document Manager.

## To rename a group in Document Manager

1. In Document Manager, click twice on the group you want to rename.
2. Type a new name for the group.
3. Press ENTER or click outside of the typing area to change the name.

## Rearranging Source Documents in Groups

Once you have added source documents to your project and placed your source documents into groups within Document Manager, you can rearrange source documents by moving the source documents within the same group or by moving source documents to a new location in a new group.

If you have a FrameMaker book (`.bk` or `.book` file) in a group, you can move the FrameMaker book to a different group, but you cannot move an individual FrameMaker document (`.fm` file) to a group if it is included in the `.book` file. `.fm` files that belong to a `.book` file must remain in the same group as the `.book` file. If you want to move a `.fm` file to a different group than the `.book` file is in, first remove the `.fm` file from the book, scan the book, and add the `.fm` file, which is no longer part of the book, to the appropriate book.



## To rearrange source documents in groups

1. ***If you want to change the order of source documents within a group,*** complete the following steps:
  - a. In Document Manager, click the source document you want to move.
  - b. Drag the source document to the desired location within the group.
2. ***If you want to move a source document to a different group,*** complete the following steps:
  - a. In Document Manager, click the document you want to move.
  - b. Drag the source document to the desired location within the new group.

## Removing Groups

If you no longer want to use a group, you can remove the group from Document Manager. When you remove a group from Document Manager, ePublisher removes any source documents associated with the group from your project.

**Note:** ePublisher does not delete the source documents from your computer. ePublisher only removes the source documents from the project.

### To remove a group

1. In Document Manager, select the group you want to remove.
2. On the **Edit** menu, click **Remove**.

## Working with Targets

This section explains how to work with targets. For more information about what targets are, see “Targets”.

### Specifying Active Targets

Within a project, you can have multiple targets. The active target is the target currently selected in the project. ePublisher uses the active target when you make modifications to your target settings or generate output.

### To specify the active target

On the **Project** menu, select the target next to **Active Target**.

## Adding Targets to Projects Based on Stationery

Every project must contain at least one target. Add targets to projects when you need to produce different kinds of output using the same source documents. Each target is associated with one output format, such as WebWorks Help, Microsoft HTML Help, or PDF. If you are generating output based on Stationery using ePublisher Express, the Stationery you use for your project defines the type of output formats you can specify for a target when you add a target to your project. You can only use output formats defined in the Stationery by the Stationery designer when you create targets. If you need to create a target for an output format not included in the Stationery, talk to the Stationery designer about updating the Stationery to include the output format.

For example, assume that you are a writer working at CompanyA, and you need to create web-based help. You have Stationery from a Stationery designer configured to support WebWorks Help, Microsoft HTML Help, and PDF output. In this scenario, you create an ePublisher project based on Stationery from the Stationery designer, and then you create a target called CompanyA WebWorks Help that specifies WebWorks Help as the output format for the target.

Next, assume that your documentation requirements change, and in addition to creating WebWorks Help for CompanyA, you must now also produce Microsoft HTML Help and PDF files for CompanyA using your same source documents. In this scenario, you update your project by adding Microsoft HTML Help and PDF as targets, and your project now contains the following targets:

- CompanyA WebWorks Help
- CompanyA Microsoft HTML Help
- CompanyA PDF Files

## To add a target to a project based on Stationery

1. On the **Project** menu, click **Manage Targets**.
2. Click **Add**.
3. In the **Format Type** field, select the output format you want to use for the format target.
4. In the **Target Name** field, type a name for the format target.
5. Click **OK**.

## Renaming Targets

You can rename targets. By default, the target name is the same as the output format in ePublisher. However, in some situations, you may want specify a different name for the target. For example, assume that you are a writer working at CompanyA, and you have the requirement to create a web-based help system using your documentation source files. In this scenario, you create an ePublisher project that specifies WebWorks help as your help system and you configure your project settings to use information and branding for CompanyA to create an target called *WebWorks Help*.

Next, assume that your requirements change, and now, based on an OEM agreement your company signed, in addition to creating WebWorks Help for CompanyA, you must use your source files to create WebWorks Help for CompanyB. In this scenario you create a new target in your project called *CompanyB WebWorks Help* and configure settings for this target. However, after configuring settings for the CompanyB WebWorks Help target, you now want to go back and rename your original WebWorks Help output format, and change the name of this output format to *CompanyA WebWorks Help*.

### To rename a target

1. On the **Project** menu, click **Manage Targets**.
2. In the **Target Name** field, click the name of the output format you want to rename.
3. Click **Edit**.
4. In the **Target Name** field, type the new name you want to specify.
5. Click **OK**.

## Deleting Targets

You can delete targets from a project if you no longer need to produce output for the target.

### To delete a target

1. On the **Project** menu, click **Manage Targets**.
2. In the **Target Name** field, click the name of the output format you want to delete.
3. Click **Delete**.
4. Click **OK**.

## Working with Projects

This section explains how to work with projects.

### Saving Projects

You should periodically save your project to ensure that you do not lose any changes you have made. By saving your project, you ensure that ePublisher stores the information in your project in the project files and all of your project information will be available the next time you open your project.

## To save a project

On the **File** menu, click **Save**. ePublisher automatically saves your ePublisher project in a file in the location you specified when you first created the project.

- **If you are saving an ePublisher Express project**, by default ePublisher saves the project file in the `My Documents\ePublisher Express Projects\ProjectName` folder, where `ProjectName` is the name of the project.
- **If you are saving an ePublisher Designer project**, by default ePublisher saves the project file in the `My Documents\ePublisher Designer Projects\ProjectName` folder, where `ProjectName` is the name of the project.

## Opening Existing Projects

You can open an existing project using one of the following methods:

- Open the project from within the ePublisher Express or ePublisher Designer user interface.
- Open the project from Windows Explorer by double-clicking the project file in the folder where you saved the project.

By default ePublisher saves project files in the following locations:

- ePublisher saves ePublisher Express project files in the `My Documents\ePublisher Express Projects\ProjectName` folder, where `ProjectName` is the name of the project. ePublisher Express project files use the `.wrp` file extension.
- ePublisher saves ePublisher Designer project files in the `My Documents\ePublisher Designer Projects\ProjectName` folder, where `ProjectName` is the name of the project. ePublisher Designer project files use the `.wep` file extension.

When you open an existing project, ePublisher opens a separate instance of the ePublisher for each project, and each project has its own window. For example, if you have *ProjectA* open, and then you decide to open a project called *ProjectB*, ePublisher opens up a new instance of ePublisher for the new project and you have two ePublisher instances with *ProjectA* and *ProjectB* open concurrently on your computer.

## To open an existing project

1. ***If you want to open an existing project from within ePublisher Express or ePublisher Designer***, complete the following actions:
  - a. In ePublisher, on the **File** menu, click **Open**.
  - b. Browse to the location of the project file you want to open.
  - c. Select the project file you want to open, and then click **Open**.
2. ***If you want to open an existing project using Windows Explorer***, complete the following steps:
  - a. In Windows Explorer, browse to the location of the ePublisher project file you want to open.
  - b. Double-click the ePublisher project file.

## Closing Projects

When you finish working with a project, you can close it. When you close the project, ePublisher prompts you to save any changes to your project that you have not already saved.



## To close a project

On the **File** menu, click **Exit**.

# Synchronizing Projects with Stationery

ePublisher Express projects use Stationery designed in ePublisher Designer by the Stationery designer. From time to time, the Stationery designer may update the Stationery used by your ePublisher Express project. When the Stationery designer updates the Stationery, you must synchronize your ePublisher Express project with the Stationery associated with the project in order to obtain the updates made by the Stationery designer. For more information about Stationery, see "Stationery".

When the Stationery designer updates the Stationery an ePublisher Express project uses, ePublisher detects the change the next time you open a project that uses the Stationery, notifies you that the Stationery used by the project has been modified, and prompts you to synchronize your project with the updated Stationery. ePublisher Express prompts you to synchronize your project with its Stationery file under the following conditions:

- ePublisher detects differences between the project manifest file and the Stationery manifest file.
- ePublisher detects modifications to the Stationery file used by the project.

When you synchronize your project with Stationery, you update your project file so that the information in your project file matches the information in the Stationery file and in the Stationery manifest file. Synchronizing the project file with the Stationery file and the manifest file ensures all of the settings and information in the project file match all of the settings and information in the Stationery file. For more information about the Stationery file and the Stationery manifest file, see "Manifest Files" and "Stationery Files".

Based on your ePublisher implementation, after you create an ePublisher Express project using Stationery, you can customize target settings for the targets available in your project if you have appropriate permissions. You can only customize target settings in your ePublisher Express project if you have target setting modification permissions. Any customizations you make to target settings will be overwritten the next time you synchronize your ePublisher Express project with Stationery. For more information, see "Working with Target Settings".

ePublisher Express allows you to synchronize your project with its associated Stationery using one of the following methods:

- Automatically synchronize projects with Stationery. For more information, see "Automatically Synchronizing ePublisher Express Projects with Stationery".

- Manually synchronize projects with Stationery. For more information, see “Manually Synchronizing ePublisher Express Projects with Stationery”.

## Manifest Files

When you create a project based on Stationery in ePublisher Express, ePublisher copies the manifest file used by the Stationery you specify for the project and places a copy of the Stationery manifest file in the project folder for the new ePublisher Express project. The manifest file is a record of all of the files associated with the Stationery file, including all of the files listed in the following project folders:

- `Formats` folder
- `Targets` folder
- `Files` folder

For more information about project folders, see “Project Folder Structure”.

Any time the Stationery designer performs one of the following actions in the Stationery `Formats`, `Targets`, or `Files` folder, ePublisher updates the Stationery manifest file:

- Modifies a file in a folder
- Adds a file to a folder
- Removes a file from a folder

When you open an existing ePublisher Express project, ePublisher compares the ePublisher Express project manifest file to manifest file of the Stationery associated with the ePublisher Express project and determines if there are differences between the manifest file.

If the Stationery designer has updated, removed, or added any files to the `Formats`, `Targets`, and `Files` folder in the Stationery since the last time you opened your ePublisher Express project, ePublisher detects these differences and prompts you to synchronize your ePublisher Express project with the Stationery file. When you synchronize your ePublisher Express project with the Stationery file, ePublisher copies the Stationery’s updated manifest file over to your ePublisher Express project file and adds, removes, and updates files in the `Formats`, `Targets`, and `Files` folders for your project as appropriate.

For example, assume that the Stationery designer updated the Stationery you use for one of your projects by adding a new `Page.asp` file. When the Stationery designer makes this change, ePublisher updates the Stationery manifest file with the change. After the Stationery designer makes this change, the next time you open up your ePublisher Express project that uses the changed Stationery,

ePublisher Express recognizes that the ePublisher Express project manifest file is different than the Stationery project file and prompts you to synchronize your project to your Stationery file. When you synchronize your project, ePublisher adds the new `Page.asp` file to your project folders.

## Stationery Files

When you open an existing ePublisher Express project, ePublisher Express determines if the Stationery used by the project has been modified by examining the checksum of the Stationery file. A checksum is a value that depends on the contents of a file. ePublisher uses the checksum to determine if a Stationery the file has changed. If the checksum of the Stationery file is different than the checksum of the project file, ePublisher Express prompts you to synchronize your project with the Stationery file associated with your project. Any changes to the following settings within the Stationery file will affect the checksum:

- Style and format information
- Conditions
- Variables
- Cross-reference definitions
- Target settings

## When to Synchronize

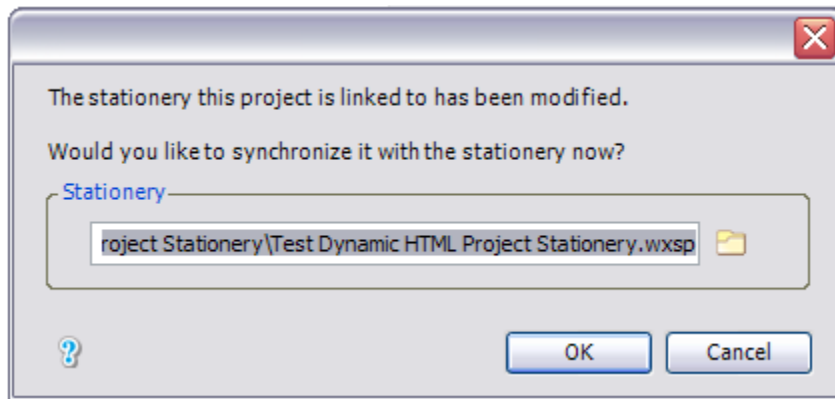
All ePublisher Express projects should be synchronized with Stationery any time the Stationery designer modifies the Stationery. ePublisher Express projects must be synchronized with the Stationery in order for ePublisher to include the changes made by the Stationery designer to the Stationery file in an ePublisher Express project. When you synchronize an ePublisher Express project (`.wfp` file) with Stationery (`.wxsp` file), ePublisher updates the information in the ePublisher Express project to match the information in the Stationery file. If you choose not to synchronize, your project will retain its old settings and the information in the project file will not match the information in the Stationery file until you synchronize.

## Automatically Synchronizing ePublisher Express Projects with Stationery

When you open an existing project, ePublisher Express automatically detects whether any modifications have been made to the Stationery file. If any changes have been made to the Stationery, ePublisher Express displays a window notifying you that the Stationery has been modified. When this window displays, you can choose to synchronize your project to the modified Stationery file. You can also choose to synchronize your project to new Stationery.

## To automatically synchronize an ePublisher Express project with Stationery

1. Open ePublisher Express. If the Stationery designer has modified the Stationery linked to your ePublisher Express project, ePublisher Express displays a window that tells you that the Stationery the ePublisher Express project is linked to has been modified. The window ePublisher displays should be similar to the following window.



2. ***If you want to synchronize your ePublisher Express project with the specified Stationery***, click **Yes**.
3. ***If you want to synchronize your ePublisher Express project with different Stationery***, complete the following steps:
  - a. Click the folder icon, and then browse to the location of the Stationery with which you want to synchronize your ePublisher Express project.
  - b. Select the Stationery (.wxsp file), and then click **Open**.
  - c. Click **OK** again.
4. ***If you do not want to synchronize your ePublisher Express project with Stationery***, click **Cancel**.

## Manually Synchronizing ePublisher Express Projects with Stationery

You can manually synchronize your project file with Stationery at any time. When you manually synchronize your project file with Stationery, ePublisher Express prompts you to specify the Stationery with which you want to synchronize your ePublisher Express project. You can synchronize your ePublisher Express project with the Stationery currently associated with your ePublisher Express project, or you can specify that you want your ePublisher Express project to synchronize with different Stationery.

## To manually synchronize an ePublisher Express project with Stationery

1. In ePublisher Express, on the **File** menu, click **Synchronize with Stationery**.

**Note:** You can only synchronize ePublisher Express projects with Stationery. You cannot synchronize ePublisher Designer projects with Stationery, because ePublisher Designer projects are not based on Stationery. ePublisher Designer projects are used to design Stationery.

2. Browse to the location of the Stationery to which you want to synchronize your ePublisher Express project. By default, ePublisher saves Stationery to the following folder:

`My Documents\ePublisher Stationery\ProjectName`, where `ProjectName` is the name of the project used to create the Stationery.

3. Select the Stationery (`.wxsp` file), and then click **Open**. ePublisher synchronizes the ePublisher Express project with the specified Stationery.

## Project Information that is not Synchronized

Most items are synchronized with Stationery but the following are not:

- Merge Settings
- All settings in the Document Manager Pane
- Preferences

## Deleting Projects

Delete a project when you no longer want to use the project to generate output.

## To delete a project

1. Open Windows Explorer.
2. Browse to the location of the project folder for the project you want to delete.
  - By default ePublisher saves ePublisher Express project files in the `My Documents\ePublisher Express Projects\ProjectName` folder, where `ProjectName` is the name of the project. ePublisher Express project files use the `.wrp` file extension
  - By default ePublisher saves the ePublisher Designer project files in the `My Documents\ePublisher Designer Projects\ProjectName` folder, where `ProjectName` is the name of the project. ePublisher Designer project files use the `.wep` file extension.
3. Delete the project folder.

When you delete a project, ePublisher continues to display the project on the Start Page until you close and then reopen the ePublisher user interface.

# Generating and Regenerating Output

When you generate output in ePublisher, ePublisher creates all of the files specified for the target. ePublisher uses the information in the project source documents and project settings to generate output files. Output files include the following types of files:

- Individual topic page `.html` files
- Image files, such as `.jpg`, `.gif`, and `.png` files
- The entry-point file, which is used to open the generated output
- All files required by the help system if you are generating output for a help system

## Output Generation and Regeneration

In ePublisher, you generate output using either the generate or regenerate option. The generate and regenerate option both create output from your project. However, there are some important differences between the options.

As you make changes to your source documents and your project settings, you need to generate output files in order to see any changes made to the following items:

- Content changes in your source documents

- Changes to project settings
- Changes in the Stationery associated with your ePublisher project

When you *generate* output for a target for the first time, ePublisher creates the output files for the first time. After you generate output files for a target the first time, if you generate output for your target again, you *update* your output files with the changes you made in your source documents and the changes you made to your project settings. Generating, or updating, your output creates output files more quickly than regenerating your output files.

Use the generate option when you have made changes to the following project settings:

- Condition settings
- Variable values
- Cross-reference definitions
- Merge settings
- Target settings
- Project preferences

When you *regenerate* output, ePublisher deletes the `Data` folder from the project folder, creates a new `Data` folder, and *creates* new output files each time. Regenerate your output any time you add new information to your source documents that is not content. Non-content modifications to source documents include adding, removing, or modifying following items:

- Paragraph, character and table styles and formats
- Marker types
- Cross-reference definitions
- Variable values in the source documents
- Condition settings in the source documents

## Generating Output

In ePublisher, you can generate output for the following items:

- The entire project, which generates output for all the groups and source documents in your project

- A single group within your project
- An individual source document within your project

Generate output for all of the groups in your project when you are generating the final, completed output or help system, when you are merging output or help systems, or when you are deploying your output.

Generate output for a single group if you have already generated output for the other groups in your project, but you have made some slight modifications to one of the groups. Using ePublisher to generate output for a single group reduces the amount of time it takes ePublisher to generate output for your project. When you generate output for a single group, ePublisher generates output for all of the source documents within the group. If you select a top-level group or a group that contains subgroups, ePublisher generates output for all of the source documents in the group and its subgroups.

Generate output for an individual source document if you have made some slight modifications to a source document and want to preview what your generated output will look like. Selecting an individual source document instead of generating output for the entire group or project reduces the amount of time it takes ePublisher to generate output.



## To generate output

1. On the **Project** menu, select the target next to **Active Target** for which you want to generate output.
2. ***If you want to generate output for an entire project***, on the **Project** menu, click **Generate All**.
3. ***If you want to generate output for a group in your project***, complete the following steps:
  - a. In **Document Manager**, select the group for which you want to generate output.
  - b. On the **Project** menu, click **Generate Selected**.
4. ***If you want to generate output for an individual source document in your project***, complete the following steps:
  - a. In **Document Manager**, select the document for which you want to generate output.
  - b. On the **Project** menu, click **Generate Selected**.

**Note:** Some formats, such as WebWorks Reverb (1 &2), must be deployed to a server for observing full functionality. WebWorks Reverb (1 & 2) does provide a convenience web server that can be used for quick, non-production preview purposes. Refer to [Deploying Output to Output Destinations](#) for further information.

## Regenerating Output

When you regenerate output, ePublisher deletes the `Data` folder from the project folder, creates a new `Data` folder, and generates new output files.

Regenerate your source document any time you have added new information to your source document that is not content, including adding, removing, or modifying the following items:

- Character styles
- Paragraph styles
- Table styles
- Cross-reference definitions
- Variable values within source documents

- Conditions settings within source documents

### To regenerate output

1. On the **Project** menu, select the target next to **Active Target** for which you want to regenerate output.
2. On the **Project** menu, click **Regenerate All**.

## Generating Output from FrameMaker or Microsoft Word

ePublisher provides the ability to generate output and reports from both FrameMaker and Word using a custom menu, often referred to as **WebWorks Transit** or **WebWorks Menu**. Output and reports generated via this menu are short-lived. They disappear once the project window is closed. Long lived projects should be created with the classic Express interface.

## In your Source Document

1. Go to the **WebWorks** menu
2. Select **ePublisher Express -> Generate Output**
3. Select the Stationery file on which you want the project to be based
4. Click **OK** once you have selected the Stationery and the Target you want to use. Now click **Finish** to generate output. ePublisher provides you a window to view the Output Explorer as well as the generated output

## Modifying Help System Title Bars

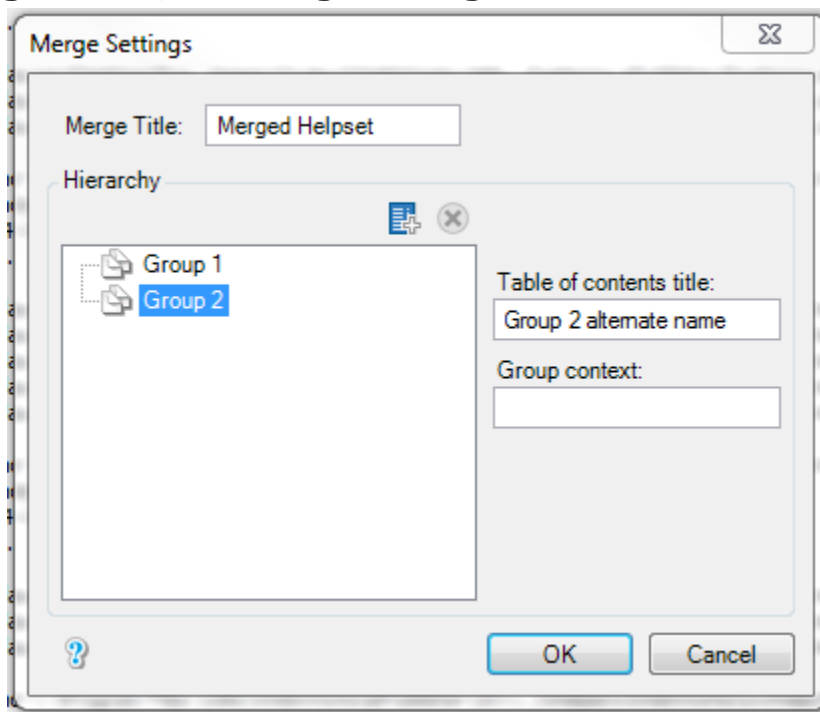
The title bar in your generated help system displays the title you assigned to your project. If you want to specify a different title in the title bar for your generated help system, you can do this in the Merge Settings window if you are generating output for the following help systems:

- Dynamic HTML
- Eclipse Help
- Microsoft HTML Help
- WebWorks Help
- WebWorks Reverb

You cannot use merge settings to modify help system title bars for other output formats.

## To modify the title bar of a help system

1. On the **Project** menu, select the target next to **Active Target** for which you want to modify the title bar of a help system.
2. On the **Target** menu, click **Merge Settings**.



3. In the **Merge Title** field, type the title you want to display in the title bar for your generated help system, and then click **OK**.
4. On the **File** menu, click **Save**.
5. Regenerate your output. For more information, see "Regenerating Output"

## Viewing Output

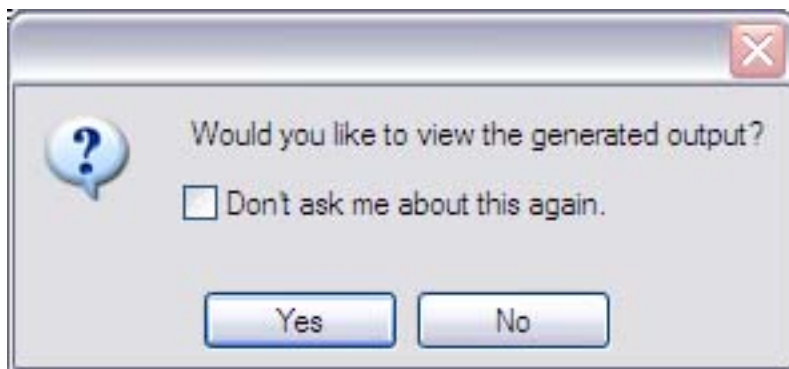
After you specify project and target settings, generate output for your target to review your changes and verify that the generated output displays and functions properly. You can generate output for all of the source documents and groups in your project, or you can generate output for a single group or source document. You can view your generated output files in one of the following ways:

- View output by automatically opening the generated output. For more information, see "Viewing Output by Automatically Opening Generated Output".

- View output in Output Explorer. For more information, see “Viewing Output in Output Explorer”.
- View output in the `Output` folder. For more information, see “Viewing Output in the Output Folder”.

## Viewing Output by Automatically Opening Generated Output

When you generate or regenerate output for a target, after ePublisher generates output, ePublisher prompts you to view the generated output by displaying the following window:



## To view output by automatically opening the output

1. Generate or regenerate output. For more information, see “Generating Output” or “Regenerating Output”.
2. When ePublisher displays a window asking if you would like to view the generated output, perform one of the following actions:
  - ***If you want to view the generated output***, click **Yes**.
  - ***If you do not want to view the generated output***, click **No**.
  - ***If you want ePublisher to automatically open the output each time you generate output and you do not want ePublisher to ask you each time if you want to view the generated output***, select the **Don’t ask me about this again** check box, and then click **Yes**.
  - ***If you do not want ePublisher to automatically open the output each time you generate output and you do not want ePublisher to ask you each time if you want to view the generated output***, select the **Don’t ask me about this again** check box, and then click **No**.

If you select the **Don’t ask me about this again** check box and specify that you always want ePublisher to display the generated output or that you never want ePublisher to display the generated output, ePublisher uses the options you specify as the default behavior for automatically displaying output when you generate or regenerate output. If you later want to change the default behavior, you can clear your preferences in the WebWorks ePublisher Preferences window, and then set new preferences the next time you generate or regenerate output.

## Viewing Output in Output Explorer

Output Explorer allows you to view output files from within the ePublisher user interface. Each time you generate output for a group of source documents or for an individual source document, ePublisher displays the generated output files in Output Explorer. The list of files ePublisher displays in Output Explorer is based on if you have a group of source documents selected or if you have an individual source document selected.

***If you select a top-level group in Document Manager***, ePublisher displays a group folder with the same name as the top-level group in Output Explorer that contains the following items:

- Navigation group. The Navigation group displays the generated entry-point file and printable reports. The entry-point file is the file that opens the generated output.

- Reports group. The Reports group displays any reports associated with the target that ePublisher generated.

***If you select a source document in Document Manager,*** ePublisher displays the source document group with the same name as the source document selected in Document Manager that contains the following items:

- Files group. The Files group contains all of the generated content files and printable reports.
- Images group. The Images group contains images associated with the source document.
- Reports group. The Reports group displays any reports associated with the generated output for the target.

***If you select a subgroup in Document Manager,*** ePublisher does not display any information in the Navigation and Reports groups in Output Explorer, because subgroups do not create a generated entry-point file and do not represent an actual table of contents group in generated output. The entry-point file is the file that opens the generated output.

***If you have two or more top-level groups in Document Manager and your output format supports merged help systems,*** ePublisher creates a Merge Output group in Output Explorer. The Merge Output group contains the entry-point file for the merged help system. For more information about merged help systems, see “Merging Top-level Groups (Multivolume Help)”.



## To view output in Output Explorer

1. ***If Output Explorer is not displayed in the ePublisher user interface***, on the **View** menu, click **Output Explorer**.
2. On the **Project** menu, select the target next to **Active Target** for which you want to view output.
3. ***If you want to view output by opening the entry-point file***, complete the following steps:
  - a. In Output Explorer, select a top-level group.
  - b. Click on the plus sign next to the top-level group to expand the group.
  - c. Click on the plus sign next to the **Navigation** group to expand the group.
  - d. Double-click on the entry-point file to open the generated output.

Output Type Generated	Default Entry-Point File to Double-Click to Open
WebWorks Reverb	index.html
Dynamic HTML	toc.html
WebWorks Help	index.html
Eclipse Help	View Eclipse Help
Microsoft HTML Help	name.chm
Oracle Help	name.jar
Sun JavaHelp	name.jar

**4. If you generated output for an HTML-based output format and you want to view the individual HTML files generated for a specific document,** complete the following steps:

**Note:** By default, ePublisher produces individual HTML files for HTML-based output formats based on the page breaks settings you specify for your project. For more information about specifying page break settings, see “Specifying Page Breaks Settings”.

- a. In Document Manager, select a source document.
- b. In the Output Explorer, click on the plus sign next to the document to expand the group.
- c. Click on the plus sign next to the **Files** group to expand the group.
- d. Double-click on the generated output file to open the file.

**5. If your output format supports merged help systems and you want to view the entry-point file for a merged help system,** complete the following steps:

- a. In Output Explorer, click on the plus sign next to the **Merged Output** group in the Output Explorer to expand the group.

- b. Double-click on the entry-point file to open the generated output.

## Viewing Output in the Output Folder

ePublisher stores generated output pages and images in the `Output` folder. By default, ePublisher creates an `Output` folder in the following location:

- ***If you are creating a project using ePublisher Express***, by default ePublisher creates the `Output` folder in the `My Documents\ePublisher Express Projects\ProjectName` folder, where `ProjectName` is the name of the project.
- ***If you are creating a project using ePublisher Designer***, by default ePublisher creates the `Output` folder in the `My Documents\ePublisher Designer Projects\ProjectName` folder, where `ProjectName` is the name of the project.

The `Output` folder contains individual output folders for each one of your targets. For example, if your project contains targets for WebWorks Help, Microsoft HTML Help, and Dynamic HTML, then there will be three folders, one for each of these targets, in the `Output` folder.

You can view output files for all output formats by opening them directly from the `Output` folder.

You can also view output files for all output formats by opening them from the ePublisher user interface. When you open output files from the ePublisher user interface, ePublisher opens the `Output` folder for the active target you are currently working with in ePublisher. For more information about specifying an active target and working with targets, see “Specifying Active Targets” and “Working with Targets”.

## To view output in the Output Folder

1. On the **Project** menu, select the target next to **Active Target** for which you want to view output.

**Note:** You must generate output before you can view output in the `Output` folder. For more information about generating output, see “Generating Output”.

2. On the **View** menu, click **Output Directory**. ePublisher opens Windows Explorer and displays a folder based on the name of your target. This `Output` folder contains the output files ePublisher generated for the active target.

## Changing the Location of the Output Folder

When you generate output, ePublisher places the output files into the Output folder. You can modify the location where ePublisher stores your output files.

**Note:** We do not recommend doing this, it makes the project non-portable

## To change the default location of the Output folder

1. On the **Project** menu, select the target next to **Active Target** for which you want to view output.
2. On the **Target menu**, click **Target Settings**.
3. In the **Generated output location** field, type the path to the folder where you want ePublisher to place the generated output, or click the folder icon to browse to and select a folder.
4. Click **OK**.

## Working with Output Log Files

Each time you generate output for a target, ePublisher creates a log file named `generate.log` and writes the following information to the log file:

- Time when output generation began
- Actions and commands ePublisher performed, such as processing, creating and copying files
- Pipelines processed by ePublisher
- Any messages, warnings, or errors generated by ePublisher when ePublisher generated output for the target
- Time when output generation ended
- Total amount of time it took ePublisher to generate the output

## To work with output log files for a target

1. If you want to view the log file for a target from within the ePublisher user interface, complete the following steps:
  - a. On the **Project** menu, select the target next to **Active Target** for which you want to view log files.
  - b. On the **View** menu, click **Log Window**.
2. If you want to save the log file as a `.txt` file, complete the following steps:
  - a. Click the **Save** button, located in the upper-right corner of the Log Window.
  - b. Specify a name for the log file and the location where you want to save the log file, and then click **Save**.
3. If you want to view the log file for a target using Windows Explorer, in Windows Explorer browse to one of the following locations:
  - **If you are using ePublisher Express**, browse to the `ProjectName\Logs\TargetName` folder, where `ProjectName` is the name of the project and `TargetName` is the name of the target for which you generated output. By default ePublisher saves project files for ePublisher Express projects in the `My Documents\ePublisher Express Projects\ProjectName` folder, where `ProjectName` is the name of the project.
  - **If you are using ePublisher Designer**, browse to the `ProjectName\Logs\TargetName` folder, where `ProjectName` is the name of the project and `TargetName` is the name of the target for which you generated output. By default ePublisher saves project files for ePublisher Designer projects in the `My Documents\ePublisher Designer Projects\ProjectName` folder, where `ProjectName` is the name of the project.

## Validating Output Using Reports

After you generate output, you can validate your output using ePublisher reports. ePublisher reports contain information about how ePublisher processed items in your source documents when ePublisher generated output. Reports also allow you to identify any problems that occurred when ePublisher generated output. If reports display notifications, such as messages, warnings, or errors, you can correct the items in your source documents that caused the error. You can then generate output again and then review the reports again to verify that any issues have been addressed as needed.

ePublisher provides the following types of reports:

- Accessibility reports. For more information, see “Accessibility Reports”.
- Baggage Files reports. For more informations, see “Baggage Files Reports”.
- Conditions reports. For more information, see “Conditions Reports”.
- Filenames reports. For more information, see “Filenames Reports”.
- Links reports. For more information, see “Links Reports”.
- Styles reports. For more information, see “Styles Reports”.
- Topics reports. For more information, see “Topics Reports”.
- Images reports. For more information, see “Images Reports”.
- Printable reports. For more information, see “Printable Reports”.

For more information about configuring and generating reports, see “Configuring Reports” and “Generating Reports”.

## Accessibility Reports

You can use markers in your source documents to create accessible online content.

You can use Accessibility reports to validate that the online content you generate using ePublisher meets your accessibility requirements. Accessibility reports provide notifications on the following items when ePublisher generates output:

- Images without alternative text
- Image maps without alternative text
- Images without long descriptions
- Tables without summaries

Configure the notifications you want ePublisher to generate for Accessibility report settings before you generate Accessibility reports. For more information about configuring Accessibility report settings, see “Configuring Reports”. For more information about generating Accessibility reports, see “Generating Reports”.

## Baggage Files Reports

You can generate baggage files by adding links to HTML or PDF files in your file system, in your source documents. For more information about baggage files, see “Targets” and “Specifying Baggage Files Settings”.

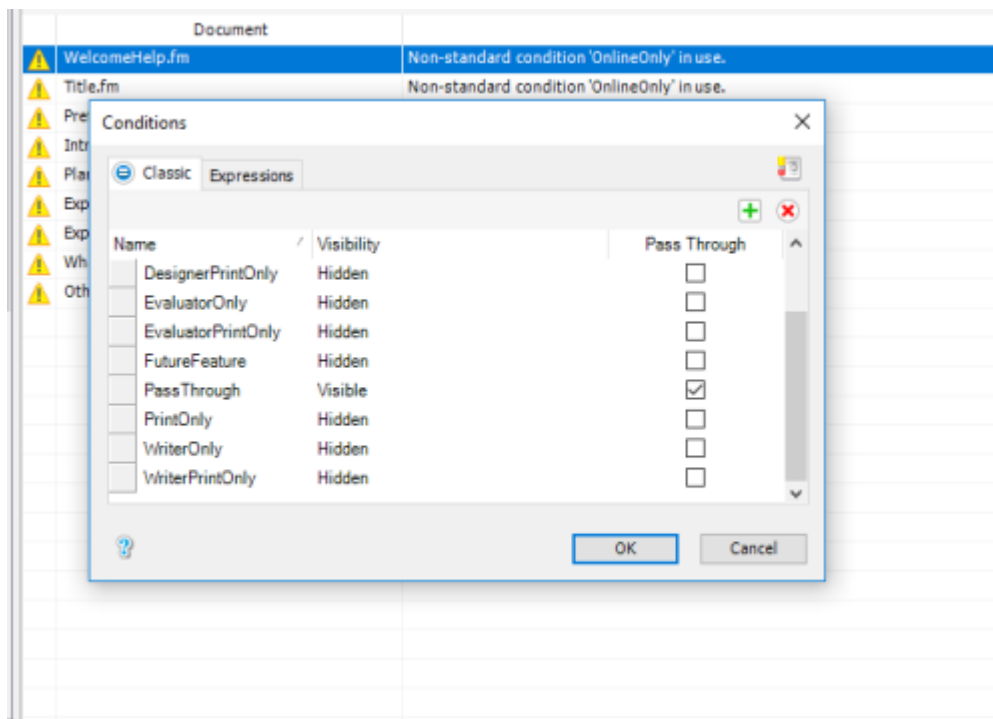
You can use Baggage Files reports to obtain information about how the baggage files are going to be shown in your Search Results. Baggage Files reports provide notifications on the following items when ePublisher generates output:

- Baggage files without summary
- Baggage files without title

Configure the notifications you want ePublisher to generate for Baggage Files report settings before you generate Baggage Files reports. For more information about configuring Baggage Files report settings, see “Configuring Reports”. For more information about generating Baggage Files reports, see “Generating Reports”.

## Conditions Reports

If you add new conditions after ePublisher already scanned the document, those new conditions won’t be picked up by ePublisher. To warn you about that, ePublisher has a Conditions Report where you can see all those details, and re-scan the document so the new conditions can be picked up or add it manually yourself using the UI:



## Filename Reports

You can specify names for output files using Filename markers.



You can use Filenames reports to validate that ePublisher named your output files correctly using the Filename markers you inserted in your source documents. The Filenames report displays the name of the Filename marker you inserted into your source document and the name of the output file ePublisher generated based on the Filename marker. The Filenames report also provides notifications on the following items when ePublisher generates output:

- The files ePublisher created that correspond to the Filename markers you inserted into your source documents
- If ePublisher ignored a Filename marker when generating output
- If duplicate Filename markers exist in the source documents used by your project to generate output

Configure the notifications you want ePublisher to generate for Filenames report settings before you generate Filenames reports. For more information about configuring Filename report settings, see “Configuring Reports”. For more information about generating Filename reports, see “Generating Reports”.

## Links Reports

You can use Links reports to verify that the links you specify to items in your source documents resolve and that ePublisher processed links in your source documents to the item referenced by the link correctly. Links reports provide notifications on the following items:

- Baggage files
- External URLs
- Unresolved links to items in other documents
- Unresolved links to missing source document
- Unresolved links to missing files
- Unresolved link within source documents document
- Unsupported baggage files
- Unsupported external URLs
- Unsupported group to group links

For the definition of a baggage file see “Targets”.

Configure the notification you want ePublisher to generate for Links report settings before you generate Links reports. For more information about configuring Links

report settings, see “Configuring Reports”. For more information about generating Links reports, see “Generating Reports”.

## Styles Reports

Styles reports allow you to verify that your source documents conform to the styles and formatting defined in the Stationery by the Stationery designer. The Styles report notifies you about the following items when ePublisher generates reports:

- Any non-standard styles used in your source documents
- Any style overrides used in your source documents

A non-standard style is any style that exists in your source document but is not defined in the stationery file used by your project. For example, if you add a new style to your source document called `BodyIndent 4`, but your stationery designer has not updated the stationery file to include the `BodyIndent 4` style, the Styles report notifies you that there is a non-standard style used in the source document.

A style override is any modification you made to the original style definition for a particular instance of a style. For example, if you have applied the `Body` paragraph style to a paragraph in your source document, and you then apply the `Bold` character style to the paragraph, the `Body` paragraph style has a style override.

If your source document contains any non-standard styles or style overrides, ePublisher will process your source documents when you generate output using the non-standard styles and style overrides you applied in your source documents.

Configure the notifications you want ePublisher to generate for Styles report settings before you generate Styles reports. For more information about configuring Styles report settings, see “Configuring Reports”. For more information about generating Styles reports, see “Generating Reports”.

## Topics Reports

Context-sensitive help topics require that you have TopicAlias markers inserted in your source documents. ePublisher generates context-sensitive help topics based on the topic IDs you specify for each TopicAlias marker you insert in your source documents. Each time ePublisher detects a TopicAlias marker in a source document, ePublisher generates a context-sensitive help topic based on the topic ID.

You can use the Topics Report to verify that context-sensitive help topics have been created for each topic ID specified in your source document. The Topics Report lists the topic ID and the topic file created for each topic ID.

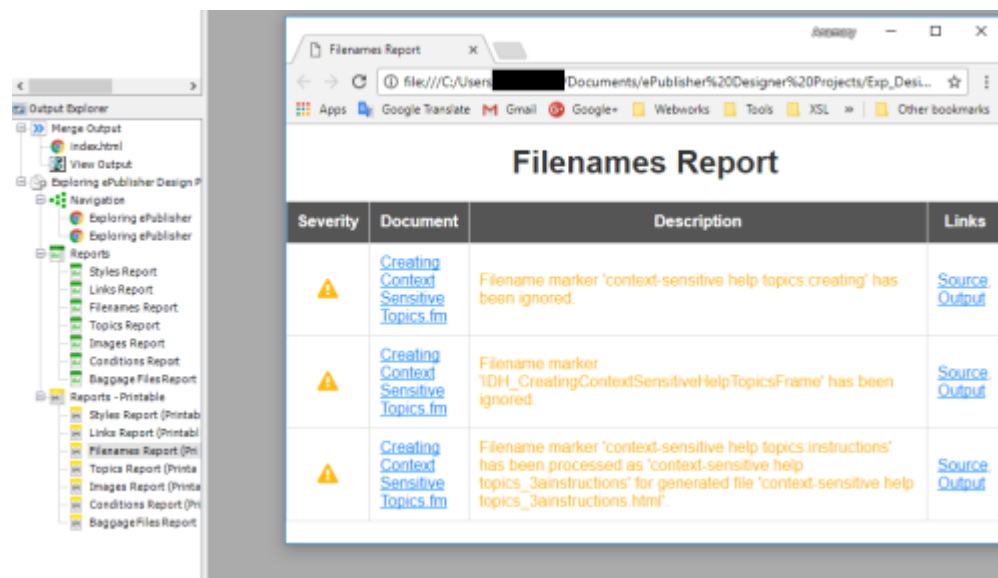
Configure the notifications you want ePublisher to generate for Topics report settings before you generate Topics reports. For more information about configuring Topics report settings, see “Configuring Reports”. For more information about generating Topics reports, see “Generating Reports”.

# Images Reports

Image reports enable you to verify the integrity and appearance of ePubublisher manage images. Users are notified any time a source image is missing or when an image occurs in a problematic structure, such as images within tables in the ePub format.

# Printable Reports

With the new ePubublisher you can see your reports in two different ways: from the UI and in your browser (Printable Reports). You can find them right after your usual reports.



# Configuring Reports

When you use reports to validate your output, you must specify the type of notification that you want to display when ePubublisher detects issues or performs actions while generating output using your source documents. When ePubublisher generates output, ePubublisher generates notifications under the following conditions:

- When ePubublisher cannot properly process elements
- When ePubublisher encounters missing information

For example, ePubublisher can generate a notification when it detects a potential error in your source documents when you generate output, such as an unresolved cross reference. ePubublisher can also generate a notification when it performs a specific action using elements contained in your source documents, such as when

ePublisher generates an output file using a filename you specified using a Filename marker.

You can specify the following values for report options when you generate output:

### **Ignore**

Specify this value if you do not want ePublisher to report any issues it identifies in the report. For example, specify this value you do not want the Styles report to report any style overrides.

### **Message**

Specify this value if you want to receive a message when ePublisher completes or fails to complete an action. For example, if you are not concerned if your source document uses non-standard styles, but you would like to see where non-standard styles are used in your source documents, specify this value.

### **Warning**

Specify this value if you want to receive a warning when ePublisher completes or files to complete an action. For example, if you want to be warned when ePublisher detects non-standard styles in your source documents, specify this value.

### **Error**

Specify this value if you want the report to display an error when ePublisher completes or files to complete an action. For example, if you want to receive an error notification when ePublisher detects unresolved cross-references in your source documents, specify this value.

## To configure report notification settings

1. On the Project menu, select the target next to **Active Target** for which you want to configure report notification settings.
2. On the **Target menu**, click **Target Settings**.
3. ***If you want to specify Accessibility report notification settings***, in the **Accessibility Report** area, specify a value for each Accessibility report notification setting you want to configure. For more information about each setting, click **Help**.
4. ***If you want to specify Baggage Files report notification settings***, in the **Baggage Files Report** area, specify a value for each Baggage File report notification setting you want to configure. For more information about each setting, click **Help**.
5. ***If you want to specify Conditions report notification settings***, in the **Conditions Report** area, specify a value for each Condition report notification setting you want to configure. For more information about each setting, click **Help**.
6. ***If you want to specify Filenames report notification settings***, in the **Filenames Report** area, specify a value for each Filename report notification setting you want to configure. For more information about each setting, click **Help**.
7. ***If you want to specify Links report notification settings***, in the **Links Report** area, specify a value for each Link report notification setting you want to configure. For more information about each setting, click **Help**.
8. ***If you want to specify Styles report notification settings***, in the **Styles Report** area, specify a value for each Style report notification setting you want to configure. For more information about each setting, click **Help**.
9. ***If you want to specify Topics report notification settings***, in the **Topics Report** area, specify a value for each Topic report notification setting you want to configure. For more information about each setting, click **Help**.
10. ***If you want to specify Images report notification settings***, in the **Images Report** area, specify a value for each Image report notification setting you want to configure. For more information about each setting, click **Help**.

## Generating Reports

You can generate reports for source documents and baggage files by selecting the group or source document that you want to generate reports for in Document

Manager. Before you generate reports, configure notification settings for each report you want to generate. For more information about configuring report notification settings, see “Configuring Reports”.

## To generate a report

1. In Document Manager, select the group or source document for which you want to generate a report.
1. ***If you want to generate all reports for the selected item,*** on the **Project** menu, click **Generate Reports > All**.
2. ***If you want to generate Accessibility reports for the selected item,*** on the **Project** menu, click **Generate Reports > Accessibility Report**.
3. ***If you want to generate Baggage Files reports for the selected item,*** on the **Project** menu, click **Generate Reports > Baggage Files Report**.
4. ***If you want to generate Conditions reports for the selected item,*** on the **Project** menu, click **Generate Reports > Conditions Report**.
5. ***If you want to generate Filename reports for the selected item,*** on the **Project** menu, click **Generate Reports > Filenames Report**.
6. ***If you want to generate Links reports for the selected item,*** on the **Project** menu, click **Generate Reports > Links Report**.
7. ***If you want to generate Styles reports for the selected item,*** on the **Project** menu, click **Generate Reports > Styles Report**.
8. ***If you want to generate Topics reports for the selected item,*** on the **Project** menu, click **Generate Reports > Topics Report**.
9. ***If you want to generate Images reports for the selected item,*** on the **Project** menu, click **Generate Reports > Images Report**.

## Report Messages

The following tables provide descriptions for report messages.

### Accessibility Report Messages

The following table lists messages in Accessibility reports.

<b>Message</b>	<b>Definition</b>
Table is missing a table summary.	The table does not contain a table summary. Insert a table summary marker within the table.
Image link '{0}' is missing alternate text.	The hotspot does not have alternate text. Insert an image area alternate text marker in a text frame within the image.
Image is missing alternate text.	The image does not have alternate text. Insert an image alternate text marker in a text frame within the image.
Image is missing a long description	The image does not have a long description. Insert an image long description marker in a text frame within the image.

## Baggage Files Report Messages

The following table lists messages in Baggage Files reports.



Message	Definition
Title missing for '{0}'.	<p>The baggage file doesn't have a title defined.</p> <p>If it's an <code>HTML</code> baggage file insert a <code>&lt;title&gt;</code> tag in the <code>&lt;head&gt;</code> tag of the <code>HTML</code>. Or add the <code>@title</code> attribute to that file entry in your baggage list info file.</p> <p>If it's a <code>PDF</code> baggage file add the <code>@title</code> attribute to that file entry in your baggage list info file.</p>
Summary missing for '{0}'.	<p>The baggage file doesn't have a summary defined.</p> <p>If it's an <code>HTML</code> baggage file you can do one of these:</p> <ul style="list-style-type: none"> <li>• insert a <code>&lt;meta&gt;</code> tag in the <code>&lt;head&gt;</code> tag of the <code>HTML</code> with <code>@name='summary'</code> and <code>@content</code> with the summary you want to define,</li> <li>• create any kind of tag inside the <code>&lt;body&gt;</code> tag that accepts the attribute <code>@class</code> with <code>@class='summary'</code> and then place your summary as the content of the element,</li> <li>• insert a <code>&lt;meta&gt;</code> tag in the <code>&lt;head&gt;</code> tag of the <code>HTML</code> with <code>@name='description'</code> and <code>@content</code> with the summary you want to define.</li> </ul> <p>Or add the <code>@summary</code> attribute to that file entry in your baggage list info file.</p> <p>If it's a <code>PDF</code> baggage file add the <code>@summary</code> attribute to that file entry in your baggage list info file.</p>

## Filename Report Messages

The following table lists messages in Filename reports.

<b>Message</b>	<b>Definition</b>
File '[NAME]' has been processed as a baggage file.	Any files not contained within your project are processed as baggage files.
Filename marker '[NAME]' has been used for generated file '[FILE PATH]'.	A file has been generated using a filename marker. This alerts you that the name of the file has been changed.
Filename marker '[NAME]' has been ignored.	The filename marker has been ignored because it is either uses a duplicate name or it has not been inserted at a heading that splits.
Filename marker '[NAME]' has been processed as '[NAME]' for generated file '[NAME]'.	The filename marker has not been used; instead, the file has been renamed to the filename indicated.

## Links Report Messages

The following table list messages in Links reports.

Message	Definition
Unresolved link to target '[NAME]' in document '[NAME]'.	There is an unresolved cross-reference in the document. The destination target either does not exist or cannot be found.
Unresolved link from document '[NAME]' to target '[NAME]' in document '[NAME]'.	There is an unresolved cross-reference from a document to a location in another document. The destination target cannot be found.
Unresolved link from document '[NAME]' to document '[NAME]'.	There is an unresolved link from one document to another document. It cannot find the referenced document.
Unresolved link from document '[NAME]' to missing file '[NAME]'.	There is an unresolved link from a document to an external file. A file refers to any file that is not part of the ePublisher project or is not of the same type as your source document (for example, <code>.jpeg</code> , <code>.gif</code> , <code>.tif</code> )
Unresolved link from document '[NAME]' to document '[NAME]'. Output format does not support group to group linking.	There is an unresolved cross-reference from one document to another document because the output format your project is using does not support linking from one top-level group to another.
Unresolved link from document '[NAME]' to file '[NAME]'. Output format does not support baggage files.	There is an unresolved cross-reference from the document to a file because the output format your project is using does not support baggage files. Files refer to any file that is not part of the project.
External URL link '[NAME]' is not supported.	The output format does not support external links.

## Styles Report Messages

The following table lists messages in Styles reports.

Message	Definition
Encountered [STYLE TYPE] style name '[NAME]' in your source file that is not defined in ePublisher.	The style name is not defined in ePublisher.
Encountered text with [STYLE TYPE] style name '[NAME]' that has modified style properties in your source file.	There is a style override. Style overrides refer to attributes that are defined within the style.
Encountered style properties not associated with a named style in your source file.	There is a style override. For example, the character style <i>bold</i> has been modified in one instance of its use.

## Topics Report Messages

The following table lists messages in Topics reports.

Message	Definition
Topic '[NAME]' resolves to the file '[FILE PATH]'.	A topic page has been created for the topic alias marker.
Topic '[NAME]' is duplicated in the file '[FILENAME]'	A duplicate topic alias has been created in that file.

## Images Report Messages

The following table lists messages in Images reports.

<b>Message</b>	<b>Definition</b>
Missing by-reference source files	An image referenced by the source document is missing.
Images in table cells	Image occurs inside a table cell (problematic for certain ePUB readers)

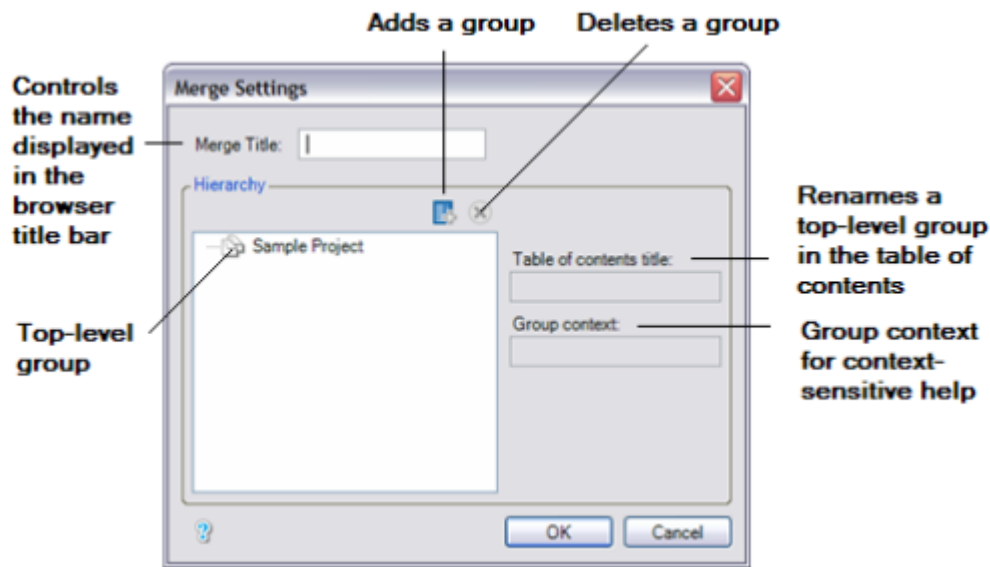
## Merging Top-level Groups (Multivolume Help)

Merged help, which is sometimes also referred to as multivolume help, is a help system with a single set of files created from output from multiple top-level groups from within a project. Merged help takes the table of contents, index, and search data from each top-level group entry-point file and combines this information to create a single, consolidated help system. You can use ePublisher to create merged help systems for the following output formats:

- WebWorks Reverb (1.0 and 2.0)
- WebWorks Help
- Eclipse Help
- Microsoft HTML Help

If you have created several top-level groups in Document Manager for your project, by default ePublisher generates its own help system with its own entry-point file when you generate output for your project. The entry-point file is the file that opens the help system. ePublisher places the merged help system in the Merged Output group in Output Explorer.

You must have at least two top-level groups in Document Manager to create merged help. By default, ePublisher uses the organizational structure specified in Document Manager to create the merged, or multivolume help system. If you want to organize and group your top-level groups using a different name than the group name specified in the Document Manager, or if you want to use a different hierarchy in your merged help system than the hierarchy you currently have specified for your project in Document Manager, you can do this using merge settings. The following figure shows the Merge Settings window.



ePublisher names the merged help system based on the name of your target. For example, if you generate output for a target named CompanyA WebWorks Help, ePublisher creates an entry-point file for the merged help system named CompanyA WebWorks Help and displays this name in the title bar when users open the merged help system.

ePublisher also creates individual help systems for each top-level group in Document Manager and names these individual help systems based on the names of the top-level groups in Document Manager. For example, if you have three top-level groups in Document Manager named FeatureA, FeatureB, and FeatureC and you are generating output for a target called CompanyA WebWorks Help, ePublisher creates FeatureA, FeatureB, and FeatureC help systems as well as a merged help system named CompanyA WebWorks Help that merges the table of contents, index, and search data from each top-level group into a single, consolidated help system. These top-level groups also display in the table of contents in your merged help system.

You can use ePublisher merge settings to perform the following actions:

- Specify a different name than the target name for the title displayed in the title bar of the merged help system
- Specify a different name for subgroups in your generated output than the names used in Document Manager
- Organize and group your top-level groups in a merged help system into a different hierarchy than the hierarchy used in Document Manager.



## To merge help systems

1. Create the top-level groups you want to use in your merged help system in Document Manager. For more information about creating top-level groups, see "Creating Top-Level Groups".
2. On the **Project** menu, select the target next to **Active Target** for which you want to create a merged help system.
3. On the **Target** menu, click **Merge Settings**.
4. ***If you want to specify a name other than your target name for the merged help system***, in the **Merge Title** field, type in the name you would like to display in the title bar of your merged help system.
5. ***If you want to specify a different name for each top-level group in the table of contents for your generated output***, complete the following steps for each top-level group you want to rename in your generated output:
  - a. In the **Hierarchy** area, select the name of the top-level group for which you want to specify a different name in the generated output.
  - b. In the **Table of contents title** field, type the name you want to display for the group in the generated output.
6. ***If you want to reorganize the table of contents in your merged help system***, select and then drag and drop any of the top-level groups to a new position.
7. ***If you want to create a new custom group for your merged help system that includes some of your existing top-level groups from Document Manager***, complete the following steps:
  - a. Click the **Add** button. The **Add** button in the Merge Settings window is an icon of a blue page with a plus (+) character ePublisher adds a new group called `Untitled Topic` to your table of contents hierarchy.
  - b. Click on the `Untitled Topic` group in the Merge Settings window and rename it.
  - c. Select and then drag and drop the top-level groups you want to include in the new group into the new group.
8. ***If you want to delete a custom group you previously created that contains top-level groups***, complete the following steps:

**Note:** You can only remove groups that you have manually added to your merged help system hierarchy. You cannot remove groups ePublisher creates by default based on the top-level groups in Document Manager.

- a. Select the group you want to remove.
  - b. Click the **Delete** button.
- 9. ***If you are generating merged, or multivolume WebWorks Help or WebWorks Reverb that includes context-sensitive help***, in the **Group context** field, specify the help context for each top-level group to use.

**Note:** In WebWorks Reverb, you can optionally include the context as a parameter when you have more than one instance of the same TopicAlias value in a multivolume help set.

- 10. Click **OK**.
- 11. Generate you output. For more information, see “Generating Output”.
- 12. Open the merged help system by completing one of the following steps:

**Note:** ePublisher creates the entry-point file using the name of the selected target. If you want to change the name of the entry-point file for the merged help system, rename your target. For more information about renaming your target, see “Renaming Targets”.

- a. On the **View** menu, click **Output Explorer**.
- b. Under the Merge Output group in the Output Explorer, double-click on the entry-point file for the merged help system to open the merged help system.

**Note:** Ensure you click under the Merge Output group in Output Explorer. You must click under the Merge Output group in Output Explorer in order to view the merged output. If you click under one of the other groups, you will only see the output generated for the specific group selected.

- 13. Review the merged help system you created based on the merge settings you specified and confirm that your merged help system displays using the help system name and table of contents group hierarchy that you want.

## Deploying Output

This section explains how you can use ePublisher to deploy output to multiple locations, such as to folders on a network, or to a Web server.

## Output Deployment

By default ePublisher places output files in the following location on your local computer:

- **If you are creating a project using ePublisher Express**, by default ePublisher creates the `Output` folder in the `My Documents\ePublisher Express Projects\ProjectName` folder, where `ProjectName` is the name of the project.
- **If you are creating a project using ePublisher Designer**, by default ePublisher creates the `Output` folder in the `My Documents\ePublisher Designer Projects\ProjectName` folder, where `ProjectName` is the name of the project.

If you would like to deploy your output files to another location in addition to this default location after ePublisher generates output, such as a folder on a network, you can deploy your output to one or more output destinations using ePublisher. The **output destination** is the location where you would like to deploy your generated output files. In ePublisher, the output destination consists of the following components:

- Output name
- Output destination location

To deploy your output, you must perform the following steps:

1. Create one or more output destinations. For more information, see “Creating Output Destinations”.
2. Specify an output destination for each target. For more information, see “Specifying Output Destinations for Targets”.
3. Deploy output to output destinations. For more information, see “Deploying Output to Output Destinations”.

## Creating Output Destinations

Before you can deploy your output, you must create output destinations. You can specify one output destination or multiple output destinations. Specify multiple output destinations when you want to deploy your output to multiple locations. For example, assume that you place your generated output to a web server computer, and you use both a staging server and a production server. You can create one output destination in ePublisher for the staging server, and another output destination in ePublisher for the production server.

Output destinations are not project or target specific. When you define output destinations in ePublisher, ePublisher saves the output destinations you define and allows you to use the output destinations you specify across multiple ePublisher projects and targets.

When you deploy output to an output destination, ensure you specify a descriptive name for the output destination. When you work with output destination, you can only see the name of the output destination. You will not be able to see the actual path you specified to the output destination. Type a descriptive name for the output destination that allows you to easily identify each output destination you specify.

For example, if you are deploying WebWorks Help output for a product to both a staging server and a production server, type `Production Server ProductA WebWorks Help` for the first output destination. When you create your second output destination, type `Staging Server ProductA WebWorks Help` for the second output destination.

## To create an output destination

1. On the **Target** menu, click **Target Settings**.
2. Click **Add deploy target**.
3. Complete the following steps:
  - a. Click **Add > Folder**.
  - b. In the **Name** field, type a descriptive name for the output destination.
  - c. In the **Directory** field, type the path to the folder you want to specify as the output destination, or click the folder icon and then browse to and select the folder where you would like to deploy your output.
  - d. Click **OK**.

After you create an output destination, you must specify which target is associated with the output destination before you can deploy output. For more information, see “Specifying Output Destinations for Targets” and “Deploying Output to Output Destinations”.

## Specifying Output Destinations for Targets

After you create an output destination, you must associate the output destination with an target before you can deploy output.

### To specify an output destination for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify an output destination.
2. On the **Target** menu, click **Target Settings**.
3. In the **Deploy to** field, select an output destination.

**Note:** You must create an output destination before you can specify an output destination for a target. For more information about creating output destinations, see “Creating Output Destinations”.

4. Click **OK**.

After you specify an output destination for a target, you can generate output and deploy the output to the output destination.

## Deploying Output to Output Destinations

After you create output destinations, specify output destinations for targets and generate output, you can use the **Deploy** command in ePublisher to copy your output files and place them into the locations you specified as output destinations. When you deploy output, ePublisher copies the target output files and places the output files in the location you specified as the output destination. For more information about creating output destinations and specifying output destinations for targets, see “Creating Output Destinations” and “Specifying Output Destinations for Targets”.

## To deploy output to an output destination

1. On the **Project** menu, select the target next to **Active Target** for which you want to deploy output.
2. On the **Target** menu, click **Deploy**. ePublisher deploys the output files to the specified output location.

# Working with Target Settings

Based on your ePublisher implementation, after you create a project using Stationery, you can customize target settings for the targets available in your project if you have appropriate permissions. You can only customize target settings in a project if you have target setting modification permissions.

If you are using ePublisher Designer, you have target setting modification permissions. If you are using ePublisher Express, you may or may not have target setting modification permissions. When you install ePublisher Express, you must select the **Allow users to modify Target Settings and Properties** check box in order to have permissions to modify the target settings for the targets available in your project. If you do not select this check box during installation, you will not be able to customize target settings in projects. However, you can enable target setting modification permissions after you install ePublisher Express if needed. For more information, see "Working with Contract IDs".

***If you have permissions to modify the target settings in projects***, you can customize the following target settings for most output formats:

**Note:** If you are using ePublisher Express, any customizations you make to target settings will be overwritten the next time you synchronize your ePublisher Express project with Stationery. For more information, see "Synchronizing Projects with Stationery".

- Accessibility settings. For more information, see "Specifying Accessibility Settings".
- Baggage Files settings. For more information, see "Specifying Baggage Files Settings".
- Company information. For more information, see "Specifying Company Information".
- File processing behavior for front matter, index files, and table of contents files. For more information, see "Specifying File Processing Behavior for Front Matter, Index, and Table of Contents Files".
- When to create new pages. For more information, see "Specifying Page Breaks Settings".

- How you want to name your page files and image files when generating output. For more information, see “Specifying Page, Image, and Table File Naming Patterns”.
- Index settings. For more information, see “Specifying Index Settings”.
- How links to files or external URLs display in browser windows. For more information, see “Specifying How Links to Files or External URLs Display in Browser Windows”.
- Character encoding settings for targets. For more information, see “Specifying Character Encoding for Targets”.
- Language used by targets. For more information, see “Specifying the Language Used by Targets”.
- PDF generation settings. For more information, see “Specifying PDF Generation Settings”.
- Table of contents settings. For more information, see “Specifying Table of Contents Settings”.
- Report settings. For more information, see “Specifying Report Settings”.
- Output format-specific settings, such as settings specific to the WebWorks Help output format or the Microsoft HTML Help output format. For more information, see “Specifying Output Format-Specific Settings”.
- Variable settings. For more information, see “Setting Variables in Projects”.
- Condition settings. For more information, see “Setting Conditions in Projects”.
- Cross-reference settings. For more information, see “Setting Cross-References in Projects”.

After you make any customizations to the target settings for the targets available in your project, generate output so that you can review your changes and verify that the generated output displays and functions properly. You can generate output for all the source documents and groups in your project, or you can generate output for a single group or source document. For more information about generating output, see “Generating Output”.

## Specifying Accessibility Settings

In ePublisher, accessibility refers to how users with disabilities access electronic information and how writers and producers of online content produce accessible output that can function with assistive devices used by individuals with disabilities. Creators of online content, such as writers who product online content and help



systems and others who are responsible for producing accessible help, or Section 508 compliant content, must follow certain guidelines established by the W3C and the U.S. government. If you are responsible for producing accessible online content, you must provide alternate text and descriptions for all images and image maps and summaries for all tables included in the online content. Ensure you specify this information when you prepare your source documents for output generation. For more information, see “Creating Accessible Online Content in FrameMaker” and “Creating Accessible Online Content in Word”.

## To specify accessibility settings for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Accessibility**, specify the appropriate values for the Accessibility settings. For more information about Accessibility settings and values, click **Help**.
4. Click **OK**.

## Specifying Baggage Files Settings

In addition to your source files content inside HTML and PDF files can be indexed by enabling Baggage File settings. In ePublisher you can access to the Baggage File settings only if your target is WebWorks Reverb (Reverb 1 and 2). There are several settings that allows you to customize the way Reverb handles the baggage files.

You can specify the following settings:

- Baggage files info list
- Copy baggage file dependents
- Index baggage files
- Index external links

We will explain each setting briefly below.

### Baggage files info list

The `baggage files info list` allows you to control index behavior and assign titles and descriptions to external files and URLs linked to in source documents. This is useful if the automatically-assigned titles and descriptions that display in the output search results are not relevant or are missing.

If your baggage files or external URLs do not have adequate title and summary values defined, then you can also use the `baggage files info list` to provide these values. These values are used for displaying search results.

If you have external URL links or baggage files in your source content that you do not want included in the search index, then use the attribute `@noindex`.

When specifying the path for baggage files, you can either use an absolute path or a path relative to the `baggage files info list` file.

You can specify the location and filename of the `baggage files info list` using the Target Settings dialog (explained below). By default the filename is called: `baggage_list.xml` and is available to override in the **Advanced > Manage Format/Target Customizations** menu. In addition to renaming this file, you can also specify an absolute path or a relative path from the project file directory. Furthermore, you can use a variable for getting the directory location of the first document in the project. Using this variable called: `$FirstDocDir;`, you can locate the `baggage files info list` file in this directory. This is a useful way to use your stationary with multiple projects.

To use this variable you need to specify it first in the Target Setting value like this:

```
$FirstDocDir;/baggage_list.xml.
```

**Note:** If you change the path of the `baggage files info list` target setting, then even if you have overridden this file, the overridden file will be ignored and the file specified will be used instead. However, if you just change the base filename, then ePublisher will look for this file as if it were an override.

## To get the path of the Baggage Files info list file we follow these steps

1. **If you change the default value or just change the name (without specifying a path)** Reverb tries to get the file from the Targets folder first and then from the Formats folder. The file will be located in the Transforms folder, so you can easily do an override of it if you want to keep the default name, otherwise you can add the new file there with the name you defined in the **Target Settings**.
2. **If you define an Absolute or Relative path** Reverb will calculate the *relative path* relative to the project file and will take the *absolute path* as is.

If you don't specify a title and/or a summary for a baggage file we will try to do it for you.

In case of an HTML file for getting a title (if you didn't define one in a baggage list file) we will search for:

1. A `<title>` tag defined in the `<head>` section of the HTML.
2. The base name without the extension of the HTML file.

In case of a PDF file for getting a title (if you didn't define one in a baggage list file) we will get the base name without the extension of the PDF file.

In case of an HTML file for getting a summary (if you didn't define one in a baggage list file) we will search for:

1. The attribute `@content` in the `<meta>` tag defined in the `<head>` section of the HTML with attribute `@name="summary"`.
2. All the text inside the first tag, contained in the `<body>` tag, with attribute `@class="summary"`.
3. The attribute `@content` in the `<meta>` tag defined in the `<head>` section of the HTML with attribute `@name="description"`.
4. All the text inside the first `<p>` tag, contained in the `<body>` tag.

In case of a PDF file for getting a summary (if you didn't define one in a baggage list file) we will get the first 300 letters from the content of the PDF file.

The attribute `@noindex` accepts 2 values: `true` | `false`, or you can even not define this attribute at all, and it will take the value `false` by default. If you define `@noindex="true"` it means Reverb won't index that file.

The attribute `@path` is for specifying the path to the file (relative or absolute) or the external URL. It should be an existing path to an HTML page or PDF, either local or in the web.

**Note:** Any text you write in this file, if it contains a reserved character you'll have to change it to use the entity corresponding. For example, instead of "&" use "&".

The following code will show you how to structure a baggage list file, as well as some examples for the entries:

```
<?xml version="1.0" encoding="UTF-8" ?>

<Files version="1.0" xmlns="urn:WebWorks-Baggage-List-Schema">

<File path="http://example.com/" noindex="true"/>

<File path="https://example.com/myfavoritepage" title="My favorite
page" summary="Favorite pages can make your day better"/>

<File path="Source-Docs\some_pdf.pdf" title="Some PDF title"
summary="Some PDF summary"/>

<File path="C:\Documents\another_pdf.pdf" title="Another PDF title"
summary="Another PDF summary"/>

...

</Files>
```

## Copy baggage file dependents

If you have this setting **Enabled** in your **Target Settings**, all the dependents of your HTML baggage files will be copied to the `baggage` folder inside the corresponding group in the output folder. These mean the final user will be able to open the HTML baggage file and it will look pretty similar to the original one. Right now we support the dependences corresponding to these tags:

- `<link>` tag inside the `<head>` tag that it's not an external URL or starts with `"javascript:"`
- `<script>` tag that it's not an external URL or starts with `"javascript:"`
- `<img>` tag that it's not an external URL or starts with `"javascript:"`
- `<input>` tag that it's not an external URL or starts with `"javascript:"`
- `<iframe>` tag that it's not an external URL or starts with `"javascript:"`
- `<video>` tag that it's not an external URL or starts with `"javascript:"`
- `<audio>` tag that it's not an external URL or starts with `"javascript:"`

- `<object>` tag inside the `<body>` tag that it's not an external URL or starts with `"javascript:"`

## Index baggage files

If you have this setting **Enabled** in your **Target Settings**, Reverb will index all the baggage files allowing them to show up in Search Results. You can override this behavior on a file by file basis by specifying the attribute `@noindex="true"` in your Baggage files info list.

In order to handle most any type of HTML file Reverb uses Tidy (tool for cleaning up HTML files) for creating a well-formed XHTML temporary copy of the files, which are valid XML files that ePublisher can read.

## Index external links

If you have this setting **Enabled** in your **Target Settings**, Reverb will index all the external links you have in your source documents and in your Baggage files info list. That means you'll have in your Search Results links to your external URLs if they match with the searched phrase.

The Reverb format downloads the file to the Data directory and then uses Tidy (tool for cleaning up HTML files) for creating an XHTML copy of the files, which are valid XML files that ePublisher can read.

**Note:** If your URL needs to execute some JavaScript code to get the content of the page, Reverb won't be able to index the dynamic content of the page. To simulate the actual content that Reverb will index at a particular URL, temporarily disable JavaScript in your browser and visit that link.

### To specify baggage files settings for a target (only for Reverb targets):

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see "Working with Target Settings".
3. Under **Baggage Files**, specify the appropriate values for the Baggage Files settings. For more information about Accessibility settings and values, click **Help**.
4. Click **OK**.

## Specifying Company Information

You can add your company's contact information to each generated output page. ePublisher can display the company contact information on the bottom and/or top of your output pages. Where the company information displays depends on what the Stationery designer specified in the Stationery file.

To display the company information in the header area, see "Header Settings".

To display the company information in the footer area, see "Footer Settings".

You can specify the following company information:

- Company email address
- Company fax number
- Company logo image
- Company name
- Company phone number
- Company web page

### To specify company information for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Company Information**, specify the appropriate values for the company information settings. For more information about the company information settings and values, click **Help**.
4. Click **OK**.

## Specifying File Processing Behavior for Front Matter, Index, and Table of Contents Files

You can specify file processing behavior for front matter, index files, and table of contents files. For example, you can specify whether or not you want to generate output for front matter included in your source documents.



### To specify file processing behavior for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see "Working with Target Settings".
3. Under **File Processing**, specify the appropriate values for file processing settings. For more information about the file processing settings and values, click **Help**.
4. Click **OK**.

## Specifying Page Breaks Settings

When ePublisher processes source documents, it creates new topic pages based on settings specified by the Stationery designer in the Stationery. However, you can modify how you would like ePublisher to handle the page breaks.

### To specify page break settings for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Files**, in the **Page break handling** field, select the value you want to specify. For more information about the values, click **Help**.
4. Click **OK**.

## Specifying Page, Image, and Table File Naming Patterns

You can specify page, image, and table file naming patterns that you want ePublisher to use when generating output.

For example, you can specify if you would like to include the following items in page, image, and table file names when generating output:

- Target name
- Name of the group in Document Manager that contains the topic
- Page heading text or title

You can use image naming patterns to specify names for embedded image output files. However, if you insert your images by reference in Adobe FrameMaker or use the **Link to File** or **Insert and Link** option in the Insert Picture window in Microsoft Word, ePublisher preserves the original file names.

**Note:** You can also use Filename markers to specify page and image output file names.

### To specify page, image, and table file naming patterns for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see "Working with Target Settings".
3. Under **Files**, specify the appropriate values for the page, image, and file naming patterns you want to use. For more information about file settings and values, click **Help**.
4. Click **OK**.

## Specifying Index Settings

In ePublisher, you can specify if you want to generate an index for your help system. If you choose to generate an index for your help system, you must have index markers in your source documents.

### To specify index settings for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Index**, specify the appropriate value for each index setting. For more information about the index settings and values, click **Help**.
4. Click **OK**.

## Specifying How Links to Files or External URLs Display in Browser Windows

ePublisher allows you to specify how you want links that open baggage files or links that open external URLs displayed in your output. For the definition of a baggage file see “Targets”.

### To specify link settings for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Links**, specify the appropriate value for each links setting. For more information about the links settings and values, click **Help**.
4. Click **OK**.

## Specifying Unknown File Links Behavior in Reverb

In the output format, WebWorks Reverb 2.0, relative file links that do not resolve to an actual file (i.e. baggage file) are not active when loaded in the browser. In this situation, Reverb assumes that this link is not available in the current session and thus prevents the user from activating them and getting an unknown file error.

If you wish to preserve these types of links in your output (there are circumstances where preserving this behavior can be useful), then you will need to enable the target setting called: `Preserve Unknown File Links`.

### To enable preservation of unknown file links

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target** menu, click **Target Settings**.
3. Under **Links**, enable the setting: `Preserve Unknown File Links`.

## Specifying Character Encoding for Targets

In ePublisher, encoding refers to the character encoding method used to convert bytes into characters. Programs use encoding when they display HTML documents. Documents in English and most other Western European languages typically use the widely supported character encoding UTF-8. If you are producing output localized for other languages, such as Japanese, Korean, Simplified Chinese, Traditional Chinese, Greek, Turkish, or Eastern European, Cyrillic, or Baltic languages, you must specify the correct encoding for each target for which you generate output.

Ensure the encoding you specify when you generate your output matches the encoding used in the environment where your output will be posted. For example, if your output will be posted on a web server, the encoding you specify when you generate your output should match the encoding used on the web server. If your output and the computer or web server hosting your output do not use the same character encoding method, some characters may not display correctly when users view your output.

### To specify character encoding for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see "Working with Target Settings".
3. Under **Locale**, specify the appropriate value for the **Encoding** setting. For more information about the encoding setting values, click **Help**.
4. Click **OK**.

## Specifying the Language Used by Targets

In ePublisher, locale refers to the language used when displaying output for a target. If you produce localized output, specify the correct language for each target in your ePublisher project.

### To specify the language to use for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Locale**, specify the appropriate value for the **Locale** setting. For more information about the locale setting values, click **Help**.
4. Click **OK**.

## Specifying PDF Generation Settings

ePublisher can generate PDFs for each source document, for each top-level group in your project, or for each source document and each top-level group in your project.



## To specify PDF generation settings for a target

1. On the **Project** menu, select the output format next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **PDF**, specify the appropriate values for the PDF settings. For more information about PDF settings and values, click **Help**.
4. Click **OK**.

## Specifying Table of Contents Settings

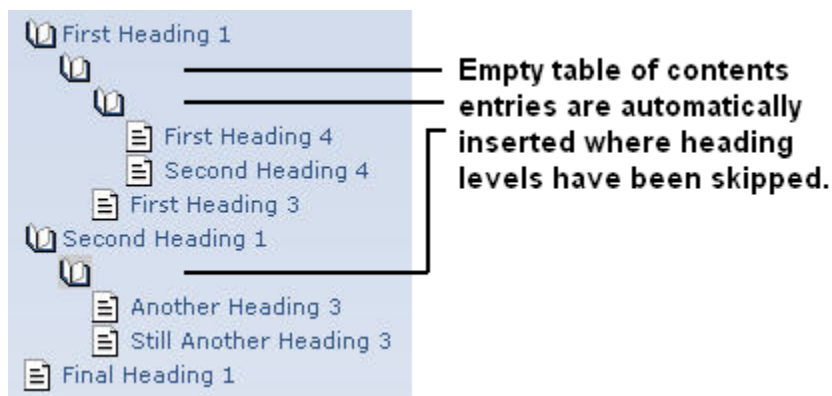
ePublisher allows you to specify whether you want to generate a table of contents, the file name you want to specify for your table of contents file, and how you want your table of contents to display in your generated output.

ePublisher provides table of contents settings to help you address how you want your table of contents to display. By default, ePublisher uses the table of contents levels specified in the project or in the Stationery file to create a table of contents for your help system based on the heading levels in your source documents. However, if you have source documents where writers skipped heading levels, you can specify how you want ePublisher to display skipped headings in the table of contents.

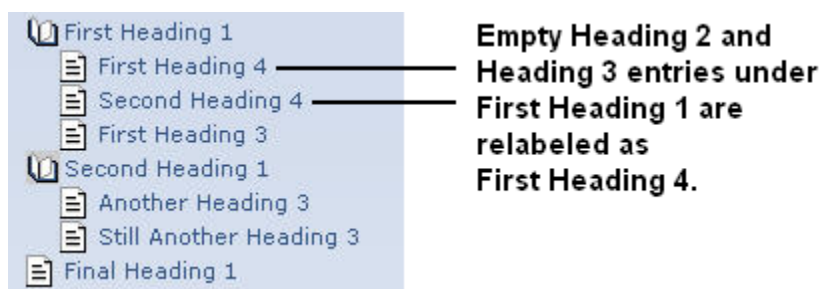
For example, assume that you have an ePublisher project that uses a Stationery file that specifies Heading 1, Heading 2, and Heading 3 as levels in the output table of contents. Then assume that in the source document, you skipped several Heading 2 levels. ePublisher displays an empty table of contents icon, similar to the following figure, in the location of the skipped Heading 2 levels unless you specify how you want to manage skipped heading levels in the generated table of contents.

You can specify the following behavior for table of contents where writers skipped headings:

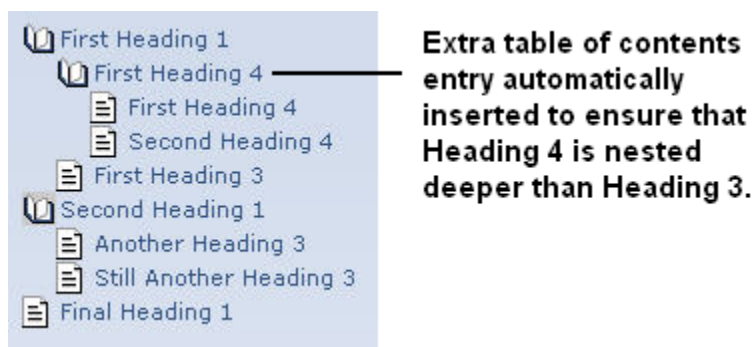
- ***If you want ePublisher to automatically insert empty table of contents entries for skipped heading levels***, select the **Don't collapse** value. The following figure shows a table of contents with this option selected.



- ***If you want ePublisher to automatically insert labeled entries for skipped heading levels***, select the **Re-label** value. ePublisher displays the heading text from the table of contents entry below the current entry as the table of contents label. The following figure shows a table of contents with this option selected.



- ***If you want ePublisher to automatically remove empty table of contents entries and move the heading that follows an empty table of contents entry up a level to replace the skipped table of contents level***, select the **Smart collapse** value. The following figure shows a table of content with this option selected.



- ***If you want ePublisher to remove all skipped heading levels and table of contents entries and place all table of contents headings at the same level, regardless of the table of contents level specified in the***

**Stationery**, select the **Fully collapse** value. The following figure shows a table of contents with this option selected.



First Heading 1
First Heading 4
Second Heading 4
First Heading 3
Second Heading 1
Another Heading 3
Still Another Heading 3
Final Heading 1

**Table of contents is fully collapsed, which allows the Heading 4 and Heading 3 to appear at the same level.**

## To specify table of contents settings for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Table of contents**, specify the appropriate values for the table of contents settings. For more information about table of contents settings and values, click **Help**.
4. Click **OK**.

## Specifying Report Settings

You can use reports to identify problems in your source documents. If an ePublisher report detects problems in your source document, ePublisher displays a notification alert in the report. You can specify which types of settings you want to use to validate your generated output and the type of notification you want to receive if ePublisher detects an issue when validating your output. For more information about using reports to validate your output and the different types of notifications you can receive, see “Validating Output Using Reports”.

## Specifying Output Format-Specific Settings

You can specify output format-specific settings for the following output formats:

- WebWorks Reverb
- PDF - XSL-FO
- eBook - ePUB 2.0
- Eclipse Help
- Microsoft HTML Help
- Oracle Help
- PDF
- Sun JavaHelp
- WebWorks Help

You must have the target that uses the output format selected in your project before you can see the output format-specific settings in the window. For example, to see WebWorks Help output format-specific settings in the window, you must have a target that uses the WebWorks Help output format selected as your active target. If you have a target that uses the Microsoft HTML Help output format selected as your active target, you will not be able to see WebWorks Help output format-specific settings in the window. You will only be able to see Microsoft HTML Help output format-specific settings.

### To specify output format-specific settings for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify output format-specific settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under the name of the output format, specify the appropriate values for each output format-specific setting. For more information about output format-specific settings and values, click **Help**.
4. Click **OK**.

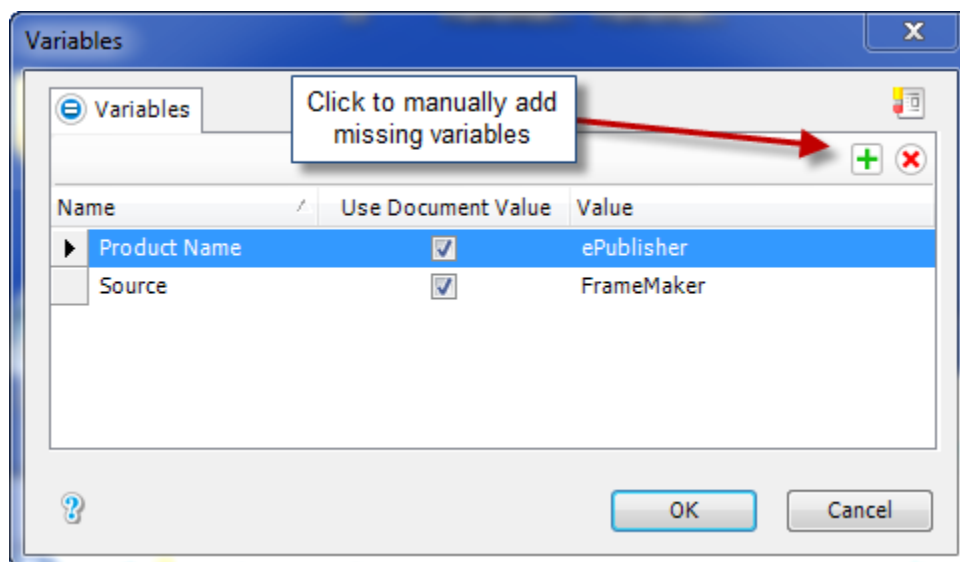
## Setting Variables in Projects

In a project, you can use the variable values defined in your source document. You can also change the value of any variable in your source document in a project. Changing the value of a variable in a project does not change or affect the value of the variable in your source document. You can use the value of the variable you defined in your project when you generate output. Before you can work with variables in projects, you must insert variables in your source documents.

## To set a variable in a project

1. On the **Project** menu, select the target next to **Active Target** for which you want to customize variable settings.
2. On the **Target** menu, click **Variables**. You must have target modification permissions to modify variable settings for a target. For more information, see "Working with Target Settings".

**Note:** For Microsoft Word documents only, if you use variables that are built-in `DocProperty` types such as `Author` or `Company`, then you will need to manually add these variables into the project or stationery as they are not detected when scanned by ePublisher Designer. However, once added into either the stationery or project then they will be available for customization from that point forward.



3. In the **Name** column, find the variable you want to modify.
4. **If you want the your ePublisher project to use the variable value defined in your source document**, click in the **Value** field for the variable, and then select **Use document value** from the drop-down list.
5. **If you want to change the variable value ePublisher uses when generating output**, click in the **Value** field for the variable, and then type in a new value for the variable.
6. Click **OK**.
7. Generate your output. For more information, see "Generating Output".

8. Review your output and confirm that variables display appropriately in your generated output. For more information, see “Viewing Output”.

## Setting Conditions in Projects

In a project, you can use the conditions defined in your source document to control the visibility of content to which you have applied conditions. You can also change the visibility specified for any condition in a project. Changing the visibility specified for any condition in a project does not change the visibility specified for the condition in your source documents. Before you can work with conditions in projects, you must apply conditions to content in your source documents.



## To customize a condition in a project

1. On the **Project** menu, select the target next to **Active Target** for which you want to customize condition settings.
2. On the **Target** menu, click **Conditions**. You must have target modification permissions to modify condition settings for a target. For more information, see “Working with Target Settings”.
3. In the **Name** column, find the condition for which you want condition to set the value.
4. Specify the appropriate value for the condition. For more information about condition values, click **Help**.
5. Click **OK**.
6. Generate your output. For more information, see “Generating Output”.
7. Review your output and confirm that conditionalized content displays appropriately in your generated output. For more information, see “Viewing Output”.

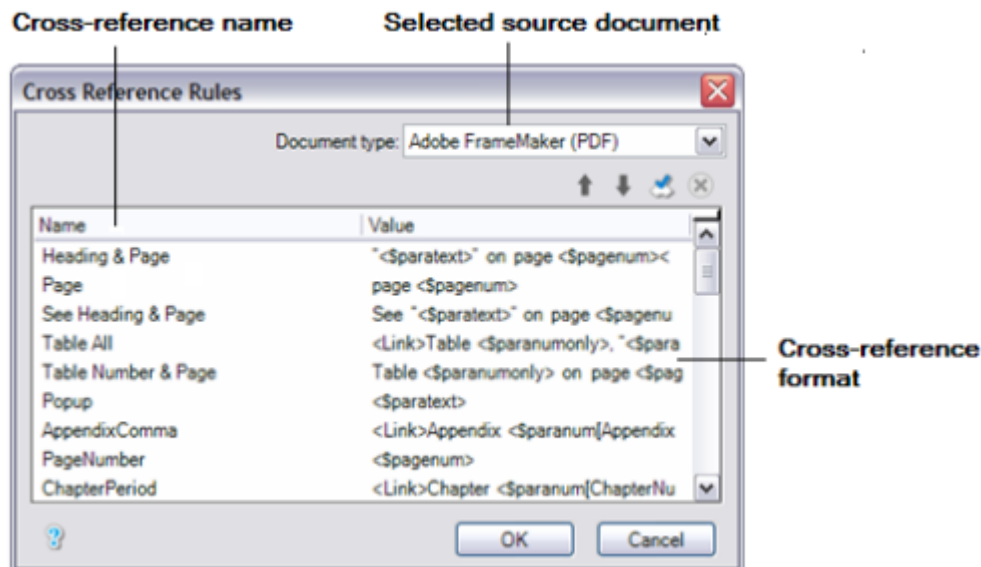
## Setting Cross-References in Projects

Cross-references help users access related information quickly in printed and online content. When you convert your source documents to online help, if you have cross-references in your source documents, ePublisher automatically converts all cross-references to hypertext links. Typically, cross-references used for printed materials have a different format than cross-references used for online help. For example, cross-references in printed content typically include page numbers, while cross-references in online help typically do not include page numbers, because page numbers are out of context in online help.

In ePublisher, you use the Cross-Reference Rules window to add, edit, or delete cross-reference formats for your project. A cross-reference format is a combination of text and code that defines how you want your cross-reference to display. For example, your source documents may display the following cross-reference format: “Modifying Cross-Reference Formats on page xxx”, where xxx is the page number where the topic “Modifying Cross-Reference Formats” begins. However, you may modify the cross-reference format in ePublisher so that when you generate online content, the “Modifying Cross-Reference Formats” topic displays as a hyperlink without a page number, such as [Modifying Cross-Reference Formats in Projects](#).

ePublisher obtains the cross-reference formats and values in the Cross-Reference Rules window from your source documents. You can modify cross-reference formats in ePublisher. For more information about cross-reference building blocks or codes, see your content authoring tool documentation.

The following figure shows the Cross-Reference Rules window in ePublisher.



## Modifying Cross-Reference Formats in Projects

Modify cross-reference formats when you want cross-references in your online content to use a different format than your printed content.

## To modify a cross-reference format in a project

1. On the **Project** menu, select the target next to **Active Target** for which you want to modify cross-reference formats.
2. On the **Target menu**, click **Cross Reference Rules**. You must have target modification permissions to modify a cross-reference format for a target. For more information, see "Working with Target Settings".
3. Specify the appropriate value for each cross reference. For more information about cross reference values, click **Help**.
4. Click **OK**.
5. Generate your output. For more information, see "Generating Output".
6. Review your output and confirm that cross-references display appropriately in your generated output. For more information, see "Viewing Output".

## Adding Cross-Reference Formats to Projects

If you are using Adobe FrameMaker or Microsoft Word documents as your source documents for ePublisher, then you can take advantage of the Cross-Reference Formats capability in ePublisher to modify cross references when generating output.

ePublisher obtains the cross-reference formats and values in the Cross-Reference Rules window from your source documents. You can also add cross-reference formats in ePublisher.

For example, if you started to use a new cross-reference format in your source document and the Stationery designer has not yet added this new cross-reference format to the Stationery associated with your project, you can add the new cross-reference format to your project and specify the cross-reference format you want to use for your new cross-reference format. After you add a new cross reference format ePublisher recognizes the new cross reference formats and applies the cross-reference format you specify.

## To add a cross-reference format to a project

1. On the **Project** menu, select the target next to **Active Target** for which you want to add a cross-reference format.
2. On the **Target menu**, click **Cross Reference Rules**. You must have target modification permissions to add a cross-reference format for a target. For more information, see "Working with Target Settings".
3. In the **Document type** field, select the content authoring tool for the cross-reference format you want to add.
4. Click the **Add New Cross Reference** icon.
5. In the **Name** field, type a name for the new cross-reference format you want to add to the project.
6. In the **Replacement** field, type a combination of text and code or building blocks that define how you want your new cross-reference to display. For more information about cross-reference building blocks or codes, see your content authoring tool Help.
7. Click **OK**.
8. Click **OK** again to close the window.
9. Generate your output. For more information, see "Generating Output".
10. Review your output and confirm that cross-references display appropriately in your generated output. For more information, see "Viewing Output".

## Deleting Cross-Reference Formats from Projects

ePublisher obtains the cross-reference formats and values in the Cross-Reference Rules window from your source documents. You can delete cross-reference formats in ePublisher. Delete cross-reference formats when you no longer want to use the cross-reference format in your source documents.

If you delete the cross-reference format in your ePublisher project, but your source documents continue to use the cross-reference format, ePublisher will detect the deleted cross-reference format in your source documents and add it to your project again the next time you scan your source documents or generate output.

## To delete a cross-reference format from a project

1. On the **Project** menu, select the target next to **Active Target** for which you want to delete a cross-reference format.
2. On the **Target menu**, click **Cross Reference Rules**. You must have target modification permissions to delete a cross-reference format for a target. For more information, see “Working with Target Settings”.
3. In the **Document type** field, select the content authoring tool associated with the cross-reference format you want to delete.
4. In the **Name** column, select the cross-reference format you want to delete.
5. Click the **Delete Cross Reference** icon.
6. Click **OK**.

# File Mappings for Source Documents

This section explains how to configure file mappings for source document types in a project.

## File Mappings

In ePublisher, a file mapping is an association between a file extension and an ePublisher adapter. An ePublisher adapter is an ePublisher component that links the content authoring tool that you used to develop your content with ePublisher. ePublisher currently provides adapters for the following content authoring tools:

- Markdown++ (helper)
- Adobe FrameMaker
- Microsoft Word
- XML

In ePublisher, you can add any source documents that can be opened with Adobe FrameMaker, Microsoft Word, DITA-XML to your ePublisher project through the use of file mappings. By default, ePublisher provides a list of file extensions that are preset to use either Microsoft Word, Adobe FrameMaker, or the built-in XML adapter. For example, you can add `.txt` files to your ePublisher project by specifying the adapter ePublisher should use in order to open the `.txt` file. You can specify whether you want the Adobe FrameMaker, Microsoft Word, or XML adapter to open the `.txt` files you add to your project.

Certain file extensions, such as `.book`, `.fm`, and `.bk` files, are unique to a specific adapter. For example, `.book`, `.fm`, and `.bk` file can only be opened by Adobe FrameMaker. `.rtf`, `.xml`, and `.doc` are specific to Microsoft Word. If you try to generate output or an output preview using a file type associated with an ePublisher adapter and the file type cannot normally be opened with the content authoring tool associated with the ePublisher adapter, ePublisher displays an error message. The built-in XML adapter ePublisher provides is configured out-of-the-box to support DITA-XML. You can also configure ePublisher Stationery to support other XML types. However, XML input formats other than DITA-XML may not be supported by the WebWorks Technical Support team.

If you have an ePublisher Contract ID that enables only the Microsoft Word, the Adobe FrameMaker, or the built-in XML adapter, then you can use only that adapter when you use ePublisher. Although the option to choose another adapter may be available in the ePublisher user interface, you will not be able to generate output or preview output using the other adapters. You can only use the adapters enabled by your Contract ID.

## Modifying File Mappings

ePublisher provides a default list of file mappings in which file extensions have been preset to use a specific adapter. However, in some cases you may need to modify file mappings for a project. For file extensions that can either be opened with Microsoft Word or Adobe FrameMaker, such as `.txt` files, you can specify the adapter you want ePublisher to use for the file extension. You can modify file mappings for a specific project or for all of your ePublisher projects.

## To modify a file mapping

1. ***If you want to modify a file mapping for a specific project***, complete the following steps:
  - a. On the **Project** menu, click **Project Settings**.
  - b. In the **File Extension** column, click the file extension for which you want to modify the file mapping.
2. ***If you want to modify a file mapping for all of your ePubisher projects***, complete the following steps:
  - a. On the **Edit** menu, click **Preferences**.
  - b. On the **File Mappings** tab, in the **File Extension** column, click the file extension for which you want to modify the file mapping.
3. In the **Adapter** column, select the ePubisher adapter you want to associate with the file extension. The ePubisher adapter you associate with the file extension will be the ePubisher adapter that opens files with the specified file extension.
4. Click **OK**.
5. Click **OK** again. Each new ePubisher project you create after you modify the file mapping will use the ePubisher adapter you associated with the file extension.

## Creating New File Mappings

If there is a file extension that you would like to use but the file extension is not available in ePubisher in the default list of file extensions, you can create a new file mapping. To create a new file mapping, add a new file extension and associate, or map, the file extension to an ePubisher adapter. You can use the new, or custom, file mapping to specify that ePubisher open files using the new file extension with Adobe FrameMaker, Microsoft Word, or the built-in XML adapter. When you create a new file mapping, ePubisher saves information about the new file mapping you created, and you can apply the new file mapping to all of the subsequent projects that you open.

When you create a file mapping and specify an ePubisher adapter for the file extension, ensure the file extension can be opened using the content authoring tool associated with the adapter outside of ePubisher before you create the new file mapping. If the file extension cannot be normally opened using the content authoring tool, then ePubisher will also not be able to generate output from the source document using the ePubisher adapter.

For example, assume your Contract ID enables licensing for ePublisher Express for FrameMaker. Next assume that you add HTML as a file mapping and associate the `.html` file extension with the Microsoft Word adapter. When you create the file mapping for the `.html` file extension with the Microsoft Word adapter, ePublisher allows you to add the HTML file to your project. However, since you do not have a valid license key for the ePublisher Express for Microsoft Word, ePublisher displays the following error message.





## To create a new file mapping

1. ***If you want to create a new file mapping for a specific project,*** complete the following steps:
  - a. On the **Project** menu, click **Project Settings**.
  - b. Click the **Add** icon.
2. ***If you want to create a new file mapping for all of your ePublisher projects,*** complete the following steps:
  - a. On the **Edit** menu, click **Preferences**.
  - b. On the **File Mappings** tab, click the **Add** icon.
3. In the **File extension** field, type the file extension you want to use for the file mapping. For example, you can add `.html` as a file extension.
4. In the **Adapter** field, select the ePublisher adapter you want to associate with the file extension. The ePublisher adapter you associate with the file extension will be the ePublisher adapter that opens files with the specified file extension. For example, you can select Microsoft Word as the adapter for the `.html` file extension.
5. Click **OK**.
6. Click **OK** again. Each new ePublisher project you create after you modify the file mapping will use the ePublisher adapter you associated with the file extension.

## Deleting File Mappings

Delete a file mapping when you no longer want to use the file mapping in your ePublisher project.

## To delete a file mapping

1. ***If you want to delete a file mapping for a specific project***, complete the following steps:
  - a. On the **Project** menu, click **Project Settings**.
  - b. In the **File extension** field, select the file extension for the file mapping you want to delete.
2. ***If you want to delete a file mapping for all of your ePublisher projects***, complete the following steps:
  - a. On the **Edit** menu, click **Preferences**.
  - b. On the **File Mappings** tab, in the **File extension** field, select the file extension for the file mapping you want to delete.
3. Click the **Delete** icon.
4. Click **OK**.
5. Click **OK** again. Each new ePublisher project you create after you delete the file mapping will not use the ePublisher adapter you deleted.

# Scheduling and Integrating Processes with AutoMap

How ePublisher Supports Automation  
Preparing Projects, Stationery, and Source Files  
Working with Jobs  
Using Scripts for Additional Custom Processing  
Using the Command-Line Interface

This section describes how to automate output generation and integrate this task with your other processes. For example, you can have the build process for a software product automatically build all the targets you define in your project using the latest version of source documents checked into your version control system or content management system.

## How ePublisher Supports Automation

ePublisher provides several components to help you process source documents and publish the generated output. The ePublisher AutoMap component helps you schedule and perform all your processing tasks, and it also provides automated pre- and post-processing capabilities to complete your production and publication processes.

## What Is ePublisher AutoMap?

ePublisher AutoMap is the automation tool that enables you to automate the content transformation process, batch processing, and integration with content management or version control systems. This component lets you schedule ePublisher projects. For example, you can schedule the output generation to occur overnight. Then, when you arrive the next morning, your transformed content is ready for you. You can also automatically generate and deploy deliverables to meet your specific needs, such as updating Web site content based on updated source documents. You can automatically create ePublisher projects and generate output without manually opening ePublisher or your source documents.

## Benefits of Using ePublisher AutoMap

ePublisher AutoMap automates the process of transforming your source documents to your output formats. This component lets you transform content at scheduled times and work seamlessly with your content management and version control systems. The following list highlights several ePublisher AutoMap features that save you time and effort:

- Automates the output generation using existing projects, including synchronizing with the Stationery

- Creates projects using the specified Stationery and applies it to your content
- Creates merged output from multiple books
- Allows you to customize conditions, variables, and cross references on a per-job and per-target basis
- Redirects and deploys output automatically
- Provides a full-featured command-line interface for performing batch transformations from other systems or scripts
- Integrates with content management and version control systems
- Offers flexible scheduling options
- Notifies relevant people when a job succeeds or fails
- Automatically updates your online content, help, or Web-based information

## **Version Control System (VCS) Integration**

ePublisher AutoMap allows you to specify scripts for retrieving files from version control systems such as:

- CVS
- ClearCase
- Visual Source Safe
- Subversion
- Mercurial
- Documentum
- Perforce
- Git

ePublisher AutoMap can work with any version control system that is scriptable from the command line. For more information, see “CVS Version Control Checkout Example”. To see more version control integration scripting examples see [wiki.webworks.com/HelpCenter/Tips/VersionControl](http://wiki.webworks.com/HelpCenter/Tips/VersionControl)

## **Content Management System (CMS) Integration**

Using the same scripting integration as used for version control system integration, ePublisher AutoMap can also work with many industry standard content management systems such as: Vasant CMS and SDL LiveContent.

## **Preparing Projects, Stationery, and Source Files**

ePublisher AutoMap can transform your source documents with no special modifications. Prepare your project, Stationery, and source documents just like you do when you generate output with the other ePublisher components. In ePublisher AutoMap, you use a job to define the output generation and the applicable options. After you create your job, ePublisher AutoMap performs one of the following tasks:

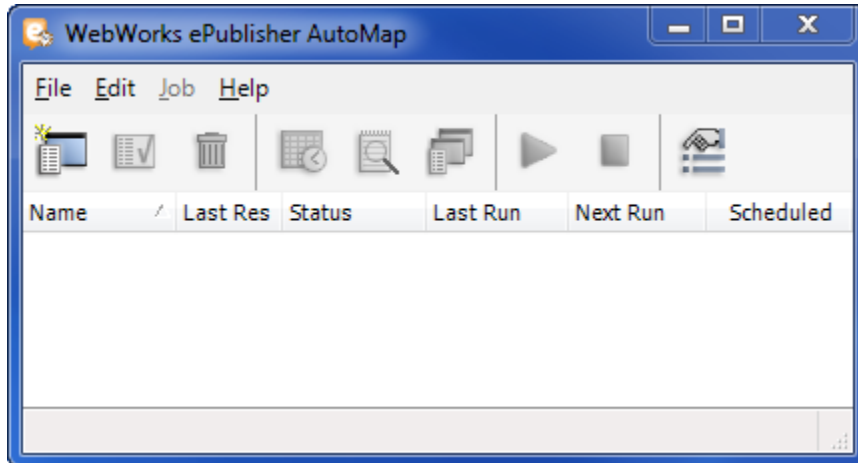
- If you created the job based on a project, ePublisher AutoMap synchronizes the project with the Stationery, and then it generates the output defined by the project.
- If you created the job based on a Stationery, ePublisher AutoMap creates a new project based on the Stationery, adds all the source documents defined by the job, and then it generates the output defined by the project.

## **Starting ePublisher AutoMap**

ePublisher AutoMap features an easy-to-use console that allows you to schedule output generation using your source documents to run either immediately or at a specified time.

## To start ePublisher AutoMap

In the **WebWorks** program group, click **ePublisher AutoMap > WebWorks ePublisher AutoMap**.



## To start ePublisher AutoMap without administrative privileges

If are running in a restrictive environment that does not allow elevation of user permissions to Administrator rights, then you will need to run ePublisher AutoMap from the following batch file. This file lives in the installation directory underneath `WebWorks\ePublisher\<version>\ePublisher AutoMap\`. The file is called `WebWorks.Automap.Administrator.restricted.bat`.

**Note:** You will not be able to launch or schedule jobs when starting ePublisher AutoMap this way. However you will be able to create and configure jobs.

## Setting ePublisher AutoMap Preferences

The Preferences window lets you specify default options and preferences that affect the behavior of ePublisher AutoMap. These preferences allow you to customize the console for your specific needs.

## Specifying the Job, Staging, and User Formats Folder Locations

ePublisher AutoMap has several folders where it stores job-related information, such as job files and log files. You can customize the location of these folders.

### ***Job Folder***

ePublisher AutoMap stores the following types of job-related files in the job folder:

#### **log file**

A text file that contains information about the last transformation process. ePublisher AutoMap creates the log file when you run the job. You can view the log file with any text editor.

#### **job file**

A proprietary XML formatted file that contains information describing the job. When you create a job, ePublisher AutoMap creates the job file and stores the information you specify in the console. The job file name is the name of the job with a `.waj` (WebWorks AutoMap Job) file extension.

**Note:** ePublisher AutoMap maintains the job files. Do not edit these files.

The default job folder is the `My Documents\ePublisher AutoMap\Jobs` folder.

### ***Staging Folder***

ePublisher AutoMap stores information needed for transforming the source documents in the staging folder. This information includes the automatically

generated ePubliher project, intermediate data files, and the output files. The staging folder provides a working folder for processing each job. ePubliher AutoMap then writes the final output files to the target output destination specified when the project was created.

**Note:** ePubliher AutoMap creates the files in the staging folder. Do not edit these files.

The default staging folder is the `My Documents\epublisher AutoMap\Staging` folder.

## ***User Formats Folder***

When ePubliher AutoMap processes a job, it searches the user formats folder for custom user formats. The default user formats folder is the `My Documents\epublisher Designer User Formats` folder.



## To specify the job, staging, and user formats folder locations

1. Start ePublisher AutoMap.
2. On the **Edit** menu, click **Preferences**.
3. Click the **General** tab.
4. In the **Job Folder** field, type the path to the folder or network share where you want ePublisher AutoMap to store the job and log files. You can click **Browse** and navigate to the folder or network share.
5. In the **Staging Folder** field, type the path to the folder or network share where you want ePublisher AutoMap to store the job projects and output created during your transformations. You can click **Browse** and navigate to the folder or network share.
6. In the **User Formats Folder** field, type the path to the folder or network share where ePublisher AutoMap checks for the custom user formats to use for creating jobs. You can click **Browse** and navigate to the folder or network share.
7. Click **OK** to save your preferences.

## Automatic Scanning for Conditions and Variables

When ePublisher AutoMap displays windows that list conditions and variables, you can specify whether ePublisher AutoMap automatically scans the source documents for updated conditions and variables. Since your source documents may not have new conditions or variables, ePublisher AutoMap attempts to save time by default and it does not automatically scan for new conditions and variables. You can click **Scan Documents** on a window to have ePublisher AutoMap scan your source documents and update the window contents.

If you want ePublisher AutoMap to always display updated condition and variable values, select the **Always scan for variables and conditions** option. Depending on the number and size of files included in the job, each scan may require several minutes before ePublisher AutoMap can display the conditions and variables windows.

### To enable automatic scanning

1. Start ePublisher AutoMap.
2. On the **Edit** menu, click **Preferences**.
3. On the **General** tab, select **Always scan for variables and conditions**.
4. Click **OK** to save your preferences.

## Keeping or Deleting Temporary Files

You can choose whether ePublisher AutoMap keeps or deletes the temporary files created in the staging folder when it runs a job. By default, ePublisher AutoMap does not delete these files. You need these files only if you want to examine the actual project and intermediate files created and used during the transformation process. If you do not want to examine these files, you can disable allow ePublisher AutoMap to delete the files and reduce the amount of disk space used.

## To delete temporary files

1. Start ePublisher AutoMap.
2. On the **Edit** menu, click **Preferences**.
3. On the **General** tab, select **Delete temporary files after generating**.
4. Click **OK** to save your settings.

## Defining File Mappings

ePublisher allows you to map files based on their file extension to a source document adapter that processes that input format. This capability provides the flexibility you need:

- You can define which source application processes a file that can be handled by multiple source applications. For example, both Adobe FrameMaker and Microsoft Word can open text files with a `.txt` extension. The File Mappings tab allows you to choose which application to use for each file based on the file extension.
- You can define custom file extensions. For example, you may choose to use a non-standard file extension for your Microsoft Word source documents, such as `.word` instead of `.doc`. The File Mappings tab allows you to add your custom file extension and map it to the appropriate source application adapter.
- You can add new input formats as they become available. ePublisher provides a powerful framework in which Quadralay and their partners can develop new input formats, such as custom XML formats. Since XML is a common format, the source files are usually given unique and descriptive file extensions. The File Mappings window ensures that your projects are ready to handle any input sources.

**Note:** ePublisher AutoMap lists only the source application adapters installed with the product. Your Contract ID specifies the adapters for which you are licensed. adapter you want to use.

## To modify your file mappings

1. Start ePublisher AutoMap.
2. On the **Edit** menu, click **Preferences**.
3. Click the **File Mappings** tab.
4. ***If you want to add a new file mapping***, complete the following steps:
  - a. Click **Add**.
  - b. In **File extension**, type a file extension to be mapped.
  - c. In **Adapter**, select one of the installed source adapters, and then click **OK**.
5. ***If you want to modify an existing file mapping***, in the **Adapter** column next to the file extension you want to modify, select the installed source adapter you want to use for files with that file extension.
6. ***If you want to delete an existing file mapping***, complete the following steps:
  - a. In the **File Extension** column, select the file extension you want to delete.
  - b. Click **Delete**.
7. Click **OK** to save your preferences.

## Defining Output Destinations

ePublisher AutoMap allows you to create and manage output locations independent of your jobs. This flexibility allows you to define your output locations and then select the one you want to use when you create a new job. You can also create a new output location as the final part of the job creation process. ePublisher AutoMap allows you to deploy the final output files to a folder on a local or shared file system, such as `C:\helpdocs` or `\\server\share`.

## To define output destinations:

1. ***If you use the ePublisher AutoMap user interface to schedule an ePublisher Express project***, you must set the deployment destination in the ePublisher Express project. For more information, see “Creating Output Destinations”.
2. ***If you use the ePublisher AutoMap CLI to run an ePublisher Express project***, you can set the deployment destination in the ePublisher Express project or in the CLI. For more information, see “Creating Output Destinations” and “Using the Command-Line Interface”.
3. ***If you use ePublisher AutoMap to schedule a Stationery project***, you must set the deployment destination in the job settings, and you must set the deployment destination independently for each target. Specify the deployment destination in the job settings by completing the following steps:
  - a. Start ePublisher AutoMap.
  - b. Select the job to edit in the ePublisher AutoMap main window.
  - c. On the **Job** menu, click **Edit**.
  - d. Select the Target Configuration tab.
  - e. In the left pane, select the target for which you want to set the output destination.
  - f. On the Info tab, in the **Deployment** area, specify the deployment destination information.
  - g. Click **OK**.
4. ***If you use the CLI to run a Stationery project***, you can set the deployment destination in the ePublisher AutoMap job settings or in the CLI. Set the deployment destination in the job settings by completing the following steps:
  - a. Start ePublisher AutoMap.
  - b. Select the job to edit in the ePublisher AutoMap main window.
  - c. On the **Job** menu, click **Edit**.
  - d. Select the Target Configuration tab.
  - e. In the left pane, select the target for which you want to set the deployment destination.

- f.** On the Info tab, in the **Deployment** area, specify the deployment destination information.
- g.** Click **OK**.

For more information about setting the deployment destination in the CLI, see “Using the Command-Line Interface”.

## Defining Email Notifications

ePublisher AutoMap can send an email notification after running a job. The notification provides information about whether the job ran successfully or encountered any errors. You can also specify whether to include the log file as a text file attachment to the email.

## To configure email notification

1. Start ePublisher AutoMap.
2. On the **Edit** menu, click **Preferences**.
3. Click the **Notification** tab.
4. Select **Enable Email Notification**.
5. Specify the email addresses and email server information. For more information about a field, click **Help**.
6. Click **OK** to save your preferences.

**Note:** You can quickly enable or disable email notification without modifying the other preferences by toggling the **Enable email notification** check box.

## Selecting Console Language (English, German, French, and Japanese)

In addition to the English language, the ePublisher consoles have been translated into German, French, and Japanese languages. When ePublisher starts, it detects the operating system locale and displays the corresponding ePublisher console. You can display the console for a locale that is not the operating system default. For example, you can display the German console on an English version of Windows.

## To select the language for the console

1. Start your ePublisher console.
2. On the **Edit** menu, click **Preferences**.
3. In the **User interface language** field on the **General** tab, select the language you want the console to display.
4. Click **OK** to save your preferences.
5. Close and restart the console to view the console in the selected language.

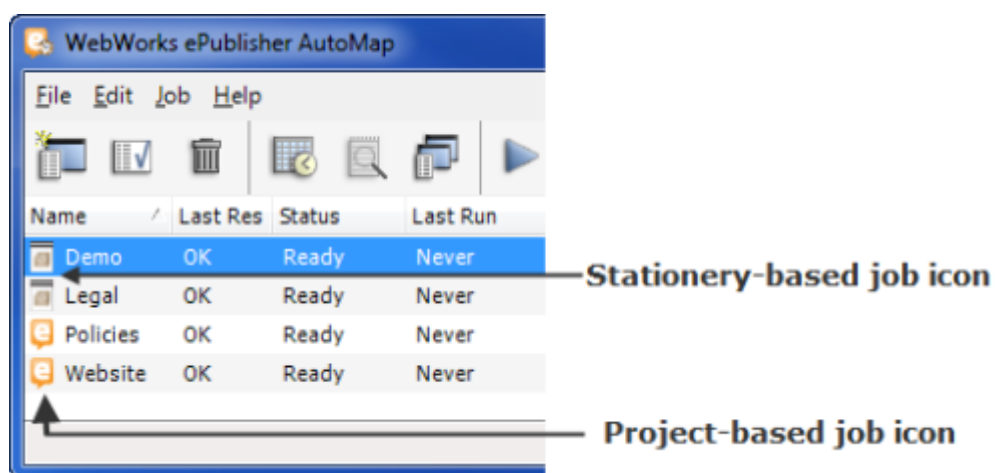
## Working with Jobs

This section describes how to create and manage ePublisher AutoMap jobs. A job is a set of tasks that ePublisher AutoMap can perform based on a Stationery or a project. You can immediately run a job, or you can schedule a job to run at a later time. By default, job files are stored in the following folder:

My Documents\WebWorks ePublisher AutoMap\Jobs

You can change the default location in the ePublisher AutoMap preferences. Job files have a `.waj` file extension. Job files depend on other files located in the job folder and should not be moved or copied independently.

ePublisher AutoMap displays a unique icon to help you identify project-based jobs and Stationery-based jobs. Each job name is preceded by the corresponding icon.



After you schedule a job in the ePublisher AutoMap user interface, you can close the ePublisher AutoMap user interface, and the job will still run based on the schedule you specified.



When ePublisher AutoMap runs a job either from a schedule or from a command prompt, Windows opens a command prompt that displays the log as it is generated. When the job finishes, the command prompt closes. After the job runs, you can view the log file using one of the following methods:

- ***If you use ePublisher AutoMap to schedule an Express or Stationery project***, the log is stored in the `Job` folder. You can view the log by clicking **View log** on the **Job** menu in the ePublisher AutoMap user interface.
- ***If you use the CLI to run an Express project***, there is no entry in the ePublisher AutoMap user interface. You can find the log in the Express project directory and view it in Notepad.
- ***If you use the CLI to run a stationery project***, the log is stored in the `Job` folder. You can view the log by clicking **View log** on the **Job** menu in the ePublisher AutoMap user interface.

**Note:** The ePublisher AutoMap user interface displays the last time that the Windows Scheduler ran the stationery project. It does not display the time the CLI ran.

## Creating a Project-Based Job

Project-based jobs allow you to schedule ePublisher AutoMap to generate the output defined by an existing ePublisher project. You first create the ePublisher project as needed, or you can use an existing project. Then, you create a project-based job so ePublisher AutoMap can use the project to generate the output defined by the project.

## To create a project-based job

1. Start ePublisher AutoMap.
2. On the **File** menu, click **New Job**.
3. Click **ePublisher project** on the New Job window.
4. Type the path or click the **Browse Folder** icon to select the ePublisher project you want to schedule, and then click **OK**.
5. In the **Job Name** field, type the desired job name.

**Note:** The **Choose ePublisher project** field displays the previously selected ePublisher project. You can change the selected project.

6. ***If you want to run a pre- or post-build script before or after the job runs***, complete the following steps:
  - a. Click the **Edit Script** button for the **Pre-build** or **Post-build** script field.
  - b. Type or paste your script into the editor, and then click **OK**. For more information about scripts, see “Using Scripts for Additional Custom Processing”.
7. Click **Next**.

ePublisher displays the Target Selection window. This window lets you select which targets to generate as part of this ePublisher AutoMap job. Depending on your project, there may be multiple targets listed in the Target Selection window.

8. Select the check box in the **Build** column next to each target you want to generate, and then click **Finish**.
9. Schedule the job as needed, and then click **OK**. You can also schedule the job at a later time. For more information, see “Scheduling Jobs with Windows Scheduler”.

Project-based jobs deploy output only if a deployment target is set in the project itself. This deployment information is specified in the target settings in ePublisher. ePublisher AutoMap reads only the deployment target name from the project. Therefore, you must make sure an output destination with the same name and is defined in ePublisher AutoMap or you may receive a deployment error. For more information, see “Defining Output Destinations”.

## Creating a Stationery-Based Job

Stationery-based jobs provide a powerful solution that can save your time and increase productivity. These jobs can also reduce errors when generating online and print content. Stationery-based jobs allow you to associate your source documents with your ePublisher Stationery. This combination lets you apply a single Stationery to many different source documents, reusing the transformation defined by the Stationery. This method ensures accuracy and consistency across many different projects and output files for your organization.

## To create a Stationery-based job

1. Start ePublisher AutoMap.
2. On the **File** menu, click **New Job**.
3. Select **ePublisher Stationery** on the New Job window.
4. Type the path or click the **Browse Folder** icon to select the Stationery upon which you want to base this job, and then click **OK**.
5. In the **Job Name** field, type the desired job name.

**Note:** The **Choose ePublisher Stationery** field displays the previously selected Stationery. You can change the selected Stationery.

6. ***If you want to run a pre- or post-build script before or after the job runs***, complete the following steps:
  - a. Click the **Edit Script** button for the **Pre-build** or **Post-build** script field.
  - b. Type or paste your script into the editor, and then click **OK**. For more information about scripts, see “Using Scripts for Additional Custom Processing”.
7. Click **Next**.

ePublisher displays the Documents window. This window lets you configure the groups and documents you want to generate output for in this job. You can either manually add the documents or you can run a script to retrieve documents within a group.

For example, you may want to retrieve documents from a version control or content management system. You can provide a script to retrieve and prepare your documents as needed. This script runs before the transformation, which ensures that you always have the most current version of your content without having to perform a manual update before each transformation.

8. ***If you want to manually add groups of documents to the job***, complete the following steps:
  - a. Click **New Group**.
  - b. Specify a name for the group.
  - c. Click **Add Document**, and then browse and select your source documents.

- d.** Repeat this process to add more groups and source documents, and then click **Next**.
- 9.** *If you want to use a script to add a group of documents*, complete the following steps:
  - a.** Click **New Group**.
  - b.** Specify a name for the group.
  - c.** Click **Edit Script**.
  - d.** Type or paste your script into the editor, and then click **OK**.
- 10.** Click **Next**.

ePublisher displays the Target Selection window. This window lets you select which targets to generate output for as part of this ePublisher AutoMap job.

- 11.** Select the check box in the **Build** column next to each target you want to generate, and then click **Next**.

ePublisher displays the Target Configuration window. The Stationery defines the configuration options for each target. You can override these configuration options for each target in a Stationery-based job. Depending on the output format of the selected target, the Target Configuration window provides tabs for adjusting various options, such as conditions and variables.

The information presented on each tab is relative to the target selected in the **Target Name** column. You can adjust values for each target independent of the other targets. For example, you can have two WebWorks Help 5.0 targets where the conditions are set differently depending on the target audience.

- 12.** *If you want to override the configuration settings defined in the Stationery for a target*, complete the following steps:
  - a.** Select the target you want to modify in the **Target Name** column.
  - b.** Specify the appropriate values for the configuration options on each tab for that target. For more information about the fields on a tab, click **Help**.
- 13.** Click **Finish**.
- 14.** Schedule the job as needed, and then click **OK**. You can also schedule the job at a later time. For more information, see "Scheduling Jobs with Windows Scheduler".

## Duplicating an Existing Job

Sometimes it is more convenient to make a copy of an existing job and adjust its options instead of creating a new job. ePublisher AutoMap allows you to duplicate an existing job. Then, you can modify the options as needed for your new job.

### To duplicate an existing job

1. Start ePublisher AutoMap.
2. Select the job you want to copy in the ePublisher AutoMap main window.
3. On the **Job** menu, click **Duplicate**.
4. Specify the new job name, and then click **OK**.
5. Specify the scheduling options as needed, and then click **OK**. For more information, see "Scheduling Jobs with Windows Scheduler".

Once the new job exists, you can modify the job for your specific needs. For more information, see "Editing an Existing Job".

## Editing an Existing Job

When you edit an existing job, ePublisher AutoMap presents the windows used to create the job as tabs of the Edit Job window. Select the appropriate tab that contains the information you want to modify.

### To edit an existing job

1. Start ePublisher AutoMap.
2. Select the job to edit in the ePublisher AutoMap main window.
3. On the **Job** menu, click **Edit**.
4. Select the appropriate tab and modify the values as needed. For more information about a field, click **Help**.
5. Click **OK**.

## Scheduling Jobs with Windows Scheduler

Immediately after creating a new job, ePublisher AutoMap starts the Windows Scheduler to allow you to schedule your job. ePublisher AutoMap uses the Windows Scheduler built into the Microsoft Windows operating system. The Windows Scheduler allows you to schedule a job to run at pre-determined times and repeating intervals. If you do not want to schedule the job, click **Cancel**. You can schedule the job at a later time, and you can modify an existing schedule.

When you schedule a job and then click **OK**, Windows prompts you for your Windows user name and password. For more information about using the Windows Scheduler, see the Windows operating system online help. To open the Windows online help, click **Help and Support** on the Start menu.



## To schedule a job or change the schedule for an existing job

1. Start ePublisher AutoMap.
2. Select the job to schedule or reschedule in the ePublisher AutoMap main window.
3. On the **Job** menu, click **Schedule Job**.
4. Specify the appropriate values, and be sure to add a Trigger that designates what schedule the Job will run, and then click **OK**.
5. If you select the Run whether user is logged on or not option, you will need to specify your Windows user name and password, and then click **OK**. Since you are scheduling a task in Windows Scheduler, you must provide your Windows user name and password to add the task. If you are part of a Windows domain, include the domain name, such as *domain\user*.

## Deleting an Existing Schedule for a Job

You can create multiple schedules for a job. These schedules are known as Triggers and can be modified in the Triggers tab in the Windows Scheduler interface. You can also delete one or more of these existing schedules. The only way to fully delete a task for a Job is to do so via the Windows Task Scheduler interface or to delete the Job itself.

### To delete an existing job

1. Start ePublisher AutoMap.
2. Select the job to delete the schedule for in the ePublisher AutoMap main window.
3. On the **Job** menu, click **Delete**.

## Running an Existing Job

You can run a job at any time whether it is scheduled or not. This feature allows you to test the job while you are configuring it. You can also run a job to quickly generate output based on a set of predefined settings.

### To run an existing job

1. Start ePublisher AutoMap.
2. Select the job you want to run in the ePublisher AutoMap main window.
3. On the **Job** menu, click **Run**.

## Viewing a Job Log File

When a job runs, ePublisher AutoMap creates a log file that documents the steps performed for that job. You can view this log file when one exists. ePublisher AutoMap keeps the log file only for the last time the job ran. Previous log files are not kept.

### To view a log file

1. Start ePublisher AutoMap.
2. Select the job to view the log for in the ePublisher AutoMap main window.
3. On the **Job** menu, click **View Log**.

## Canceling a Job

When you cancel a running job, ePublisher AutoMap, Microsoft Windows, Adobe FrameMaker, and Microsoft Word must do some clean-up work. During this process, Windows may display the standard Microsoft Windows End Program window. Do *not* click the **End Now** or **Cancel** buttons. The window will close usually in less than 15 seconds once everything has finished properly.

### **To cancel a running job**

- 1.** Start ePublisher AutoMap.
- 2.** Select the running job you want to cancel in the ePublisher AutoMap main window.
- 3.** On the **Job** menu, click **Stop**.

## **Deleting an Existing Job**

If you no longer need a job, you can delete it from the ePublisher AutoMap main window.

### To delete an existing job

1. Start ePublisher AutoMap.
2. Select the job to delete in the ePublisher AutoMap main window.
3. On the **Job** menu, click **Delete**.
4. Click **Yes** to confirm that you want to delete the selected job.

## Using Scripts for Additional Custom Processing

You do not need to write scripts to use ePublisher AutoMap. However, scripts allow you to extend ePublisher AutoMap and integrate it with other products and custom workflows. You can define scripts to run before and after a job generates content, to run before and after each target generates content, and to retrieve source documents on a per group basis.

## Writing Scripts

You can write scripts directly in the ePublisher AutoMap script editor window, or you can cut and paste scripts from another text editor. The ePublisher AutoMap script editor supports only text-based scripts. Any formatting or additional information available in a third-party script editor is lost when the script is pasted into the ePublisher AutoMap script editor.

ePublisher AutoMap treats scripts like DOS batch files. The script must be complete and valid or it will fail when the job runs. Although you can write all your scripts directly in the ePublisher AutoMap script editor; this approach may not be your best option.

When calling complex scripts, you may want to create and store those complex scripts in your file system and simply call those files from the script you create in the ePublisher AutoMap script editor. Since the script editor treats the scripts like DOS batch files, you can call other batch files, applications, or scripts written in any scripting language, such as VBScript, Perl, and Python. You can also pass parameters and script variables to the files you call. However, some variables have meaning only for certain types of scripts. For example, the `DeployFolder` variable is specific to a target and does not have meaning in a pre-build or post-build job script.

## Working Folder

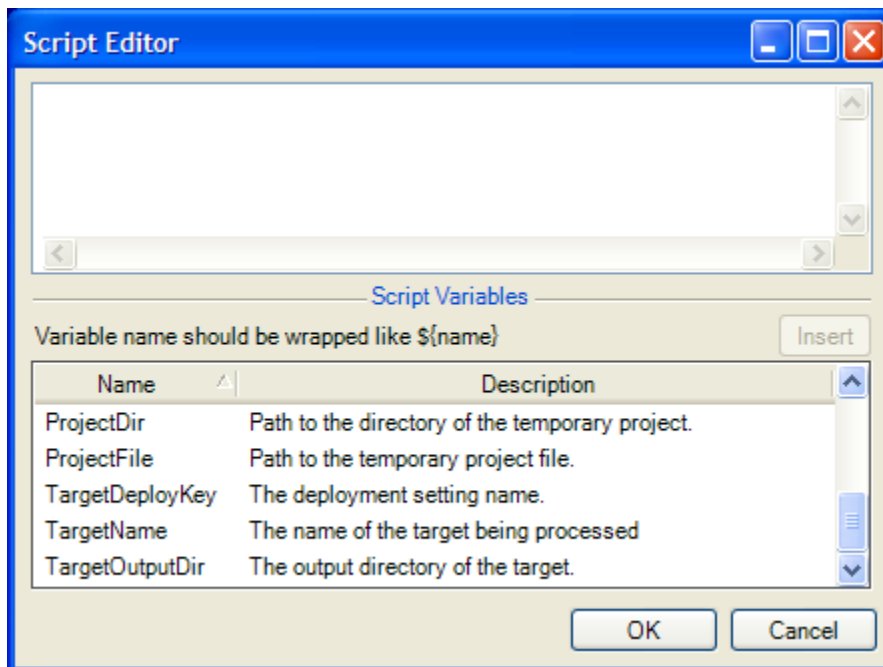
Windows associates batch files with a working folder. The working folder is the current working directory used by an application or batch file for processing.

For example, if a batch file creates a new file without specifying a path, the file is created in the working folder. If a batch file attempts to read a file without a specifying a path, Windows assumes the file is in the working folder.

The working folder for an ePublisher AutoMap batch file is the job folder itself. You can locate this folder using the job directory `${JobDir}` scripting variable. For more information, see “Using Scripting Variables Example”.

## Opening and Using the Script Editor

ePublisher AutoMap includes an editor for specifying and writing job scripts. This editor provides a text area for writing or pasting your script and a list of ePublisher AutoMap variables that you can use in your scripts or pass to other scripts and applications.



## To open the script editor and edit a script

1. Start ePublisher AutoMap.
2. Select the job to edit in the ePublisher AutoMap main window.
3. On the **Job** menu, click **Edit**.
4. Click the appropriate **Edit Script** button to open the script editor window for the script you want to edit.
5. Click in the text editing area.
6. Type your script, or press Ctrl+V to paste it from the clipboard.
7. ***If you want to insert an ePublisher AutoMap scripting variable,*** complete the following steps:
  - a. Click in the editor window where you want to insert the variable.
  - b. Double-click the variable name in the **Script Variables** pane to insert it. You can also type the variable name surrounded by `${}`. For example, to include the project directory variable in your script, type `${ProjectDir}`.

## Scripting Variables

The following table provides a summary of the available scripting variables.



Variable	Description	Scope
FileListName	Returns the name of the file that contains the list of source documents.	Document scripts only
FileListPath	Returns the path of the file that contains the list of source documents.	Document scripts only
GroupName	Returns the name of the documents group that is currently being processed.	Document scripts only
BuildAction	Returns whether the current build action is pre-build or post-build.	Job scripts, target scripts, and document scripts
JobDir	Returns the path of the job <code>.waj</code> file.	Job scripts, target scripts, and document scripts
JobFile	Returns the name of the job <code>.waj</code> file.	Job scripts, target scripts, and document scripts
JobName	Returns the name of the job.	Job scripts, target scripts, and document scripts
ProjectDir	Returns the path of the temporary project file that ePublisher AutoMap created for the job.	Job scripts, target scripts, and document scripts
ProjectFile	Returns the name of the temporary project file that ePublisher AutoMap created for the job.	Job scripts, target scripts, and document scripts
DeployFolder	Returns the deployment directory path.	Target scripts only

Variable	Description	Scope
ErrorCount	Returns the number of errors reported during the generation process.	Target scripts only
TargetDeployKey	Returns the deployment target name.	Target scripts only
TargetName	Returns the name of the target being generated.	Target scripts only
TargetOutputDir	Returns the output path of the target.	Target scripts only

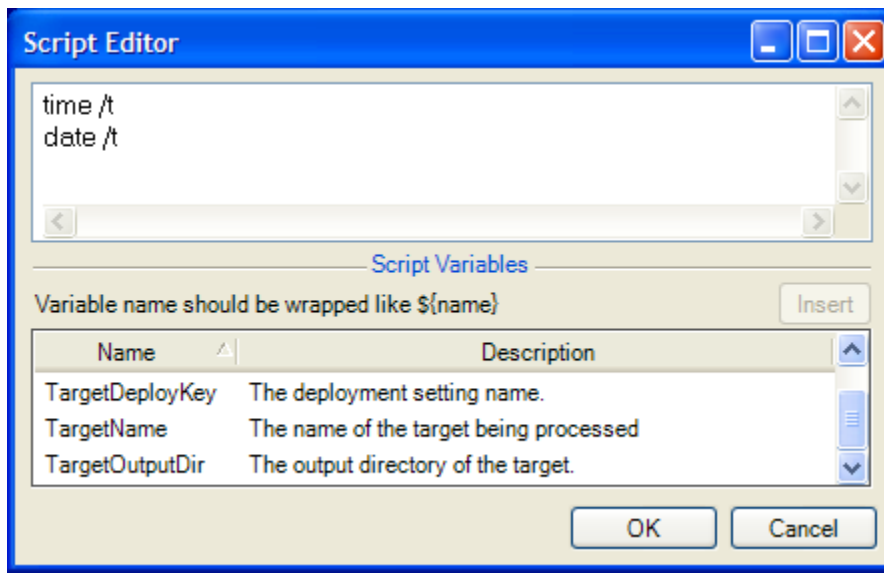
## Scripting Examples

The following list highlights a few of the ways you can use scripts to customize and integrate the content publication process:

- You can write scripts to get the latest version of files from your version control system or content management system. For more information, see “CVS Version Control Checkout Example”.
- You can use variables within scripts. For more information, see “Scripting Variables” and “Using Scripting Variables Example”.
- You can customize the log file created when ePublisher runs a job. For more information, see “Show Time and Date Example” and “Using Scripting Variables Example”.
- You can use post-processing scripts to customize the deployment process. These scripts can modify files and even rebuild the final deliverable. For example, you can use scripts to further customize the generated files and then automatically run Microsoft HTML Help Workshop to rebuild the `.chm` file.

## Show Time and Date Example

This example simply displays the time and date in the log file by calling the time and date commands built into the DOS command-line interface. In this case, the script was added as a pre-build step for the job running before the first target starts generating output.



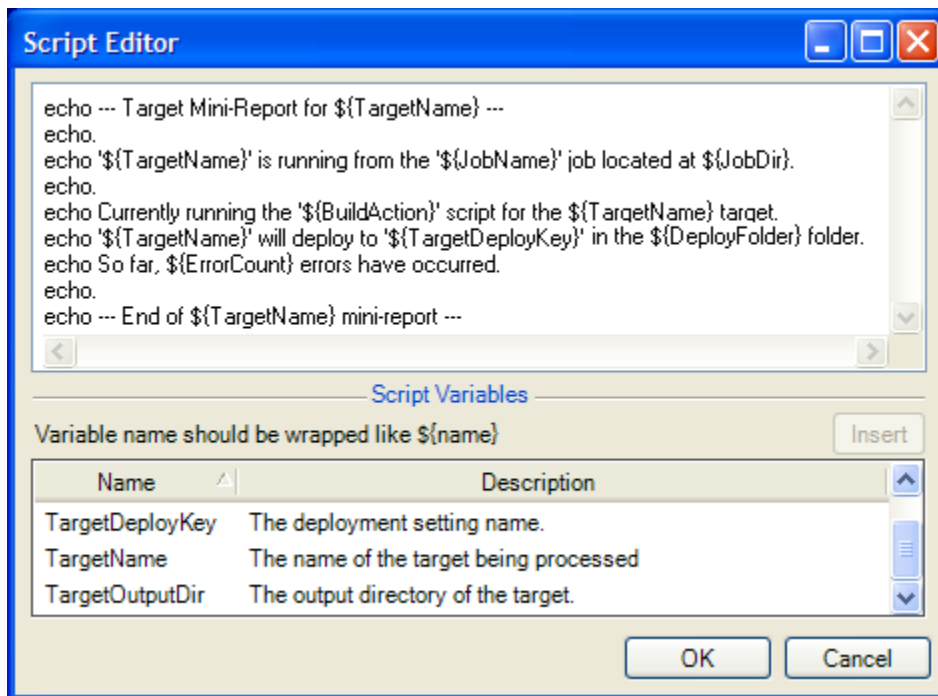
The following output shows the time and date included in the log before the first target starts the generation process. This information can automatically timestamp a job log file. You can also use these values in your script to log elapsed time and make decisions.

```
WebWorks Automap
09:57 AM
Mon 6/23/2008
Scanning 1 document(s)...
Building target: WebWorks Help 5.0
Generation started at 9:57:15 AM
Initializing file information
Updating documents.
Applying settings to WebWorks.doc, 1 of 1.
```

## Using Scripting Variables Example

This example creates a simple mini-report inside the log file for each target in the job. This simple example demonstrates the use of the scripting variables. In this case, the values of the variables are displayed as part of the report using the echo command built into the DOS command-line interface.

The following figure shows some of the variable names surrounded by single quotes. You do not need to enclose variable names in quotes. These quotes are there only to show emphasis on the variable values in the mini-report example.



The following output shows the mini-report displayed immediately after the target starts to be built. You could add this same script as a post-build step for the target and a similar mini-report would be created when the target is done being generated.

```
Building target: WebWorks Help 5.0

--- Target Mini-Report for WebWorks Help 5.0 ---
'WebWorks Help 5.0' is running from the 'Scripting Demo' job located
at C:\Documents and Settings\doc\My Documents\WebWorks Automap\Jobs
\Demo.
Currently running the 'PreBuild' script for the WebWorks Help 5.0
target. 'WebWorks Help 5.0' will deploy to 'Online Help' located in
the C:\AutoMapOutput\Help Systems\Online Help folder.
So far, 0 errors have occurred.
--- End of WebWorks Help 5.0 mini-report ---

Generation started at 9:57:15 AM
Initializing file information
Updating documents.
Applying settings to WebWorks.doc, 1 of 1.
```

## CVS Version Control Checkout Example

The previous example scripts demonstrate some simple capabilities of using scripts in ePublisher AutoMap. This example shows how to check out source files from a version control system at the start of a job.

Version control systems let you store files and retain information concerning different versions of those files. You can store multiple versions of files, revert to previous versions of files, and share files among large groups of people. The advantage of using a version control script to retrieve source files rather than adding them directly to a project ensures that your job is always working with the latest checked in version of each source file.

Each version control system works in a slightly different way. ePublisher AutoMap is not tied to a specific version control solution. ePublisher AutoMap provides an abstract way of retrieving a list of source files to transform. There are a few rules about how this list is named and formatted, but the actual creation of the list is left to the script and scriptwriter.

The scripting capabilities in the Documents panel are not limited to checking files out of a version control system. The only requirement is that the `FileList.txt` file includes a list of files and paths to be transformed. You can access the `FileList.txt` file and the path to it using the ePublisher AutoMap `${FileListName}` and `${FileListPath}` scripting variables. For more information, see "Scripting Variables".

The following script example calls `getfilesaction_cvs.vbs`, which is installed with ePublisher AutoMap. By default, this script is located in the following location:

```
\Program Files\WebWorks\ePublisher AutoMap\Scripts  
\getfilesaction_cvs.vbs
```

The `getfilesaction_cvs.vbs` script is written using the Visual Basic Scripting (VBS) capabilities built into Microsoft Windows to check out files from a CVS (Concurrent Version System) server and create a file list to be processed by ePublisher AutoMap. This script is not meant to be used in its current form. This script provides a starting point that includes everything you need to set up a working script for your particular environment.

The `getfilesaction_cvs.vbs` script includes comments to help someone with scripting experience understand the actions it performs. Since this script provides only a starting point, you need to modify this script for your environment. For example, you need to set the CVS attributes for your user name, server name, and type of authentication. You also need to adjust for how CVS is installed and called in your environment. These areas of the script are highlighted in the following figure:

```

: name and the CVSROOT parameters.
CVSRoot = ":pserver:username@machinename:/remote_cvs_directory"

: Main Script:
: Checks out document files from cvs and creates the
: file FileList.txt that contains each file that matches a specified
: file extension.

: This variable will hold a list of filenames and paths to be written to
: FileList.txt. This list will be filled out in the GetMatchingFiles
: function defined below.
Set VarFileList = CreateObject("Scripting.Dictionary")

: Set the CVSROOT environment variable, which is required by CVS
: so that it can locate files to be checked out.
GlobalEnv("CVSROOT") = CVSRoot

: Set the current working directory so that the checked out files
: will be placed into the correct directory.
GlobalShell.CurrentDirectory = CVSLocalDir

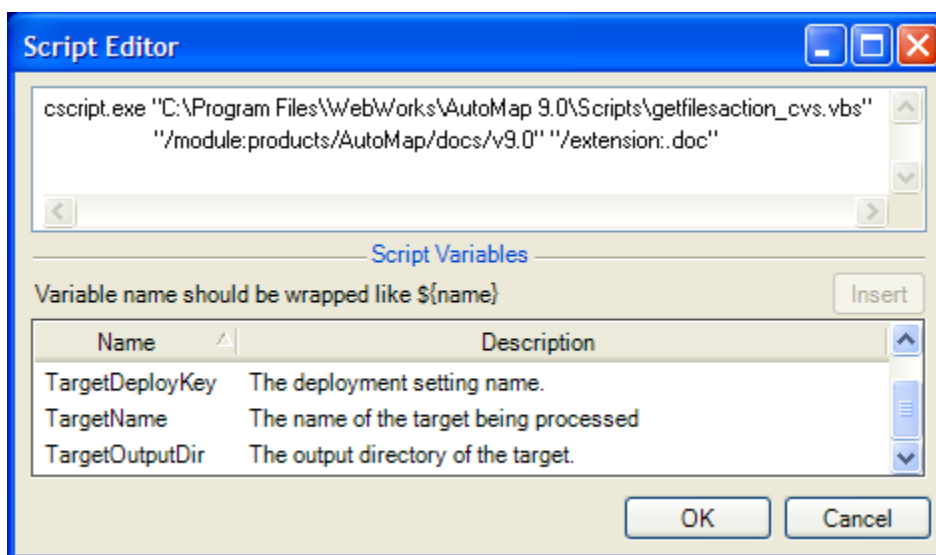
: Execute the CVS command for checking out a module. Double-quotes are placed
: around the CVSModule value to allow for spaces in the CVS module name passed in
: on the command line.
NOTE: If the files being checked out are on a cvs branch, you must customize the
line below by removing the -A part of the command and replacing it with
-r <BranchName>.
NOTE: Also, if the path to cvs.exe is not listed in your system environment
variables, you will need to add the path to cvs to your system environment
variables or change the line below to point to the full path of cvs.exe
on your system. To test whether the call below will work, open a command
prompt window and type cvs at the prompt. If cvs is recognized by your
system, then the call below will work as is.

set ExecCall = GlobalShell.Exec("%comspec% /c cvs checkout -A " & chr(34) & CVSModule & chr(34))

: Loop and sleep until the cvs checkout command is finished executing. The loop will
: end if the execution completes or the loop counter reaches 11. If it is necessary
: to wait longer for large cvs checkouts, you can customize the loop counter limit or
: the sleep time between checks.

```

The following figure shows an example of how to call the `getfilesaction_cvs.vbs` script. The parameters passed indicate the module to checkout and to filter on the `.doc` extension. When this script runs, it checks out the indicated module from CVS, filters those files based on the `.doc` extension, and adds those `.doc` files to the `FileList.txt` file. That file list is then used by the job as the list of files to process and transform.



This example is only one possible way of creating a list of files to transform. Once you understand the mechanism that ePublisher AutoMap uses to retrieve its list of source documents to transform, you can create that list using the process you prefer. Whether you want to use a script or an application, ePublisher AutoMap can run it from the script editor as long as it is available to ePublisher AutoMap.

## Using the Command-Line Interface

In addition to the ePublisher AutoMap console that allows you to create, edit, and schedule jobs, ePublisher AutoMap also provides a command-line interface (CLI). The CLI gives you control of ePublisher AutoMap from the Windows command-line interface. You can use the CLI and the supported options in scripts and batch files to automate and streamline your processes.

## Running ePublisher AutoMap from the Command Line

You can run ePublisher AutoMap from the command line using the ePublisher AutoMap `WebWorks.Automap.exe` command-line application. This application is installed at the root of your ePublisher AutoMap installation location. The default location for this file is:

```
Program Files\WebWorks\ePublisher AutoMap\WebWorks.AutoMap.exe
```

### Notes:

- If you install ePublisher AutoMap in another location, the `WebWorks.Automap.exe` file is installed in that location and you need to adjust your specified path.
- The `WebWorks.Automap.exe` file depends on other files installed in the ePublisher AutoMap folder. Do not move or copy the `WebWorks.Automap.exe` file.

## To run ePublisher AutoMap from the command line

1. Open a Windows command prompt by completing the following steps:
  - a. On the **Start** menu, click **Run**.
  - b. Type `cmd`, and then click **OK**.
2. Enter the following command to change to the default ePublisher AutoMap folder:

```
CD \Program Files\WebWorks\ePublisher AutoMap
```

3. Enter the `WebWorks.AutoMap.exe` command with the appropriate options.

If you run the `WebWorks.AutoMap.exe` command without any options, ePublisher AutoMap displays a typical usage syntax statement.

## CLI Syntax and Reference

The ePublisher AutoMap command-line interface can run a job or an ePublisher project. The command-line interface uses the following syntax:

```
WebWorks.Automap.exe [-f] [-n] [-c] [-l] [-u] [-j] [-d directory]
```

```
[[[-s directory] jobfile]|{[-t targets] projectfile}]
```

The supported command-line options are defined in the following table.



Option	Description
<code>-c</code>	Deletes cached information and builds the output from scratch. Use this option only when passing an ePublisher project <code>.wrp</code> file. This option is equivalent to selecting <b>Regenerate All</b> . The default is to generate only what is needed and to use all available cached information. You can also specify this option as <code>--clean</code> .
<code>-d</code> <i>directory</i>	Specifies an alternate deployment directory. This setting overrides the specified default values. You can also specify this option as <code>--deployfolder</code> <i>directory</i> .
<code>-u</code>	Scans all source documents for new styles, conditions, variables and xrefs. After the conversion is completed the resulting Express project file will be saved with all the updated information. You can also specify this option as <code>--update</code> .
<code>-j</code>	Just scans all source documents for new styles, conditions, variables and xrefs. Saves Express project file without running a conversion. You can also specify this option as <code>--justupdate</code> .
<code>-f</code>	Does not send an email notification when this job is finished. This setting overrides the default value specified in the ePublisher AutoMap preferences. You can also specify this option as <code>--nonotify</code> .
<code>-l</code>	Deletes all files in the deployment location before deploying the newly-generated output. This option ensures that all files in the deployment folder are from this last generation process. You can also specify this option as <code>--cleandeploy</code> .
<code>-n</code>	Generates output and does not deploy it. This setting overrides the default deployment settings specified in

Option	Description
	the job or ePublisher project. You can also specify this option as <code>--nodeploy</code> .
<code>-s</code> <i>directory</i>	Specifies the staging directory to use. Use this option only when passing a job <code>.waj</code> file. This setting overrides the default value specified in the ePublisher AutoMap preferences. You can also specify this option as <code>--stagingdir</code> <i>directory</i> .
<code>-t</code> <i>targets</i>	Builds only the specified targets. Separate multiple targets with a comma and no space on either side of the comma. Use this option only when passing an ePublisher project <code>.wrp</code> file. The default is to build all targets. You can also specify this option as <code>--target</code> <i>targets</i> .
<i>jobfile</i>	Specifies the name of the ePublisher AutoMap <code>.waj</code> job file.
<i>project</i>	Specifies the name of the ePublisher <code>.wrp</code> project file.

As with other command-line applications, specify the command-line options on the command line after the application name itself. Separate multiple command-line options with a space. Review the following additional notes concerning ePublisher AutoMap command-line options:

- The command-line options have both shortened and verbose formats. For example, you can specify the clean option as `-c` or `--clean`. Both options mean exactly the same thing and you can mix the shortened and verbose formats across the options.
- Some options require additional information. For example, the staging folder option requires the name of the directory and is specified as `--stagingdir` *directory*. You need to include the equals sign without spaces on either side.
- In cases where you specify paths with spaces, enclose those paths in double quotes to correctly handle the paths. You do not have to enclose paths without spaces in double quotes, but you can without an issue. For example, `c:\projects` works without enclosing it in double-quotes. However, `c:\my projects` does not work as expected. The correct format for the latter path is `"c:\my projects"`.

## CLI Examples

The following examples illustrate how to use the command-line interface.

### Running a Project and Updating the Express project file

The following example runs the `c:\Projects\MyProject.wrp` project and updates `MyProject.wrp` with the new styles, conditions, variables and xrefs:

```
WebWorks.AutoMap.exe -u "c:\Projects\MyProject.wrp"
```

### Running a Project and Generating Only One Target

The following example runs the `c:\Projects\MyProject.wrp` project and generates only the `WebWorks Help` target:

```
WebWorks.AutoMap.exe -t "WebWorks Help" "c:\Projects\MyProject.wrp"
```

### Running a Project from Scratch and Deploying to a Clean Location

The following example runs the `c:\Projects\MyProject.wrp` project and generates all targets defined in the project. This example also deletes the cached information in ePublisher to ensure it generates all the content from scratch, and it deletes all the files in the deployment location before deploying the newly-generated files:

```
WebWorks.AutoMap.exe --clean --cleandeploy "c:\Projects\MyProject.wrp"
```

### Running a Project and Deploying to an Alternate Location

The following example runs the `c:\Projects\MyProject.wrp` project and generates all targets defined in the project. This example also deploys all the newly generated files to the `\\TestServer\Review` folder:

```
WebWorks.AutoMap.exe -d "\\TestServer\Review" "c:\Projects\MyProject.wrp"
```

### Running a Job Without Sending Notification When Done

The following example runs the `c:\Jobs\MyJob.waj` job and does not send any email notification when the job is done:

```
WebWorks.AutoMap.exe --nonotify "c:\Jobs\MyJob.waj"
```

## Running a Job and Deploying to a Clean Location

The following example runs the `c:\Jobs\MyJob.waj` job and deletes all the files in the deployment location before deploying the newly-generated files:

```
WebWorks.AutoMap.exe --cleandeploy "c:\Jobs\MyJob.waj"
```

## Running a Job Without Deploying the Content

The following example runs the `c:\Jobs\MyJob.waj` job and does not deploy the generated output files. This example also uses the `C:\Temp\Stage` folder as the staging folder when running the job:

```
WebWorks.AutoMap.exe --nodeploy -s "C:\Temp\Stage" "c:\Jobs\MyJob.waj"
```

The staging folder provides a working folder for processing the job. ePublisher AutoMap stores the automatically generated ePublisher project, intermediate data files, and the output files in this folder.

# Markdown++ Source Documents

[Introduction](#)  
[Getting Started with Markdown](#)  
[Learning Markdown](#)  
[Learning Markdown++](#)

## Introduction

Markdown is a text-based authoring format created by [John Gruber](#). It's a simple, light-weight and robust language that puts emphasis on efficiency and readability.

### Quick Links

[Getting Started with Markdown](#)

[Learning Markdown](#)

[Learning Markdown++](#)

[Markdown Cheat Sheet](#)

These sections will go into detail on authoring in Markdown and Markdown++, how-tos for syntax features, as well as how to prepare documents for publishing in ePubliher.

## Getting Started with Markdown

Markdown is a text-based authoring format. Any text editor can be used to create Markdown documents. Here are some popular ones to try out:

- [Notepad++](#). Simple, lightweight notepad app with syntax highlighting based on file extension.
- [Visual Studio Code](#). Code-centric text editor. Has community-made extensions, including many for Markdown.
- [Obsidian](#). A Markdown-specific note taking app. Has themes, a writing mode, and a preview mode.
- [Typora](#). A Markdown editor with many authoring experience features.

After choosing a text editor, Markdown and Markdown++ documents can be created using the `.md` extension.

## Learning Markdown

This section will detail the features of Markdown, how to write them, and how to use them in ePublisher. For quick reference material, see the [Markdown Cheat Sheet](#).

Any text can be Markdown. Common prose parses into Markdown with no issue. Simple text documents can be formatted into Markdown documents quickly and easily because of this.

## Quick Links

[Paragraphs](#)

[Titles](#)

[Headings](#)

[Lists](#)

[Tables](#)

[Blockquotes](#)

[Code Fences](#)

[Code Blocks](#)

[Horizontal Rule](#)

[Block HTML](#)

[Bold, Italic, Strikethrough, Code](#)

[Links](#)

[Images](#)

[Link References](#)

[Inline HTML](#)

# Paragraphs

The basic organization of block-level text, the paragraph is the building block of a Markdown document.

## Syntax

A Paragraph is created by writing any text content on a line. It is the default block-level element, meaning all content is considered a Paragraph if the content does not have any recognizable block-level syntax.

## Basics

Any amount of text will do to create a Paragraph. Start the line with non-space characters to avoid indentation-related parsing issues.

```
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do  
eiusmod tempor incididunt ut labore et dolore magna aliqua.
```

## Separate with Empty Lines

Keep an empty line between Paragraphs that should be separated. This is a general good rule of thumb for all Markdown content.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

## Multi-Line Paragraphs

Multiple lines not separated by an empty line will be treated as parts of the same Paragraph. The lines will be consolidated and separated by space in the output.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

## Preserve Line Breaks

Ending a line with a space character at the end will preserve the line break within the Paragraph. Useful for poetry, or other types of content where line structure is important.

Nature's first green is gold,

Her hardest hue to hold.

Her early leaf's a flower;

But only so an hour.

## Markdown++

A custom Paragraph Style can be given to a Paragraph using a Markdown++ style tag on the line directly above the Paragraph.

```
<!--style:CustomParagraph-->
```

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

# ePublisher Style Information

## Default Style Properties

Style Type: **Paragraph**  
Style Name: **Paragraph**



Property	Value
font family	Arial
font size	12pt
line height	1.2em
padding top	0pt
padding bottom	6pt

If a custom style name is assigned to a Paragraph, that style name will still inherit all of the listed default style information.

## Titles

Also referred to as a *setext heading*, Titles are useful to communicate the central idea of a document. Titles are most useful as the leading content of a set of text material.

### Syntax

Titles are created by writing a single line of content followed by a line containing at least 1 of either `=` or `-` characters. The second line shouldn't contain text other than these two characters.

### Basics

The most basic example, a line of content with a following line with a `=` character.

```
My Document Title
=
```

Titles can be written in the same way using `-` characters.

```
My Document Title
-
```

### Any Amount of Characters

The amount of `=` or `-` characters that are used is not important. Having a matching amount of characters on both lines can be a nice touch for readability, though.

My Document Title

=====

## Markdown++

A custom Paragraph Style can be given to a Title using a Markdown++ style tag on the line directly above the Title.

```
<!--style:CustomTitle-->
```

My Document Title

=====

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Style Behavior

The style name a Title will get is dependent on the characters used in the second line. **Title 1** is given to Titles that use = characters, and **Title 2** is given to Titles that use - characters.

### Default Style Properties

Style Type: **Paragraph**

Style Name: **Title 1, Title 2**

Property	Value
font family	Arial
font size	24pt
font weight	bold
line height	1.2em
padding top	0pt
padding bottom	12pt

## Default Style Options

Option	Value
Table of Contents level	1

If a custom style name is assigned to a Title, that style name will still inherit all of the listed default style information.

## Headings

Originally named the *ATX heading*, a Heading communicates a central idea for a topic. Headings should contain the main idea for a section, and have useful keywords to make the section easy to find in a search.

### Syntax

Headings are created by starting a line of content with the `#` character. The `#` characters and the text content of the Heading need to be separated by a space character. The amount of `#` characters used indicates the level of heading which will be created.

### Basics

Create a Heading 1 with a single `#`, a space, and some text.

```
# Heading 1
```

More `#` characters can be added to the Heading to increase the heading level, up to 6.

```
# Heading 1
```

```
## Heading 2
```

```
### Heading 3
```

```
#### Heading 4
```

```
##### Heading 5
```

```
##### Heading 6
```

## Markdown++

A custom Paragraph Style can be given to a Heading using a Markdown++ style tag on the line directly above the Heading.

```
<!--style:CustomHeading-->
```

```
# Heading 1
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## Heading Behavior

### Heading Alias

Each created Heading gets an alias that can be used to [link to it](#) from another place in the publication.

To determine the alias value, ePublisher takes the text of the Heading, lower-cases it, removes all non-alphanumeric characters, and replaces space with `-` characters.

The below Heading will get the alias value `lets-go-to-the-moon`.

```
# Let's Go to the Moon!
```

Any time the text of a Heading is changed, the alias will also change. It's recommended to use a [Custom Alias](#) to avoid having to change link paths when Headings change.

# ePublisher Style Information

## Style Behavior

The style name ePublisher will create for a Heading will be dependent on the number of # characters used at the front of the line. One # character creates the style name **Heading 1**, two # characters creates **Heading 2**, etc.

## Default Style Properties

Style Type: **Paragraph**

Style Name: **Heading 1, Heading 2, Heading 3, Heading 4, Heading 5, Heading 6**

**Heading 1**

Property	Value
font family	Arial
font size	21pt
font weight	bold
line height	1.2em
padding top	0pt
padding bottom	12pt

## Heading 2

Property	Value
font family	Arial
font size	18pt
font weight	bold
line height	1.2em
padding top	0pt
padding bottom	12pt

### Heading 3



Property	Value
font family	Arial
font size	15pt
font weight	bold
line height	1.2em
padding top	0pt
padding bottom	12pt

## Heading 4, Heading 5, Heading 6

Property	Value
font family	Arial
font size	12pt
font weight	bold
line height	1.2em
padding top	0pt
padding bottom	12pt

## Default Style Options

### Heading 1

Option	Value
Table of Contents level	2

**Heading 2**

Option	Value
Table of Contents level	3

**Heading 3**

Option	Value
Table of Contents level	4

**Heading 4**

Option	Value
Table of Contents level	5

**Heading 5**

Option	Value
Table of Contents level	6

**Heading 6**

Option	Value
Table of Contents level	none

If a custom style name is assigned to a Heading, that style name will still inherit all of the listed default style information for the matching Heading syntax.

## Lists

Lists are a structural feature in Markdown. They're useful for many things, such as itemizing a collection of information, providing steps to a procedure, or numbering sections of information.

### Syntax

There are two types of lists that can be created: Ordered Lists and Unordered Lists.

Ordered Lists are created by starting a line with a number or letter, followed by a single `.`, then space (two is recommended), then some text content.

Unordered Lists are created by starting a line with any `-`, `*`, or `+` character, followed by a space, then some text content.

Beyond these differences, both types of Lists have the same behavior when it comes to syntax and authoring.

### Basics

#### Ordered List

Create a simple Ordered List using a number and a `.` character. Each list item is written on it's own line.

```
1. list item one
2. list item two
3. list item three
```

Ordered Lists can also be created using letters and `.`.

```
a. list item one
b. list item two
c. list item three
```



Roman numerals are fine, too. Remember to keep the vertical spacing consistent.

```
i. list item one  
ii. list item two  
iii. list item three
```

Re-using the same letter or number is OK.

```
1. list item one  
1. list item two  
1. list item three
```

```
a. list item one  
a. list item two  
a. list item three
```

## Unordered List

Create a simple Unordered List using `-`. Each list item is written on it's own line.

```
- list item one  
- list item two  
- list item three
```

Unordered Lists can also be created using `*`.

```
* list item one  
* list item two  
* list item three
```

The `+` character can be used as well.

```
+ list item one  
  
+ list item two  
  
+ list item three
```

## One Empty Line Between List Items

Put a single empty line between list items to give room. More than one empty line will break the list into two.

```
- list item one  
  
- list item two  
  
- list item three
```

## Don't Use Unlike Characters

Using non-matching characters on the same list level will break the list in two.

```
- list item one  
  
* list item one  
  
+ list item one
```

```
1. list item one  
  
a. list item one
```

## Multi-Line Content in List Items

List Item content can span multiple lines. Use a blank line to separate elements. Make sure all lines of content retain the same vertical spacing.

## 1. ### Cities in the US

Here is a sample of some cities in the United States.

Name	State
Austin	Texas
Tulsa	Oklahoma

## 2. list item two

## - ### Cities in the US

Here is a sample of some cities in the United States.

Name	State
Austin	Texas
Tulsa	Oklahoma

## - list item two

## Nested List Items

To nest List items, make sure the vertical spacing of the nested List item matches up with the content of the parent List item.

1. list item one

1. nested list item one

2. list item two

3. list item three

- list item one

- nested list item one

- list item two

- list item three

## Nesting Different Types of Lists

Nesting Lists of different types is acceptable. Use the same spacing rules as usual.

1. list item one

- nested list item one

- nested list item two

2. list item two

3. list item three

```
- list item one

  1. nested list item one

  2. nested list item two

- list item two

- list item three
```

## Markdown++

A custom Paragraph Style can be given to a List using a Markdown++ style tag on the line directly above the List.

```
<!--style:CustomOList-->

1. A customized ordered list, style name "CustomOList"

2. CustomOList item two

3. CustomOList item three
```

```
<!--style:CustomUList-->

- A customized unordered list, style name "CustomUList"

- CustomUList item two

- CustomUList item three
```

## Customizing Nested Lists

Nested Lists can be customized as well.

1. A default list, style name "OList"

```
<!--style:CustomOList-->
```

a. A customized list, style name "CustomOList"

b. CustomOList item two

2. OList item two

- A default list, style name "UList"

```
<!--style:CustomUList-->
```

- A customized list, style name "CustomUList"

- CustomUList item two

- UList item two

## Default Until Customized

Nested Lists are treated as standalone; they will not inherit the outermost style name if customized.

```
<!--style:CustomOList-->
```

1. A customized list, style name "CustomOList"

a. A default list, style name "OList"

b. OList item two

2. CustomOList item two

```
<!--style:CustomUList-->
```

```
- A customized list, style name "CustomUList"
```

```
- A default list, style name "UList"
```

```
- UList item two
```

```
- CustomUList item two
```

Add a style tag to each list individually for consistency with custom styles.

```
<!--style:CustomOList-->
```

```
1. A customized list, style name "CustomOList"
```

```
<!--style:CustomOList-->
```

```
a. A customized list, style name "CustomOList"
```

```
b. CustomOList item two
```

```
2. CustomOList item two
```

```
<!--style:CustomUList-->
```

```
- A customized list, style name "CustomUList"
```

```
<!--style:CustomUList-->
```

```
- A customized list, style name "CustomUList"
```

```
- CustomUList item two
```

```
- CustomUList item two
```

## Nested Content in Lists

The tagging convention can be used for other Markdown elements inside List items. The resulting style name will be appended with the style name of the containing List.

```
1. <!--style:CustomParagraph-->

    A customized paragraph, style name "OList CustomParagraph"

2. list item two
```

```
- <!--style:CustomParagraph-->

    A customized paragraph, style name "UList CustomParagraph"

- list item two
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Style Behavior

To allow full styling of Lists, ePublisher creates a number of style names when a list is detected inside a Markdown source document.

#### List Style

The List Style is the first style that ePublisher adds to the Style Designer when a list is detected in a source document. The default name is **OList** for ordered lists, and **UList** for unordered lists, but could also be a custom name if the style tag syntax is used on the list.

This style applies to the container area surrounding the lists's items. It's style rules can also apply to list items or nested content, if the same rule isn't already applied on a nested style.

#### Customizing the List Style

By adding a Markdown++ custom style tag, the List Style name can be changed. The example below changes the List Style name to **CustomUList**:



```
<!--style:CustomUList-->

- This is a custom list

- unordered

- named "CustomUList"
```

## List Item Style

The list items inside a list also get a style name. To determine the List Item Style's name, ePublisher takes the List Style and adds `Item` to the end, separated by a space. The default name is **OList Item** for ordered list items, and **UList Item** for unordered list items, but could also be a custom name if the style tag syntax is used on the list.

### Customizing the List Item Style

By adding a Markdown++ custom style tag, the List Item Style name can be changed. The example below adds the List Item Style **CustomUList Item**, because the List Style name has been set to **CustomUList**:

```
<!--style:CustomUList-->

- This is a custom list

- unordered

- named "CustomUList"
```

List items can't be styled individually. This will break the list into two separate lists. All list items in a given list are styled by the same List Item Style.

## Nested Styles

Nested content inside of list items also get a new style name. To determine the Nested Style's name, take the List Style and add the style name of the nested content to the end, separated by a space.

The example below populates the Style Designer with 3 Paragraph Styles: **UList** (the List Style), **UList Item** (the List Item Style), and **UList Paragraph** when scanned into ePublisher.

- This is a simple list
- unordered
- default style names

## Customizing Nested Styles

By adding a Markdown++ custom style tag, the Nested Style name can be changed. The example below changes the Nested Style name to **UList CustomParagraph**:

```
- <!--style:CustomParagraph-->  
  
  This is a custom paragraph.
```

Custom Style Names can be used on both the list and nested content simultaneously. This example creates the style names **CustomUList**, **CustomUList Item**, and **CustomUList CustomParagraph**:

```
<!--style:CustomUList-->  
  
- <!--style:CustomParagraph-->  
  
  This is a custom paragraph inside a blockquote.
```

## Default Style Properties

Style Type: **Paragraph**

Style Name: **OList**, **OList Item**, **UList**, **UList Item**

**OList**

Property	Value
padding top	0pt
padding right	0pt
padding bottom	0pt
padding left	0pt
margin top	0pt
margin right	0pt
margin bottom	0pt
margin left	0pt
tag	ol

## UList

Property	Value
padding top	0pt
padding right	0pt
padding bottom	0pt
padding left	0pt
margin top	0pt
margin right	0pt
margin bottom	0pt
margin left	0pt
tag	ul

## **OList Item, UList Item**

Property	Value
padding top	0pt
padding right	0pt
padding bottom	0pt
padding left	0pt
margin top	0pt
margin right	0pt
margin bottom	0pt
margin left	36pt
tag	li

If a custom style name is assigned to a List, that style name will still inherit all of the listed default style information.

## Tables

Tables lay out multiple lines of detailed data in an organized way. In Markdown, Tables are used to display cells of inline content. This often means that table structure is kept simple.

If a Table with complex structure is needed, it can be created as an HTML fragment in a [Block HTML](#) element.

**Markdown++ also supports multiline tables, which provide additional flexibility for structuring content.** To learn more, see the [Multiline Tables in Markdown++](#) page.

## Syntax

Markdown Tables consist of 3 things:

- A **header row**, which contains header cell content separated by `|` characters.
- An **alignment row**, that indicates the alignment of the body cells' text. Each cell in this row contains at least 3 `-` characters, and an optional `:` character to indicate alignment. Each cell is separated by a `|` character.

- Default alignment only uses `-` characters; 3 or more.
- Left align the column by starting the cell with `:` and filling in the rest with `-` characters; 3 or more.
- Right align the column by starting the cell with 3 or more `-` characters, ending with a `:` character.
- Center align the column by starting and ending the cell with `:` characters. Put `-` characters between them; 3 or more.
- 1 or more **body rows**, that contain body cell content separated by `|` characters.

Each row's content should be confined to a single line. The table will not parse properly if rows have multi-line content.

Optionally, all lines in the table can start and end with `|` characters. Be sure to apply them to all lines if they are to be used.

## Basics

Two basic Tables; one with wrapping `|` characters, one without.

```
| name | age | city |
|---|---|---|
| Bob | 42 | Dallas |
| Mary | 37 | El Paso |
```

```
name | age | city
---|---|---
Bob | 42 | Dallas
Mary | 37 | El Paso
```

Line up the `|` characters in each row for a nice touch for readability.

```
| name | age | city      |
|-----|-----|-----|
| Bob   | 42   | Dallas    |
| Mary  | 37   | El Paso   |
```

```

name | age | city
-----|-----|-----
Bob   | 42   | Dallas
Mary  | 37   | El Paso

```

Left-align the text of cells in a column by starting the alignment cell with `:`. The first column is left-aligned in this example:

```

| name | age | city   |
|:-----|-----|-----|
| Bob   | 42   | Dallas |
| Mary  | 37   | El Paso |

```

```

name | age | city
:-----|-----|-----
Bob   | 42   | Dallas
Mary  | 37   | El Paso

```

Right-align the text of cells in a column by ending the alignment cell with `:`. The first column is right-aligned in this example:

```

| name | age | city   |
|-----:|-----|-----|
| Bob   | 42   | Dallas |
| Mary  | 37   | El Paso |

```

```

name | age | city
-----:|-----|-----
Bob   | 42   | Dallas
Mary  | 37   | El Paso

```

Center-align the text of cells in a column by starting and ending the alignment cell with `:`. The first column is center-aligned in this example:

```
| name | age | city |
|:----:|----:|-----|
| Bob  | 42  | Dallas |
| Mary | 37  | El Paso |
```

```
name | age | city
:----:|----:|-----
Bob  | 42  | Dallas
Mary | 37  | El Paso
```

Each column gets its own alignment. Mix them together as needed.

```
| name | age | city |
|:-----|:---:|-----:|
| Bob  | 42  | Dallas |
| Mary | 37  | El Paso |
```

```
name | age | city
:-----|:---:|-----:
Bob  | 42  | Dallas
Mary | 37  | El Paso
```

## Markdown In Tables

Inline Markdown elements, like bold, italic, and even inline HTML, can be used with cell text content.



```
| name      | age | city      |
|-----|-----|-----|
| **Bob**   | 42  | Dallas    |
| **Mary**  | 37  | El Paso   |
```

```
name      | age | city
-----|-----|-----
**Bob**   | 42  | Dallas
**Mary**  | 37  | El Paso
```

## Markdown++

A custom Table Style can be given to a Table using a Markdown++ style tag on the line directly above the Table.

```
<!--style:CustomTable-->

| name | age | city      |
|-----|-----|-----|
| Bob   | 42  | Dallas    |
| Mary  | 37  | El Paso   |
```

```
<!--style:CustomTable-->

name | age | city
-----|-----|-----

Bob   | 42  | Dallas
Mary  | 37  | El Paso
```

## Content in Cells

Inline text content can be customized using the inline tag convention.

```
| name | age | city |
|-----|-----|
| <!--style:CustomText-->*Bob* | 42 | Dallas |
| <!--style:CustomText-->*Mary* | 37 | El Paso |
```

```
name | age | city
-----|-----|
<!--style:CustomText-->*Bob* | 42 | Dallas
<!--style:CustomText-->*Mary* | 37 | El Paso
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Style Behavior

In order to style a Table and its cells in detail, a few different styles are needed in ePublisher. A Table gets 3 styles when ePublisher detects one in a document.

The example below populates the Style Designer with 1 Table Style called **Table**, and 2 Paragraph Styles: **Table Cell Head**, and **Table Cell Body** when scanned into ePublisher.

```
> # Heading 1 element inside a blockquote
>
> This is a Paragraph element inside of a blockquote.
>
```

### Table Style

The Table Style is the first style that ePublisher adds to the Style Designer when a table is detected in a source document. The default name is **Table**, but could also be a custom name if the style tag syntax is used on the table.

This style applies table-specific style rules to the entire table. It is the only style in Markdown++ that creates an entry in the **Table Styles** area in the Style Designer.

## Customizing the Table Style

By adding a Markdown++ custom style tag, the Table Style name can be changed. The examples below change the Table Style name to **CustomTable**:

```
<!--style:CustomTable-->

| name | age | city      |
|-----|-----|-----|
| Bob   | 42  | Dallas    |
| Mary  | 37  | El Paso   |
```

```
<!--style:CustomTable-->

name | age | city
-----|-----|-----
Bob   | 42  | Dallas
Mary  | 37  | El Paso
```

## Header & Body Cell Styles

Every cell on a header row gets a Header Cell Style. Each cell on body rows get a Body Cell Style as well. To determine the Header Cell Style's name, ePublisher takes the Table Style and adds `Cell Head` to the end for Header Cell Styles, and `Cell Body` to the end for Body Cell Styles. The default names are **Table Cell Head** and **Table Cell Body**, but these will also be customized if the Table Style has a custom name.

### Customizing Header & Body Cell Styles

By adding a Markdown++ custom style tag, the Header & Body Cell Style names can be changed. The example below changes the style names to **CustomTable Cell Head** and **CustomTable Cell Body** because the Table Style has been given the custom style name **CustomTable**:

```
<!--style:CustomTable-->

| name | age | city      |
|-----|-----|-----|
| Bob   | 42  | Dallas   |
| Mary  | 37  | El Paso  |
```

```
<!--style:CustomTable-->

name | age | city
-----|-----|-----

Bob   | 42  | Dallas
Mary  | 37  | El Paso
```

## Default Style Properties

Style Type: **Table, Paragraph**

Style Name: **Table, Table Cell Head, Table Cell Body**

### Table

Property	Value
border top color	#222222
border top style	solid
border top width	1px
border right color	#222222
border right style	solid
border right width	1px
border bottom color	#222222
border bottom style	solid
border bottom width	1px
border left color	#222222
border left style	solid
border left width	1px

## Table Cell Head

Property	Value
font family	Arial
font size	11pt
font weight	bold
padding top	6pt
padding right	6pt
padding bottom	6pt
padding left	6pt

## Table Cell Body

Property	Value
font family	Arial
font size	11pt
padding top	6pt
padding right	6pt
padding bottom	6pt
padding left	6pt

If a custom style name is assigned to a Table, the style names will still inherit all of the listed default style information.

## Blockquotes

Blockquotes are unique block-level elements that can contain other block-level elements. They have a diverse set of usages due to this, such as capturing a sequence of conversation, or being a container for an important presentation of concepts.

### Syntax

Blockquotes are created by starting a line with the `>` character. A space between text content and the `>` character is optional, but recommended. Any Markdown or Markdown++ convention is acceptable as text content inside of Blockquotes.

### Basics

A basic Blockquote containing a single Paragraph

```
> A paragraph inside a blockquote.
```

### Markdown In Blockquotes

Other Markdown elements, like headings and lists, can be used inside Blockquotes. Use the same spacing and indentation rules as usual when inside Blockquotes.

```
> ### How to Publish Content with ePublisher  
  
>  
  
> Here's some steps to publish your content with ePublisher.  
  
>  
  
> 1. Open ePublisher  
> 2. Add source documents  
> 3. Select Format  
> 4. Click Generate All
```

## Nested Blockquotes

Other Blockquotes can also be nested inside of Blockquotes, and so on.

```
> First level blockquote  
  
>  
  
> > Second level nested blockquote.  
  
> >  
  
> > > Third level nested blockquote.  
  
> > >
```

## Markdown++

A custom Paragraph Style can be given to a Blockquote using a Markdown++ style tag on the line directly above the Blockquote.

```
<!--style:CustomBlockquote-->  
  
> A customized blockquote, style name "CustomBlockquote"
```

## Customizing Nested Blockquotes

Nested Blockquotes can be customized as well.



```

> A default blockquote, style name "Blockquote"

>

> <!--style:CustomBlockquote-->

> > A customized blockquote, style name "CustomBlockquote"

> >

```

## Default Until Customized

Nested Blockquotes are treated as standalone; they will not inherit the outermost style name if customized.

```

<!--style:CustomBlockquote-->

> A customized blockquote, style name "CustomBlockquote"

>

> > A default blockquote, style name "Blockquote"

> >

```

Add a style tag to each blockquote individually for consistency with custom styles.

```

<!--style:CustomBlockquote-->

> A customized blockquote, style name "CustomBlockquote"

>

> <!--style:CustomBlockquote-->

> > A customized blockquote, style name "CustomBlockquote"

> >

```

## Markdown in Blockquotes

The tagging convention can be used for other Markdown elements inside Blockquotes. These style names will inherit the Blockquote's style name as a prefix. See [Nested Styles](#) for more info.

```
> <!--style:CustomParagraph-->

> A customized paragraph, style name "Blockquote CustomParagraph"

>

> <!--style:CustomList-->

> - an unordered list

> - customized

> - style name "Blockquote CustomList"

>
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Style Behavior

Blockquotes are considered containers; they *contain* other block-level elements, like Paragraphs, Lists, and Tables. Because of this, ePublisher creates a number of different styles when it detects blockquotes in source documents.

### Blockquote Style

The Blockquote Style is the first style that ePublisher adds to the Style Designer when a blockquote is detected in a source document. The default name is **Blockquote**, but could also be a custom name if the style tag syntax is used on the blockquote.

This style applies to the container area surrounding the blockquote's content. It's style rules can also apply to nested content, if the same rule isn't already applied on the nested style.

### Customizing the Blockquote Style

By adding a Markdown++ custom style tag, the Blockquote Style name can be changed. The example below changes the Blockquote Style name to **CustomBlockquote**:

```
<!--style:CustomBlockquote-->
> This is a custom named blockquote.
>
```

## Nested Styles

Nested content inside of Blockquotes also get a new style name. To determine the Nested Style's name, take the Blockquote Style and add the style name of the nested content to the end, separated by a space.

The example below populates the Style Designer with 3 Paragraph Styles: **Blockquote** (the Blockquote Style), **Blockquote Heading 1**, and **Blockquote Paragraph** when scanned into ePubublisher.

```
> # Heading 1 element inside a blockquote
>
> This is a Paragraph element inside of a blockquote.
>
```

## Customizing Nested Styles

By adding a Markdown++ custom style tag, the Nested Style name can be changed. The example below changes the Nested Style name to **Blockquote CustomParagraph**:

```
> <!--style:CustomParagraph-->
> This is a custom paragraph.
>
```

Custom Style Names can be used on both the blockquote and nested content simultaneously. This example creates the style names **CustomBQ**, and **CustomBQ CustomParagraph**:

```
<!--style:CustomBQ-->  
> <!--style:CustomParagraph-->  
> This is a custom paragraph inside a blockquote.  
>
```

## Default Style Properties

Style Type: **Paragraph**  
Style Name: **Blockquote**

Property	Value
background color	#efefef
border left style	solid
border left color	#DFE2E5
border left width	3pt
padding top	12pt
padding right	12pt
padding bottom	12pt
padding left	12pt

If a custom style name is assigned to a Blockquote, that style name will still inherit all of the listed default style information.

## Code Fences

A Code Fence preserves all text content it encapsulates and presents it exactly how it was written. Code Fences are useful for presenting information that needs to be written explicitly, like examples of code. They can provide users with content that can be copied and used for their own purposes, too.

### Syntax

Code Fences are created in three steps:

1. Start with a line containing ````` or `~~~`.
2. A following line or lines of text content.
3. End with a line containing ````` or `~~~`, matching the starting line.

### Basics

A simple example using ````` tags. Any amount of text can be written between the two tags, as long as the tags match and are written correctly.

```
```  
  
function addTwoNumbers(num1, num2) {  
  
    return num1 + num2;  
  
}  
  
```
```

Code Fences can be created using `~~~`, too.

```
~~~  
  
function addTwoNumbers(num1, num2) {  
  
    return num1 + num2;  
  
}  
  
~~~
```

## No Parsing in Code Fences

Markdown written inside of Code Fences will render as plain text.

```
```  
  
# Heading 1 in Plain Text  
  
```
```

HTML will also render as plain text when written inside Code Fences.

```
```  
  
<p>HTML in plain text</p>  
  
```
```

## Markdown++

A custom Paragraph Style can be given to a Code Fence using a Markdown++ style tag on the line directly above the Code Fence.

```
<!--style:CustomCodeFence-->
```

function addTwoNumbers(num1, num2) {

    return num1 + num2;

}

```
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Default Style Properties

Style Type: **Paragraph**

Default Style Name: **Code Fence**

Property	Value
background color	#efefef
font family	Consolas
font size	11pt
margin top	6pt
margin bottom	6pt
padding top	12pt
padding right	12pt
padding bottom	12pt
padding left	12pt
overflow	auto
white space	pre

If a custom style name is assigned to a Code Fence, that style name will still inherit all of the listed default style information.

## Code Blocks

A Code Block preserves all text content it encapsulates and presents it exactly how it was written. Code Blocks are useful for presenting information that needs to be written explicitly, like examples of code. They can provide users with content that can be copied and used for their own purposes, too.

### Syntax

Code Blocks are created by adding 4 spaces before text content. A Code Block can consist of one or more lines created in this manner.

### Basics

Starting a line with 4 spaces will create a basic Code Block.

```
var firstName, lastName;
```



## Multi-Line Code Blocks

Multiple lines can be used; start all lines with at least 4 spaces.

```
var firstName, lastName;

    firstName = "John";

    lastName = "Doe";
```

### Space is Preserved.

Any spaces after the first 4 will be used as indentation for the content of the Code Block.

```
function addTwoNumbers(num1, num2) {

    return num1 + num2;

}
```

### No Parsing in Code Blocks

Markdown written inside of Code Blocks will render as plain text.

```
# Heading 1 in Plain Text
```

HTML will also render as plain text when written inside Code Blocks.

```
<p>HTML in plain text</p>
```

## Markdown++

A custom Paragraph Style can be given to a Code Block using a Markdown++ style tag on the line directly above the Code Block.

```
<!--style:CustomCodeBlock-->

function addTwoNumbers(num1, num2) {

    return num1 + num2;

}
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

# ePublisher Style Information

## Default Style Properties

Style Type: **Paragraph**

Style Name: **Code Block**

Property	Value
background color	#efefef
font family	Consolas
font size	11pt
margin top	6pt
margin bottom	6pt
padding top	12pt
padding right	12pt
padding bottom	12pt
padding left	12pt
overflow	auto
white space	pre

If a custom style name is assigned to a Code Block, that style name will still inherit all of the listed default style information.

## Horizontal Rules

A Horizontal Rule provides a visual separation between sections of content. They're useful to separate unrelated ideas on a single page.

### Syntax

A Horizontal Rule is created by using at least 3 `-`, `_`, or `*` characters. These should be the only characters on the line, but any combination of them is acceptable.

### Basics

A simple Horizontal Rule using `-` characters.

```
---
```

An example using `*` characters.

\*\*\*

And one with ☐ characters.

—

### 3 or More Characters

More than 3 characters can be used, if desired.

-----

### Spaces OK

Spaces are acceptable between the characters.

- - - - -

### Mixed Characters

Combinations of the 3 characters is fine to use as well.

-\* \*\_\* \*\_\* \*\_\*

-\_-\_-\_-\_-

\* \_ \* \_ \* \_ \*

### Markdown++

A custom Paragraph Style can be given to a Horizontal Rule using a Markdown++ style tag on the line directly above the Horizontal Rule.

```
<!--style:CustomHR-->
```

---

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Default Style Properties

Style Type: **Paragraph**

Style Name: **Horizontal Rule**

Property	Value
border top color	#222222
border top style	inset
border top width	1px
border right color	#222222
border right style	inset
border right width	1px
border bottom color	#222222
border bottom style	inset
border bottom width	1px
border left color	#222222
border left style	inset
border left width	1px
display	block
margin top	6pt
margin bottom	6pt
tag	hr

If a custom style name is assigned to a Horizontal Rule, that style name will still inherit all of the listed default style information.

## Block HTML

The common markup language for web technology, HTML, can be used in Markdown documents on the block level. Refer to [W3Schools' HTML Tutorial](#) to learn more about how to write and use HTML.

### Syntax

Block HTML is created by writing a valid HTML fragment on a line or set of lines. HTML syntax must be the first thing on the line to be considered Block HTML.

## Basics

Simple Block HTML using a `p` element.

```
<p>A simple paragraph element.</p>
```

## Multi-Line HTML

HTML can span multiple lines. Keep it compact. An empty line will break the fragment in two, so it is best used to separate the fragment from other content.

```
<table>

  <tr>

    <th>Name</th>

    <th>Age</th>

    <th>Country</th>

  </tr>

  <tr>

    <td>John Doe</td>

    <td>35</td>

    <td>USA</td>

  </tr>

  <tr>

    <td>Jane Doe</td>

    <td>32</td>

    <td>USA</td>

  </tr>

</table>
```

## No Markdown in Block HTML

Markdown syntax can't be used inside of Block HTML. The entire HTML fragment is passed straight to the output as-is.

```
<p>No Markdown here.</p>
```

## Markdown++

A custom Paragraph Style can be given to Block HTML using a Markdown++ style tag on the line directly above the Block HTML.

```
<!--style:CustomHTML-->

<p>HTML block given the style name "CustomHTML"</p>
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Style Behavior

All HTML fragments are wrapped in a container element, which is given a style name. The default name is **HTML**, but can also be a custom name if the style tag is used directly above an HTML fragment.

HTML is unavailable for publishing in PDF or PDF XSL-FO output due to incompatibility with those technologies. ePublisher will remove any HTML content it detects before generating PDF output.

### Default Style Properties

Style Type: **Paragraph**

Style Name: **HTML**

Property	Value
display	block
overflow	auto

If a custom style name is assigned to a Block HTML, that style name will still inherit all of the listed default style information.

## Bold, Italic, Strikethrough, Code

Inline text can be styled to put emphasis or formatting on certain phrases. Markdown offers wrappers for Bold, Italic, Strikethrough, and Code.

### Syntax

Bold text is created by wrapping a set of text with a pair of either `**` or `__` characters.

Italic text is created by wrapping text with a pair of either `*` or `_`.

Strikethrough text is created by wrapping text between a pair of `~~` characters.

Code spans are created by wrapping text between a pair of ``` characters.

### Basics

Two simple examples for Bold text. Notice either `*` or `_` can be used, but there must be two on each side of the wrap. The start and end characters must also match.

```
Here's **bold** and here's also __bold__.
```

Italic text is written similarly, using one `*` or `_` instead of two.

```
Here's *italic* and here's also _italic_.
```

Same rules apply to Strikethrough text, using `~~`.

```
Using ~~strikethrough~~ text.
```

Code spans follow the same rules, too.

```
Defining a `technical term`.
```



## Mixing Styles of Text

Combinations of these can be used together. Make sure the innermost pair of tags is closed before closing an outer pair. Using unlike characters for different pairs helps with readability. (Using `*` for bold, `_` for italic, etc.)

```
We can write bold and _italic_.
```

## Spanning Multiple Lines

Inline text decorators can span multiple lines, as long as there are no empty lines between the start and end tags.

```
Writing a sentence that has  
  
bold text across lines.
```

## Markdown++

A custom Character Style can be given to Inline Text using a Markdown++ style tag directly before the start tag of the Inline Text.

```
Styling <!--style:CustomBold-->inline text. Style name  
"CustomBold"
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

# ePublisher Style Information

## Default Style Properties

Style Type: **Character**

Style Name: **Bold, Italic, Strikethrough, Code**

**Bold**

Property	Value
font weight	bold

## **Italic**

Property	Value
font style	italic

## Strikethrough

Property	Value
text-decoration	line-through

## Code

Property	Value
background color	#efefef
font family	Consolas
white space	pre

If a custom style name is assigned to Inline HTML, that style name will still inherit all of the listed default style information.

## Links

Links are an inline Markdown convention used to connect users to other locations and resources in a set of information.

### Syntax

Link syntax looks interesting, but is simple enough once written a few times. Write the link's displayed text in between `[` and `]` characters, and directly next to it write the link's URL between `(` and `)` characters. Optionally, a title can be given to the Link, written next to the link URL, separated by a space and wrapped in `"` characters.

### Basics

A basic Link example.

```
[Link Text] (path/to/my_doc.md)
```

Titles are optional. Keep the URL and title separate with a space. Wrap the title in `"` characters.

```
[Link Text] (path/to/my_doc.md "Link Title")
```

Links can be the only thing on a line or mixed in anywhere inline text can go.

```
To see more, follow the [Link] (path/to/my_doc.md) .
```

Relative paths, absolute paths, web links, and [Aliases](#) are all valid path values.

```
[Link Text](../my_doc.md)
```

```
[Link Text](D:/Markdown/Docs/my_doc.md)
```

```
[Link Text](https://www.webworks.com)
```

```
[Link Text](#my-doc)
```

## Using Link References

Links can make use of [Link References](#) to simplify URL management for documents with many different link paths.

```
[Link Text][0]
```

```
[0]: my_image.png
```

Titles are also available and written the same way using Link References.

```
[Link Text][0]
```

```
[0]: my_image.png "Link Title"
```

## Markdown++

A custom Character Style can be given to an Image using a Markdown++ style tag immediately before the Link syntax.

```
<!--style:CustomLink-->[Link Text](path/to/my_doc.md)
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## Link Behavior

Links in ePubliker have a variety of ways to connect to other resources in a publication. What they connect to depends on what is used as a path value. All path values inside links also apply to path values in Link References.

## Web Links

Write a fully qualified web URL in the path area to link to an external resource. Make sure to start the URL with `http://` or `https://`.

```
[WebWorks Website] (https://www.webworks.com)
```

## Link to Other Documents

Write the file path for the intended file in the path area to link to another document. Relative paths need to resolve from the file the link is written in. Absolute paths can also be used.

```
[link text] (path/to/my_doc.md)
```

```
[link text] (C:/Users/me/path/to/my_doc.md)
```

## Link to Topics in Other Documents

To link to a specific section in a document, write the file path followed immediately with the alias for the topic. This can be either a [Heading Alias](#) or a [Custom Alias](#). Relative paths need to resolve from the file the link is written in. Absolute paths can also be used.

The examples link to the alias `#my-alias` in `my_doc.md`.

```
[link text] (path/to/my_doc.md#my-alias)
```

```
[link text] (C:/Users/me/path/to/my_doc.md#my-alias)
```

## Link to Topics in Same Document

Use an alias by itself to link to a topic in the current document. This can be either a [Heading Alias](#) or a [Custom Alias](#).

The example links to the alias `#my-alias` in the current document.

```
[link text] (#my-alias)
```

## ePublisher Style Information

### Default Style Properties

Style Type: **Character**

Style Name: **Link**

Property	Value
text decoration	underline
color	#0078d7

If a custom style name is assigned to a Link, that style name will still inherit all of the listed default style information.

## Images

Images are an inline Markdown convention used to display graphics in a document.

### Syntax

The image syntax is identical to the [Link](#) syntax, with the addition of a `!` character at the beginning. Alt text is written between `!` and `]` characters. Inside of `(` and `)`, the URL path to the image should be written, then, optionally, a title wrapped in `"` characters.

### Basics

A basic Image example.

```
![alt text](path/to/my_image.png)
```

Titles are optional. Keep the URL and title separate with a space.

```
![alt text](path/to/my_image.png "Image Title")
```

Images can be the only thing on a line or mixed in anywhere inline text can go.

```
Images can go anywhere text can: ![alt text](path/to/my_image.png)
```

Relative paths and absolute paths can both be used.

```
![alt text](../my_image.png)
```

```
![alt text](D:/Images/my_image.png)
```

### Using Link References



Images can also make use of [Link References][md-link-reference] in the same way Links do.

```
![alt text][0]
```

```
[0]: my_image.png
```

Titles are also available and written the same way using Link References.

```
![alt text][0]
```

```
[0]: my_image.png "Image Title"
```

## Markdown++

A custom Graphic Style can be given to an Image using a Markdown++ style tag immediately before the Image syntax.

```
<!--style:CustomImage-->![alt text] (path/to/my_image.png)
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

# ePublisher Style Information

## Default Style Properties

Style Type: **Graphic**  
Style Name: **Image**

If a custom style name is assigned to an Image, that style name will still inherit all of the listed default style information.

## Link References

Link References accompany Links and Images, and are used to keep path values in a separated location from text content. They're useful for readability because they simplify links and images in inline text, and can be written in a standalone location for editing en masse.

This section requires familiarity with [Links](#) and [Images](#) to teach referencing concepts.

## Syntax

A Link Reference must be the only thing on a line. The **link key** is written between `[` and `]`. The URL path is written next, separated by a space. Optionally, a title can be written after the URL, separated by a space.

Once a Link Reference has been written, the **link key** from it can be used with a Link or Image. Replace the Link/Image's parenthesis `()` section with the link key between `[` and `]`.

## Basics

A Link Reference example used with an accompanying Link. The Link is written first and makes use of the link key, in this example `[0]`.

```
[Link Text][0]
```

```
[0]: path/to/my_doc.md
```

Titles are optional. Keep the URL and title separate with a space.

```
[Link Text][0]
```

```
[0]: path/to/my_doc.md "Link Title"
```

A Link Reference example with an accompanying Image.

```
![alt text][0]
```

```
[0]: path/to/my_image.png
```

Make sure to keep the Reference on it's own line. The Link or Image can be used anywhere text is allowed, though.

```
For more info, check the [Link][0].
```

```
[0]: my_doc.md
```

## Use Unique Values for Link Keys

Any text will work for the link key, but something unique that can be searched for will help in the authoring process. Link keys must be one-of-a-kind as well. In the case of overlapping link keys, the last link key written will be the accepted one.

```
[wwdoc_0001]: my_doc.md
```

```
[wwdoc_0002]: doc2.md
```

```
[wwdoc_0003]: doc3.md
```

```
[wwimg_0001]: img1.png
```

```
[wwimg_0002]: img2.png
```

```
[wwimg_0003]: img3.png
```

## Inline HTML

The common markup language for web technology, HTML, can be used in Markdown documents mixed with text and other inline elements. Refer to [W3Schools' HTML Tutorial](#) to learn more about how to write and use HTML.

### Syntax

Inline HTML is created by writing a valid HTML fragment in an area where other inline content exists.

### Basics

Simple Inline HTML using a `strong` element.

```
Write words with <strong>bold</strong> emphasis.
```

### Markdown and Inline HTML

Markdown syntax can be mixed with Inline HTML.

```
We can <span>write bold text</span>.
```

### Markdown++

A custom Character Style can be given to Inline HTML using a Markdown++ style tag directly before the Inline HTML.

```
Styling <!--style:CustomHTML--><span>inline HTML. Style name  
"CustomHTML".</span>
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Style Behavior

All HTML fragments are wrapped in a container element, which is given a style name. The default name is **HTML**, but can also be a custom name if the style tag is used directly before an HTML fragment.

HTML is unavailable for publishing in PDF or PDF XSL-FO output due to incompatibility with those technologies. ePublisher will remove any HTML content it detects before generating PDF output.

### Default Style Properties

Style Type: **Character**

Style Name: **HTML**

If a custom style name is assigned to Inline HTML, that style name will still inherit all of the listed default style information.

## Learning Markdown++

This section will detail the features of Markdown++, how to write them, and how to use them in ePublisher. For quick reference material, see the [Markdown Cheat Sheet](#).

Markdown++ is a superset of Markdown, meaning that any convention available in Markdown also applies to Markdown++. [Learn about Markdown](#) before reading this section.

### Quick Links

[Markdown++ Basics](#)

[Multiline Tables](#)

[Custom Styles](#)

[Aliases](#)

[Markers](#)

[Conditional Text](#)

[File Includes](#)

[Variables](#)

## Markdown++ Basics

Markdown++ is a superset of Markdown. Because of this, *all Markdown files are also Markdown++ files*. Any tools used for Markdown also work well for Markdown++.

Filling out Markdown with a full designing & publishing experience, while also maintaining readability, is a major design goal of Markdown++. Another goal is to preserve the integrity of rendering and previews across the many Markdown tools out there.

Markdown++ uses the HTML Comment tag with a set of commands inside them for most of its features. Using these enables Markdown++ syntax to be transparent when documents are rendered or previewed, and aids in quick learning by using a well-established pattern.

Learning the HTML Comment tag opens the door to learning most of the features Markdown++ offers.

## Syntax

Write an HTML Comment by starting with `<!--` and ending with `-->`. Any text can be written between these two patterns. Keeping the entire comment on a single line is required by Markdown++.

Any unrecognized text inside of a comment gets treated as a regular HTML comment and carries through to the output.

## Basics

A simple comment tag with a [Custom Style Name](#) command. Keep the tag on a single line.

```
<!--style:CustomStyle-->
```

Apply commands to block-level elements by adding the tag to the line directly above the block element. The style **CustomParagraph** is added to a paragraph below.

```
<!-- style:CustomParagraph -->
```

A customized paragraph, named "CustomParagraph".

Apply commands to inline elements by adding the tag directly before the inline syntax. Don't put space between the comment tag and the inline syntax. The style **CustomBold** is added to bold text below.

```
Customizing some <!--style:CustomBold-->**bold text**.
```

## Whitespace OK

In general, it is safe to include any amount of whitespace *between the comment tags*. Use it as necessary for readability.

```
<!-- style: CustomStyle -->
```

## Multiple Commands

Any number of commands can be put inside the comment. Separate the commands with a `;` character. This example applies two commands to a Heading 1: a Custom Style Name, and a [Custom Alias](#).

```
<!-- style:CustomHeading1 ; #custom-heading1 -->

# Heading 1
```

## Start & End Tags

Some features, like [Conditional Text](#), require start & end tags. The example wraps condition tags around content meant only for printed publications.

```
<!--condition:print_only-->

## Print Only

This content is meant for print only.

<!--/condition-->
```

# Multiline Tables in Markdown++

Multiline tables in Markdown++ provide an enhanced way to organize detailed content across multiple lines, allowing for more readable and flexible structures. Unlike standard tables, multiline tables allow rows to span multiple lines by using a special tag and structured spacing.

## Introduction

Multiline tables are very similar to regular Markdown tables, but they are more flexible when it comes to the table structure. In Markdown++, multiline tables are indicated by placing a `<!-- multiline -->` tag directly above the table. Additionally, a new row is created by adding a row of cells that are either empty or contain only whitespace.

A key feature of multiline tables is that you can include full block Markdown elements within cells, such as lists, code blocks, or paragraphs. This makes multiline tables particularly useful when content within a cell becomes too detailed to fit conveniently on a single line. By utilizing multiline tables, writers can keep content organized and easier to edit.

## Syntax

Multiline tables in Markdown++ consist of the same main elements as standard tables:

- A **header row**, which contains header cell content separated by `|` characters.
- An **alignment row**, which indicates the alignment of the body cells' text. Each cell in this row contains at least 3 `-` characters, and an optional `:` character to indicate alignment. Each cell is separated by a `|` character.
  - Default alignment only uses `-` characters; 3 or more.
  - Left align the column by starting the cell with `:` and filling in the rest with `-` characters; 3 or more.
  - Right align the column by starting the cell with 3 or more `-` characters, ending with a `:` character.
  - Center align the column by starting and ending the cell with `:` characters. Put `-` characters between them; 3 or more.
- 1 or more **body rows**, that contain body cell content separated by `|` characters.

The key difference is the `<!-- multiline -->` tag and the ability to use blank rows to create new table rows, along with the capability to write full block Markdown inside cells.

## Multiline Tag

To use a multiline table, add the following tag directly above your table:

```
<!-- multiline -->
```

This tag tells Markdown++ to treat the following table as a multiline table.

## Creating New Rows

To create a new row, simply add a row of cells that are either empty or contain only whitespace. This indicates to Markdown++ that the next line of cells should be treated as part of a new row.

## Basics

Below is an example of a multiline table:

```
<!-- multiline -->

| name | details |
|-----|-----|
| Bob  | Lives in Dallas. |
|      | - Enjoys cycling |
|      | - Loves cooking  |
|      |                  |
| Mary | Lives in El Paso. |
|      | - Works as a teacher |
|      | - Likes painting  |
```

In this example, the empty row acts as a separator, creating a break before the next row. Notice how detailed information, including lists, is included within a cell.

You can also use multiline tables without wrapping `<!--` characters:

```
<!-- multiline -->

name | details
-----|-----

Bob  | Lives in Dallas.
      | - Enjoys cycling
      | - Loves cooking
      |
Mary | Lives in El Paso.
      | - Works as a teacher
      | - Likes painting
```

## Alignment in Multiline Tables



Just like with standard tables, multiline tables can have different alignments for each column.

Left-align the text of cells in a column by starting the alignment cell with `:`:

```
<!-- multiline -->

| name | details |
|:-----|-----|
| Bob | Lives in Dallas. |
|      | - Enjoys cycling |
|      | - Loves cooking  |
|      |                  |
| Mary | Lives in El Paso. |
|      | - Works as a teacher |
|      | - Likes painting  |
```

Center-align the text of cells in a column by starting and ending the alignment cell with `:`:

```
<!-- multiline -->

| name | details |
|:-----:|-----|
| Bob | Lives in Dallas. |
|      | - Enjoys cycling |
|      | - Loves cooking  |
|      |                  |
| Mary | Lives in El Paso. |
|      | - Works as a teacher |
|      | - Likes painting  |
```

## Markdown In Multiline Tables

Multiline tables can also include full Markdown blocks, such as code blocks or lists:

```
<!-- multiline -->

| name          | hobbies                      |
|-----|-----|
| **Bob**      | - Biking                    |
|              | - Cooking                  |
|              | - Reading                  |
|              |                             |
| **Mary**     | Here is a code example:    |
|              |                             |
|              | ```                        |
|              | function greet() {         |
|              |     console.log("Hi");     |
|              | }                           |
|              | ```                        |
```

Including full Markdown blocks like code and lists helps emphasize complex content within cells, making tables more versatile.

## Markdown++ Custom Styles

A custom Table Style can be given to a multiline Table using a Markdown++ style tag directly above the multiline tag. Additionally, multiple Markdown++ commands can be combined using `!>`:

```
<!--style:CustomTable; multiline -->

| name | description |
|-----|-----|
| Bob | - Loves biking |
|      | - Enjoys programming |
|      | |
| Mary | A passionate teacher. |
|      | - Loves painting |
|      | - Enjoys hiking |
```

## Content in Cells

Inline text content can be further customized using the inline tag convention, and multiple commands can be used in a single tag:

```
<!-- multiline -->

| name | age | city |
|-----|-----|
| <!--style:CustomText-->*Bob* | 42 | Dallas |
| | | |
| <!--style:CustomText-->*Mary* | 37 | El Paso |
```

To learn more about Markdown++ tagging, see [Learning Markdown++](#).

## ePublisher Style Information

### Style Behavior

In order to style a multiline Table and its cells in detail, the same styles are needed as with standard tables in ePublisher. A multiline table gets 3 styles when detected in a document: **Table**, **Table Cell Head**, and **Table Cell Body**.

### Table Style

The Table Style is the primary style that ePublisher adds when a multiline table is detected. The default name is **Table**, but this can be customized using a Markdown++ style tag.

## Customizing the Table Style

By adding a Markdown++ custom style tag, the Table Style name can be changed:

```
<!--style:CustomTable; multiline -->

| name | age | city      |
|-----|-----|-----|
| Bob   | 42  | Dallas    |
|       |     |           |
| Mary  | 37  | El Paso   |
```

## Header & Body Cell Styles

Header and body cell styles function in the same way for multiline tables as they do for standard tables. Each header cell gets a **Table Cell Head** style, and each body cell gets a **Table Cell Body** style. These styles can also be customized if the Table Style is given a custom name:

```
<!--style:CustomTable; multiline -->

| name | age | city      |
|-----|-----|-----|
| Bob   | 42  | Dallas    |
|       |     |           |
| Mary  | 37  | El Paso   |
```

## Default Style Properties

Style Type: **Table, Paragraph**

Style Name: **Table, Table Cell Head, Table Cell Body**

### Table

Property	Value
border top color	#222222
border top style	solid
border top width	1px
border right color	#222222
border right style	solid
border right width	1px
border bottom color	#222222
border bottom style	solid
border bottom width	1px
border left color	#222222
border left style	solid
border left width	1px

## Table Cell Head

Property	Value
font family	Arial
font size	11pt
font weight	bold
padding top	6pt
padding right	6pt
padding bottom	6pt
padding left	6pt

## Table Cell Body

Property	Value
font family	Arial
font size	11pt
padding top	6pt
padding right	6pt
padding bottom	6pt
padding left	6pt

If a custom style name is assigned to a multiline Table, the style names will still inherit all of the listed default style information.

## Custom Styles

The Custom Style command overrides the default Style Name of a Markdown element with a user-defined Style Name. Using this feature enables a virtually limitless amount of styles for designing & publishing in ePublisher.

### Syntax

The Custom Style command is created by writing `style:` followed by the name of the intended style.

### Basics

A basic Custom Style command applied to a Paragraph.

```
<!--style:CustomParagraph-->
```

```
This paragraph has it's style name customized to "CustomParagraph".
```

Put the tag on the line above any block-level element to customize it. Make sure there are no empty lines between the tag and the block element. The tag needs to be the only thing on it's line.

```
<!--style:CustomHeading1-->
```

```
# This Heading has been renamed to "CustomHeading1".
```

For Custom Styles on inline text content, put the tag directly before the starting syntax. No space should be put between the tag and the inline syntax. A string of bold text is customized with the Style Name **CustomBold** below.

```
This paragraph has customized <!--style:CustomBold-->**bold text**.
```

## Mix with Other Commands

Custom Styles can be in the same comment tag with other commands. Separate them with a `;` character. A Custom Style and [Custom Alias](#) are written in the same tag below.

```
<!-- style:CustomStyle ; #custom-alias -->
```

## Custom Style Command Behavior

Through the Custom Style command, it is possible to get name entries for almost any type of style into ePublisher's Style Designer. How to do so varies based on style type.

### Custom Paragraph Style

Add the style tag to the line directly above a block-level element to give it a custom Paragraph Style. This applies to any block-level element, except for Tables.

```
<!--style:CustomParagraph-->
```

```
This is a custom paragraph called "CustomParagraph".
```

```
<!--style:CustomHeading1-->
```

```
# This is a custom paragraph called "CustomParagraph".
```

```
<!--style:UList-->
```

```
- custom list  
- unordered  
- called "CustomUList"
```



```
<!--styleCustomBlockquote-->
> This is a custom blockquote
>
```

```
<!--style:CustomHTML-->
<div>
  <p>This is customized HTML. Named "CustomHTML".</p>
</div>
```

## Custom Character Style

Add a style tag directly before inline syntax to create a custom Character Style. Remember, no space between the tag and the inline syntax. This applies to any inline syntax, except for Images.

```
This is customized <!--style:CustomBold-->**bold text**.
```

```
This is customized <!--style:CustomItalic-->*italic text*.
```

```
This is a customized <!--style:CustomLink-->[link] (my_doc.md) .
```

```
This is a customized <!--style:CustomHTML--><span>Inline HTML</span>.
```

## Custom Graphic Style

Add a style tag directly before image syntax to create a custom Graphic Style. Remember, no space between the tag and the image syntax.

```
<!--style:CustomImage-->![alt text] (my_image.png)
```

```
<!--style:CustomImage-->![alt text] [link_key]
```

## Custom Table Style

Like custom Paragraph Styles, add the tag to the line directly above Table syntax to give it a custom Table Style.

```
<!--style:CustomTable-->

| name | age | city      |
|-----|-----|-----|
| Bob   | 42  | Dallas  |
| Mary  | 37  | El Paso |
```

```
<!--style:CustomTable-->

name | age | city
-----|-----|-----

Bob   | 42  | Dallas
Mary  | 37  | El Paso
```

## Custom Page Style

Custom Page Styles need to be created through the [Custom Markers](#) command. Refer to the linked section for details.

```
<!--markers:{"PageStyle": "CustomPage"}-->

# Topic Heading
```

## Custom Marker Style

Custom Marker Styles need to be created through the [Custom Markers](#) command. Refer to the linked section for details.

```
<!--marker:Keywords="topic, heading, markers"-->

# Topic Heading
```

## Custom Aliases

Use a Custom Alias to give an element a unique pointer for linking to in other places in the publication. The Custom Alias is a powerful tool that can simplify link management and speed up authoring and editing.

## Syntax

Create a Custom Alias by starting with a single `#` character, followed any alphanumeric characters, `-`, or `_`. Space characters cannot be used in an alias; the Alias will cut off before the space.

## Basics

A basic Custom Alias applied to a Heading 1.

```
<!--#custom-alias-->

# Custom Aliased Heading
```

Inline syntax can be given an Alias as well.

```
This <!--#bold-keyword-->**bold text** has a custom alias.
```

## Mix with Other Commands

Custom Aliases can be in the same comment tag with other commands. Separate them with a `;` character. A [Custom Style](#) and Custom Alias are written in the same tag below.

```
<!-- style:CustomStyle ; #custom-alias -->
```

## Custom Alias Behavior

Custom Aliases create an entry in the document's WIF that enable linking to the element it was created with.

## Using a Custom Alias

The first step in using a Custom Alias is to create one by adding the Alias tag to the location that will be linked to. Below, the Custom Alias `#my-alias` is created, associated with a Heading 1.

```
<!--#my-alias-->

# A Topic Heading
```

After that, the Alias `#my-alias` can be used in the path value for [Links](#) and [Link References](#). Once the link is clicked, it will take readers to the location the Alias was created.

Refer to the linked sections for detailed instructions on using these elements.

Some quick examples for Links:

```
[link text] (#my-alias)
```

```
[link text] (my_doc.md#my-alias)
```

Some quick examples for Link References:

```
[link text][link_key]
```

```
[link_key]: #my-alias
```

```
[link text][link_key]
```

```
[link_key]: my_doc.md#my-alias
```

## Markers in Markdown++

Markers are pieces of metadata that can be inserted into a document to add different features or change the behavior of the publication. They are useful for adding keywords to improve search relevance, or for instructing ePublisher to pass text through to the output without processing.

### Syntax

The Markers command has two main formats. The preferred format is to start with `marker:`, followed by the key and value directly as `key="value"`. This simpler version is ideal when you need to write a single marker value.

For more advanced scenarios where multiple markers are required in one statement, you can use the `markers:` command, followed by a [JSON Object Literal](#). In this case, the marker names are written as keys and their respective values as values. Make sure all keys and values are wrapped in `"`, and separated by `:`. Multiple key/value pairs can be included, separated by `,`.

## Basics

A basic Markers command using the preferred compact version:

```
<!--marker:Keywords="webworks"-->

# About WebWorks
```

Adding multiple `marker:` statements in a single tag is also acceptable:

```
<!--marker:Keywords="webworks"; marker:Category="documentation"-->

# About WebWorks
```

Or, using the JSON format for multiple markers:

```
<!--markers:{"Keywords": "webworks", "Category": "documentation"}-->

# About WebWorks
```

The marker command is also available for inline-level syntax:

```
Add a custom <!--marker:Keywords="inline"-->**marker**.
```

Or using the JSON format for multiple values:

```
Add a custom <!--markers:{"Keywords": "inline, marker"}-->**marker**.
```

## Mix with Other Commands

Markers can be used in the same comment tag with other commands. Separate them with a `;` character. Below, a [Custom Style](#) and a marker are written in the same tag:

```
<!-- style:CustomStyle ; marker:Keywords="webworks" -->
```

Or using the JSON format for multiple markers:

```
<!-- style:CustomStyle ; markers:{"Keywords": "webworks"} -->
```

## Markers Behavior

Markers associate a piece of metadata with an element on the page. To learn more about ePubublisher's built-in markers and what they do, see the [Markers reference table](#).

## Using a Marker

To use a marker, first add it to the intended area of the content. The marker should be tagged to a content element, such as a paragraph or heading.

Below, a Keywords marker is tagged to a Heading 1 using both formats:

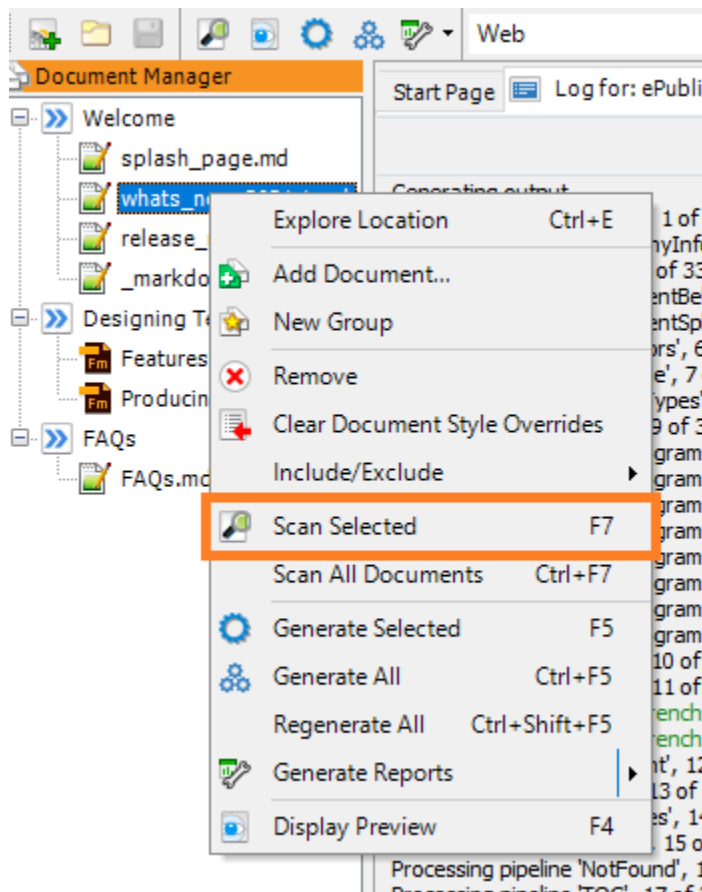
Using the preferred compact version:

```
<!--marker:Keywords="markers"-->  
  
# Using Markers in Source Content
```

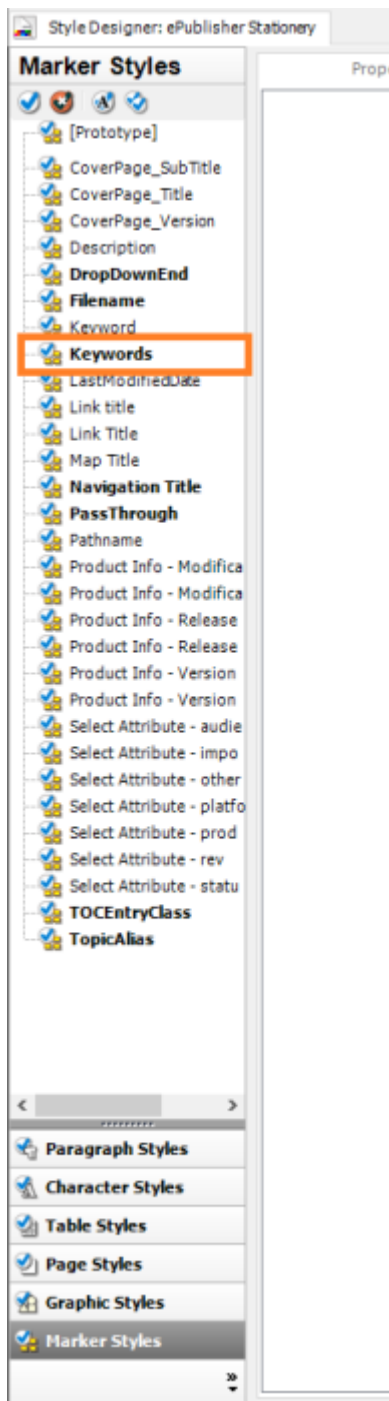
Using the JSON format for multiple values:

```
<!--markers:{"Keywords": "markers, content, create"}-->  
  
# Using Markers in Source Content
```

Next, scan the document in ePublisher. This will add the marker to the Marker Styles area of the Style Designer.



With the marker added to ePublisher, you can now generate output with the new keywords. This marker will improve search relevance in outputs with searching capabilities, such as [Reverb 2.0](#).



## Conditions

Conditions allow an author to create sections of content that are available in certain contexts, and excluded in others. They can be useful for the Edit/Review process, or to switch out sections meant only for print or the Web.



## Syntax

Conditional text is written between two HTML Comment Tags. In the first tag, write one or more conditions between `<!--condition:` and `-->`. The second closing tag is always written as `<!--/condition-->`. Between the tags, any valid Markdown + content can be written. Condition names must only use alphanumeric characters, `_`, and `-`. Do not use spaces in Condition names.

See [Condition Operators](#) for details on advanced conditional logic syntax.

## Basics

A Condition containing a single Paragraph, using the Condition `print_only`.

```
<!--condition:print_only-->

This paragraph is meant only for print.

<!--/condition-->
```

Conditions can contain multiple block-level elements.

```
<!--condition:print_only-->

# The Print Section

This sections is meant only for print.

It will not be visible if `print_only` is set to `Hidden`.

<!--/condition-->
```

Conditions can be used with inline content, too.

```
Go to the Section <!--condition:print_only-->on page 304<!--/
condition--> for more details.
```

## Operators

Complex logic can be used with Conditions using a set of [Operators](#). This block is hidden when `production` is set `Visible` using the `!` (logical NOT) operator.

```
<!--condition:!production-->
```

This paragraph is not meant for production publications.

```
<!--/condition-->
```

## Multiple Conditions

Multiple Conditions can be used in a single conditional block [Operators](#). This example only show the block of text if `print_only` AND `production` are set to `Visible`.

```
<!--condition:print_only production-->
```

This paragraph is meant only for print and production.

```
<!--/condition-->
```

## Conditions Behavior

Conditions are rendered or removed from the document when publishing based on values set in the [Conditions Window](#). Conditions are considered unset, and therefore always render, if they haven't been scanned into ePublisher. Conditions are also considered unset if they still have the default value `Use document condition` in the Conditions Window.

## Using Conditions

First, create a condition by writing it in a source document. Below, a block of conditional text is created with the Condition `print_only`.

```
<!--condition:print_only-->
```

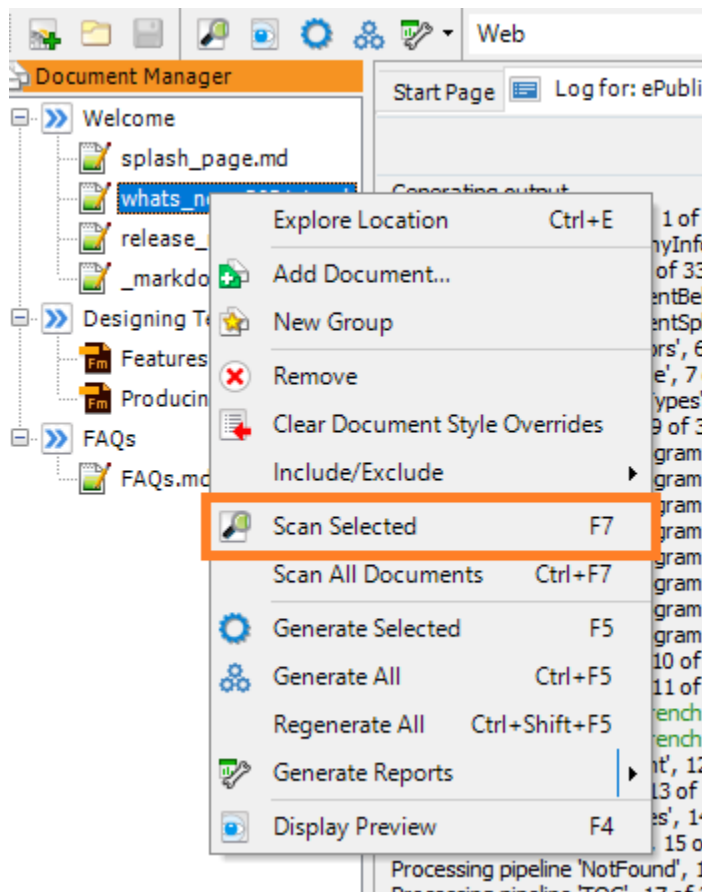
```
# The Print Section
```

This sections is meant only for print.

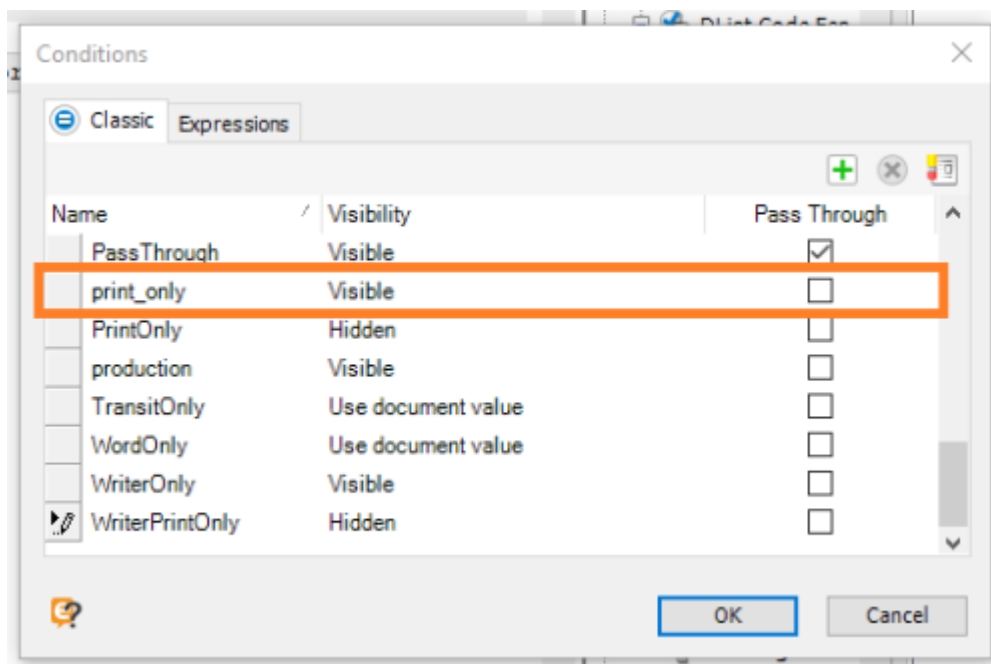
It will not be visible if `'print_only'` is set to `'Hidden'`.

```
<!--/condition-->
```

Next, scan the document in ePublisher. This will add the Condition `print_only` to the Conditions Window.



Once scanned, the `print_only` Condition's value can be changed to either `Visible` or `Hidden`. The Condition is considered unset, and will always render, if the value is left with the default, `Use document value`.



## Condition Operators

Using Operators, an author can create additional logic to determine whether conditional text should be rendered or hidden.

In a conditional statement, the block of text renders if the entire statement evaluates to `true`. The block is hidden if the statement evaluates to `false`.

In this context, `Visible` is considered `true`, and `Hidden` is considered `false`. `Use document value` disables the conditional statement and always renders the block.

Combine Condition names and Operators to create complex statements to determine if the content should be rendered or removed in the publication.

### The Space Operator - Logical AND

The space character in a conditional statement is a Logical AND. Meaning, if the statements on both sides of  evaluate to `true`, the statement passes.

The conditional text below is rendered if `print_only` AND `production` are set to `Visible`.

```
<!--condition:print_only production-->
```

This paragraph is meant only for print and production.

```
<!--/condition-->
```

## The `,` Operator - Logical OR

The `,` character in a conditional statement is a Logical OR. Meaning, if a statement on either side of `,` evaluates to `true`, the statement passes.

The conditional text below is rendered if one of `print_only` OR `production` are set to `Visible`.

```
<!--condition:print_only,production-->

This paragraph is meant for either print or production.

<!--/condition-->
```

## The `!` Operator - Logical NOT

The `!` character in a conditional statement is a Logical NOT. Adding `!` to the beginning of a Condition reverses it's truthiness. Meaning, a Condition with `!` on the front of it's name evaluates `Visible` to `false` and `Hidden` to `true`.

The conditional text below is removed if `production` is set to `Visible`.

```
<!--condition:!production-->

This paragraph is not meant for production.

<!--/condition-->
```

## Conditions and Includes

Conditions can be used as expected in [File Includes](#) since they are processed at the same time as Includes. Additionally, Includes can be used inside of conditions to import entire documents based on certain Conditions.

## File Includes

A File Include statement points to another Markdown++ file and imports the file's contents at the location of the statement. This enables multi-file structure in a single document.

### Syntax

An Include statement is created by writing a path to a Markdown++ document between `<!--include:` and `-->`. Relative paths and absolute paths are both valid

path values. Web paths are not supported. The include statement must be the only thing on a line.

## Basics

A basic Include statement. The Include must be written on it's own line to work properly.

```
<!--include:my_file.md-->
```

Relative paths and absolute paths are fine to use.

```
<!--include:my_file.md-->
```

```
<!--include:C:/Users/Me/Docs/my_file.md-->
```

Multiple includes can be used in the same document.

```
<!--include:my_file.md-->
```

```
<!--include:doc_2.md-->
```

## File Includes Behavior

When ePublisher detects a File Include statement, the file is read, and the Include tag is replaced with the content of the file. If the no file is found at the path given, the Include tag will be passed through to the output as an HTML Comment.

## Using a File Include

To use an Include statement, all that needs to be done is write the tag where the file's content is to be imported. Below, an include statement is written below a Title.

```
Learning ePublisher
```

```
=====
```

```
<!--include:epublisher_basics.md-->
```

This can even be done inside of documents used in an Include statement, as long as it is not a [Recursive Include][mdp-includes-recursion]. Use this feature to create Map Files for many Markdown++ documents, or create documents needed for content re-use.

## Recursive Includes

If an Include statement tries to insert a document that has already been inserted by a parent file, ePublisher's generation log will display a message like this one:

```
[Warning]

Skipping recursive include file:

'C:\Users\Me\Documents\include_doc.md'

in file: 'C:\Users\Me\main_doc.md'
```

This message displays because ePublisher cannot insert the document. Doing so would create a recursive loop and would break the generation. If this message is received, it's time to look at the layout of Includes in the source documents.

The message can be useful to track down the file in error. The first file path refers to the file in the attempted Include statement. The second file path refers to where the Include occurred.

## Variables

Variables represent a shorthand to store a value that can be re-used across a set of documents. They're useful to store content that only needs to be written once, but is used in the same way in many places. Store values like product names, copyright text, publication dates, and more inside of Variables.

### Syntax

Variables are the only syntax in Markdown++ that doesn't use the HTML Comment Tag.

To write one, start with `$`, write the variable name using alphanumeric characters, `-`, and `_`. End the Variable with `;`. Do not use spaces in the Variables's name.

### Basics

A simple example of writing a Variable, called `product_name`.

```
$product_name;
```

Variables can be intermixed with other text content.

```
Document last published on $publish_date;.
```

The full range of Markdown features is available with Variables as well, both around them and written in their values.

```
The documentation for our product, **$product_name**.
```

## Variable Behavior

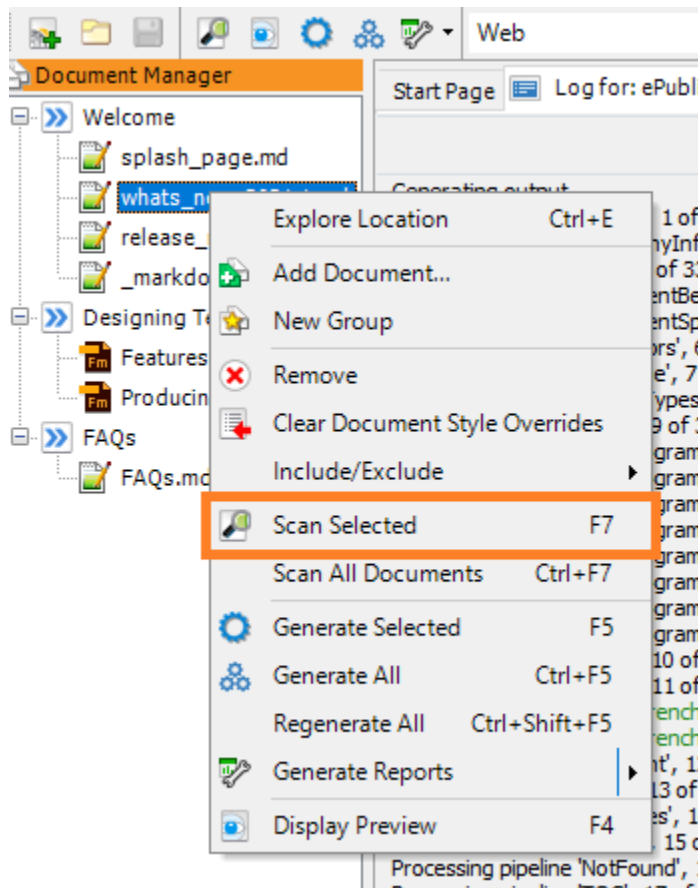
Variables, once scanned into ePublisher, can be given values that are saved to an ePublisher Project.

### Using Variables

First, a Variable must be created by writing it into a document. Here, we create a Variable called `publish_date`.

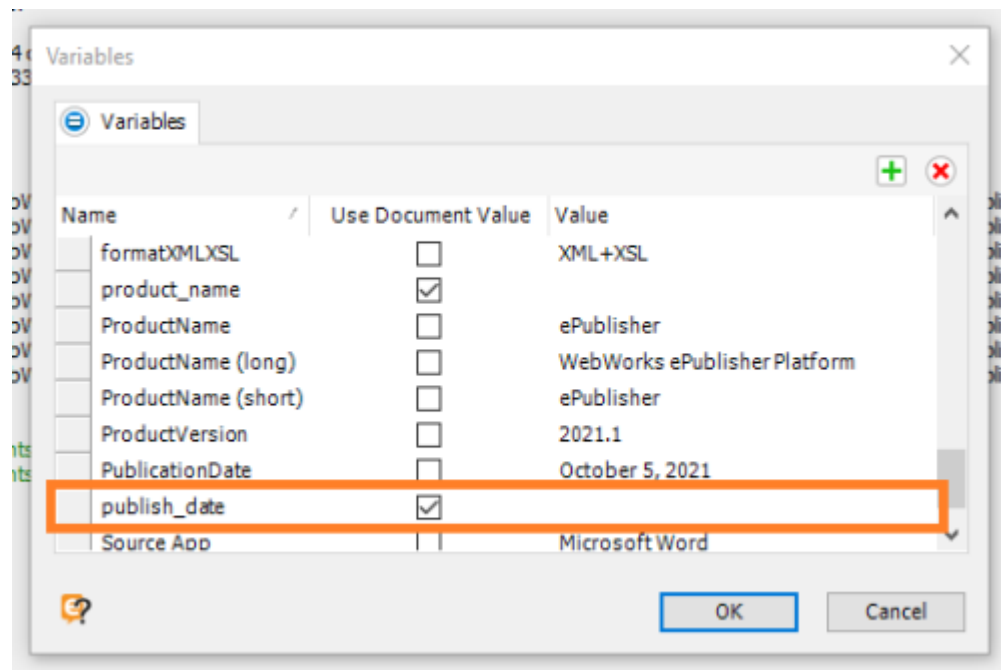
```
Document last published on $publish_date;.
```

Next, scan the document in ePublisher. This will add the Variable to the [Variables Window](#).





The Variable can now be given a value, typed in the input field next to the Variable's name in the Variable Window.



## Use Document Value

The **Use Document Value** checkbox inside the Variables Window does not apply to Markdown++ Variables, since their values are instead maintained in ePublisher. There's no change in behavior based on if the box is checked or not. This feature applies to legacy source document types, such as FrameMaker and Word.

See Online Help for Markdown++ cheatsheet.

# Adobe FrameMaker

- Adobe FrameMaker Formats and Standards
- Implementing Online Features in FrameMaker
- Working with Tables in FrameMaker
- Working with Images in FrameMaker
- Working with Videos in FrameMaker
- Creating Index Entries in FrameMaker
- Using Variables in FrameMaker
- Using Conditions in FrameMaker
- Specifying Output File Names in FrameMaker
- Creating Context-Sensitive Help in FrameMaker
- Creating Popup Windows in FrameMaker
- Creating Expand/Collapse Sections (Drop-Down Hotspots) in FrameMaker
- Creating Related Topics in FrameMaker
- Creating See Also Links in FrameMaker
- Creating Meta Tag Keywords in FrameMaker
- Assigning Custom Page Styles in FrameMaker
- Opening Topics in Custom Windows in FrameMaker
- Customizing TOC Entry in FrameMaker
- Customizing Table of Contents Icons in FrameMaker
- Specifying Context Plug-ins in FrameMaker
- Creating Accessible Online Content in FrameMaker
- Troubleshooting FrameMaker issues

If you want to implement online content features in your generated output, you need to prepare your Adobe FrameMaker source documents for output generation. This section explains how to prepare your Adobe FrameMaker source documents.

## Adobe FrameMaker Formats and Standards

Adobe FrameMaker provides a comprehensive publishing solution with XML-based structured authoring. You can develop the templates you need to deliver polished technical documentation for large product libraries. FrameMaker allows you to create both structured and unstructured content. You can also create DITA-compliant content.

This section describes the design considerations for a FrameMaker catalog and template files. By effectively designing your FrameMaker template, and by consistently applying formats throughout your source documents, you can streamline single-sourcing processes and reduce your production and maintenance costs. This section does not describe all FrameMaker processes, but it focuses on the design considerations related to ePublisher.

## Standards for Single-Sourcing

To define your FrameMaker standards, create a FrameMaker file with all the elements you need in it, including formats, markers, conditions, variables, tables,

and master pages. You can use this file to import and update your standards. For example, you can use this file to update variables and conditions across your FrameMaker source files. You can also use this file as a source document in your Stationery design project. To create a template file, start with one of the default templates provided with FrameMaker that most closely matches the format you want. Then, customize the default file to meet your specific needs. The following sections describe various template areas and considerations, and how to effectively design your FrameMaker template file to support single-sourcing with ePublisher.

## Planning for Importing Elements Across Files

You need to carefully plan your FrameMaker standards so you can import all elements, such as formats, page layouts, variables, and conditions across all source files. To avoid issues, do not reuse an element for two different purposes in two different files. For example, if the footer text differs between the front matter and the main chapters of a book, do not use the same variable to define the footer in both files. Otherwise, you cannot import that variable across files.

To avoid conflicts when importing master pages, use different page layout names for special pages in all files, such as Title, TOC right, TOC left, and Index first. In addition, delete unneeded formats, variables, and conditions to simplify your template use and maintenance.

## Paragraph Formats in FrameMaker

Create paragraph formats for items based on function, not based on formatting. This approach allows you to modify formatting over time and the format names continue to apply. It also prepares you for structured writing in the future. If you are using DITA, paragraph formats are already defined.

Name your paragraph formats starting with naming conventions that group formats by function. For example, group procedure-related formats together by starting the format names with Procedure, such as ProcedureIntro, ProcedureStep1, ProcedureStep, ProcedureSubStep1, and ProcedureSubstep. You do not need to restart numbering using a step1 format. If you have a format that always proceeds a numbered list, such as a ProcedureIntro, you can restart the numbering with that format, which allows you to not use a step1. Either method is fine, but one can require less maintenance when updating steps in a procedure topic.

**Note:** Format names should not include a period in their name. The period can cause display issues when ePublisher creates the cascading style sheet entry that defines the appearance of the format.

To simplify formatting and save time for future maintenance and customization, set the default paragraph font for all formats, then customize specific formats that need customization. You may need multiple paragraph formats to define functions that

support pagination settings, such as a BodyListIntro format that has **Keep with next paragraph** set.

In ePublisher, you can scan the source documents to list all the paragraph formats. Then, you can organize them in ePublisher to allow property inheritance and to streamline the customization process for your generated output.

To automate and simplify template use, define the paragraph format that follows each paragraph format. This process allows the writer to press Enter after writing a paragraph and the template creates the next paragraph with the format most commonly used next. For example, after a Heading format, the writer most often writes a body paragraph of content.

Common paragraph formats include:

- Anchors for images and tables. You may need multiple indents, such as Anchor, AnchorInList, and AnchorInList2.
- Body paragraphs. You may need multiple indents, such as Body, BodyInList, and BodyInList2.
- Headings, such as ChapterTitle, AppendixTitle, Heading1, Heading2, Heading3, and Heading4. You may also need specialized headings, such as Title, Subtitle, FrontMatterHeading1, FrontMatterHeading2, and FrontMatterHeading3.
- Bulleted lists. You may need multiple bullet levels, such as Bullet, Bullet2, Bullet3. You may also need a bullet item within a procedure, such as a ProcedureBullet and a bullet item within a table, such as a CellBullet. For more information, see “Bulleted and Numbered Lists in FrameMaker”.
- Numbered lists. You may need multiple levels, such as ProcedureStep that uses numbers and ProcedureSubstep that uses lowercase letters. You can use ProcedureStep1 and ProcedureSubstep1 to restart numbering, or you can use a common paragraph that precedes each list to restart numbering. You may also need numbered list items in tables, such as CellStep and CellStep1. Be sure to consider related supporting formats, such as ProcedureIntro. For more information, see “Bulleted and Numbered Lists in FrameMaker”.
- Examples, such as code or command syntax statements, usually in a fixed font. To keep the lines of a code example together, you can set the Example format to keep with next paragraph and use an ExampleLast format to identify the end of the example. You may also need multiple example levels, such as ExampleInList and ExampleInListLast.
- Paragraphs in tables, such as CellHeading, CellBody, CellBody2, CellStep, CellStep1, and CellBullet. Although you can reuse paragraph formats in tables and adjust the margins when those formats are in a table in FrameMaker, create unique paragraph formats for use in tables to give you complete control in ePublisher.

- Legal notice and copyright or trademark formats for inside the cover page.
- Table of contents and Index formats. However, these formats are defined on the reference pages rather than as paragraph formats.
- Definition lists, such as term and definition or description. You can use a two-column table for this purpose, but a definition list allows long terms, such as field labels in a user interface, to run across the page without wrapping. Then, the definition or description are indented below the term.
- Header and footer formats to control formatting.
- Notes, cautions, tips, and warnings. You can use the numbering property of a paragraph format to insert default text at the beginning of a paragraph, such as Note, Caution, Tip, or Warning. In ePublisher, you can use the **Bullet** properties for the paragraph style to add an image to the left of each note, caution, tip, or warning.
- Page breaks, which can be identified with a small-font paragraph format with **Keep with previous paragraph** set and a large space below the paragraph that pushes the next paragraph to the next page. This paragraph format is hidden in online content. This approach allows you to put page breaks in the content where needed to achieve the cleanest printed output without customizing the pagination settings for individual paragraphs. However, you need to review all these paragraphs each release and remove unneeded PageBreak paragraphs. This approach increases maintenance, but it prevents format customization for pagination.

ePublisher projects use custom marker types, paragraph formats, and character formats to define online features. You need to give the list of markers and formats to the writers so they know how to implement each online feature. The writers use the markers and formats you create to define online features.

The Stationery defines the custom markers and formats. To reduce complexity, you can use the format names defined in the documentation, or you can define the online feature to a different format. The following list identifies additional paragraph formats you may need to support ePublisher online content features:

- Paragraph or character formats to support multiple languages, such as bidirectional languages and text.
- Dropdown paragraph format that identifies the start of an expand/collapse section. You can end the section with a paragraph format defined to end the section, or with a DropDownEnd marker.
- Popup paragraph formats that define several aspects of popup window content:

- Popup paragraph format identifies the content to display in a popup window and in a standard help topic. This format is applied to the first paragraph of popup content.
- Popup Append paragraph format identifies the content to display in a popup window and in a standard help topic. This format is applied to additional popup paragraphs when you have more than one paragraph of content to include in a popup window.
- Popup Only paragraph format identifies the content to display only in a popup window. This format is applied to the first paragraph of popup content.
- Popup Only Append paragraph format identifies the content to display only in a popup window. This format is applied to additional popup paragraphs when you have more than one paragraph of content to include in a popup window.
- Related topics paragraph format that identifies a link to a related topic, such as a concept topic related to a task or a task related to a concept.
- See Also paragraph format that identifies the text you want to include in an inline See Also link.

For more information about enabling a specific online feature, see “Designing, Deploying, and Managing Stationery”.

## Character Formats in FrameMaker

Create character formats for items based on function, not based on formatting or appearance. This approach allows you to modify formatting over time and the format names continue to apply. It also prepares you for structured writing in the future. If you are using DITA, character formats are already defined.

For character formats, use As Is to start with as base, which allows you to apply multiple character formats to the same text. It also allows each character format to define only the aspects of the formatting required for that character format. Customize each format for your specific need by specifying only the properties required for that format.

Common character formats include:

- Book titles in cross references
- Emphasized text
- Command names

- File and folder names
- User interface items
- Optional steps or if clauses used to introduce optional steps
- Links
- New terms
- Step numbers, which allows you to apply formatting to the number for a step
- Text the user must type
- Variables

ePublisher projects use custom marker types, paragraph formats, and character formats to define online features. You need to give the list of marker types and formats to the writers so they know how to implement each online feature. The writers use the markers and formats you create to define online features.

The Stationery defines the custom marker types, paragraph formats, and character formats. To reduce complexity, you can use the format names defined in the documentation, or you can define the online feature to a different format. The following list identifies additional character formats you may need to support ePublisher online content features:

- Link character format, which identifies the text to include in the link. Include the marker and text in the Link character format.
- Multiple language support, such as bidirectional languages and text, can require a paragraph or character format with Bidi support enabled.
- Abbreviation character format identifies abbreviation alternate text for browsers to display for abbreviations, such as SS#, when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. This character format is used in combination with the AbbreviationTitle marker type.
- Acronym character format identifies acronym alternate text for browsers to display for acronyms, such as HTML, when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. This character format is used in combination with the AcronymTitle marker type.
- Citation character format identifies the source of a quote using a fully-qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. This character format is used in combination with the Citation marker type.

- See Also character format identifies the text you want to include in a See Also button. This format controls the appearance of the text on the button.

For more information about enabling a specific online feature, see “Designing, Deploying, and Managing Stationery”.

## Bulleted and Numbered Lists in FrameMaker

ePublisher uses a table-like structure with two columns to display any paragraph format with a hanging indent, such as bulleted and numbered list items, in generated output. ePublisher uses the numbers, characters, formats, and fonts from the source documents for the bullets or numbers. Since some fonts are not available on all computers, you should use character formats in ePublisher to override the formatting of the bullets or numbers. You can also use an image in ePublisher for bullets.

For bulleted lists, you may need multiple bullet levels, such as Bullet, Bullet2, and Bullet3. You may also need a format for a bullet within a procedure, such as a ProcedureBullet, and a bullet within a table, such as a CellBullet. Make sure you consider all supporting formats you may need, such as a ListIntro format for the paragraph that introduces the bulleted list, which should be set to stay with the list (Keep with Next).

For numbered lists, you may need multiple levels, such as a ProcedureStep that uses numbers and a ProcedureSubstep that uses lowercase letters. To restart numbering, you can use a ProcedureStep1 and a ProcedureSubstep1 format. If you have a common paragraph format that precedes each list, you can use that paragraph format to restart numbering, which would eliminate the need for a ProcedureStep1 format. You may also need a numbered list item in tables, such as CellStep and CellStep1. Make sure you consider all supporting formats you may need, such as a ProcedureIntro format

**Note:** Be aware of paragraphs that have a hanging indent. The hanging indent can cause incorrect alignment of text on the first line of your generated output. For more information see “Defining the Appearance of Numbered Lists”.

## Image Formats and Considerations in FrameMaker

If ePublisher cannot use an original image in the output, or if ePublisher determines it needs to modify the image based on how it is included in the source document, ePublisher rasterizes the image using the options you define for your graphic styles in Style Designer. For example, you can define the dots per inch (DPI) and format for the final images. Rasterization of an image can cause the image to be less clear in the output.



To avoid reduced image quality in your output, and to avoid an extended transformation time during the Image stage and pipeline, review the following considerations:

- When ePublisher encounters an anchored frame in your FrameMaker source documents, ePublisher checks for the following conditions:
  - \_ Is the frame a different size than the original image?
  - \_ Is there white space in the frame?
  - \_ Is the image copied into the document, rather than imported by reference?
  - \_ Is the original image a file format other than `.jpg`, `.gif`, `.png`, or `.svg`?
  - \_ Are there additional elements in the frame, such as text boxes, multiple images, or callouts?

If ePublisher determines that any of these conditions apply, ePublisher rasterizes the entire frame and applies the options you defined in Style Designer.

- To display images at full size in online output and avoid resizing, which can cause the image to be rasterized, set the **By reference graphics use document dimensions** option for your graphic styles to **Disabled**.
- If you want ePublisher to rasterize all images according to your Style Designer options, set the **By reference graphics** option to **Disabled** for all graphic styles.
- When ePublisher finds an image included by-reference that is the original size, is shrink-wrapped, and contains no callouts, ePublisher copies the image directly into the output folder in most cases, bypassing the graphic style options.
- To improve the image quality in your output, resize your images as needed using an image editing application before importing them, rather than adjusting the DPI or scale in FrameMaker. Otherwise, an image included by reference retains its original file size, and it is either scaled by the browser or rasterized according to the size of the anchored frame, which can result in a distorted image.
- For the best compatibility with most computer monitors, save and import your images at 96 DPI using a format that ePublisher does not rasterize.
- Image callouts are useful in many publications. However, text boxes and line drawings cause images to be rasterized, which can make images less clear in

your output. Add and edit callouts in your image editing application and then import the single, final image to avoid the rasterization process.

- If you use `.svg` image files, you need to configure the `.svg` options to specify whether to rasterize these images. Some output formats and some browsers do not support `.svg` image files.
- You can add text boxes with GraphicStyle markers to your images without causing the image to be rasterized, since markers do not affect the appearance or format of an anchored frame.
- ePublisher does not include images from FrameMaker Master or Reference pages, and it does not include content outside the main text flow.
- Store image files and source documents on the local computer when generating output.

## To achieve the best results when inserting images in FrameMaker

1. Create a unique paragraph format for images. Use the paragraph alignment properties to control the position of your images. Make sure the **Fixed** check box in the **Line Spacing** section is *not* selected for the paragraph format.
2. Insert an anchored frame in an empty paragraph of the format created in step 1.
3. Import your image file by reference into the anchored frame rather than copying it into the document. ePublisher supports only `.jpg`, `.gif`, `.png`, and `.svg` files. ePublisher rasterizes all other formats.
4. Import the image at the native resolution of the image.
5. Shrink-wrap the frame (type Esc+m+p with the frame selected) and change its **Anchoring Position** to **At Insertion Point** or **Below Current Line**.

## Table Formats in FrameMaker

Table formats allow you to define standard tables and quickly apply those standards to tables in your source documents. When you define your table formats, be sure to consider the various types of tables you may need, such as with lines, without lines, checklists, and action/result tables. You can use a table without lines to layout content within an area on a page, such as a definition list with short terms. You can also create a table format for each indent position needed. For example, you can create a table format to use for tables within a bulleted list that is indented to align with the text of each bulleted list item.

ePublisher allows you to define how the header, footer, and main rows of a table appear in your generated output. To support these formatting properties, your tables must have each of these parts defined in your source documents. If a table does not have a header defined, ePublisher cannot apply the formatting defined for the header row.

ePublisher applies the paragraph and character formats you define for content within each cell. You can also configure ePublisher to ignore character formats in a table. You may need additional paragraph formats to use in tables, such as CellBody and CellBullet, so you can define the proper margins and appearance for your generated output. You cannot adjust paragraph formats to change their appearance when used within tables.

## Cross Reference Formats in FrameMaker

Cross reference formats allow you to quickly use consistent cross references throughout your source documents. However, you probably want to change the appearance of your cross references in your generated output. For example, you

may not want to include page numbers in your online content. ePublisher allows you to define how each cross reference format appears in your generated output.

Define the cross reference formats you need in your source documents, such as references to headings, steps, figures, tables, and chapters. Then, you can define each of these formats separately in ePublisher.

## Markers in FrameMaker

FrameMaker uses markers to implement standard features, such as index entries and hypertext links. ePublisher recognizes these standard markers and uses them to implement these standard features in your generated output.

ePublisher projects also use custom marker types, paragraph formats, and character formats to define online features. You need to give the list of marker types and formats to the writers so they know how to implement each online feature. The writers use the markers and formats you create to define online features.

The Stationery defines the custom marker types, paragraph formats, and character formats. Markers with reserved names have their functions defined by default. You can use these default names, or you can create your own markers. To reduce complexity, use the default marker names, which are also used throughout the documentation. You can also use the format names defined in the documentation to reduce complexity. The following table lists the default custom marker types used to implement online features.

<b>Marker Type</b>	<b>Description</b>
AbbreviationTitle	Specifies abbreviation alternate text for browsers to display for abbreviations such as SS# when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. Used in combination with the Abbreviation character format.
AcronymTitle	Specifies acronym alternate text for browsers to display for acronyms such as HTML when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. Used in combination with the Acronym character format.
Citation	Specifies the source of a quote using a fully qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. Used in combination with the Citation character format.
Context Plugin	Specifies context plug-ins for Eclipse help systems. Other Eclipse plug-ins can use the context plug-in IDs to call the Eclipse help system. For more information, see "Using Markers to Specify Context Plug-ins in Eclipse Help".
DropDownEnd	Marks the end of an expand/collapse section. Used in conjunction with an Expand/Collapse paragraph format.
Filename	Specifies the name of an output file for a page or an image.
GraphicScale	Specifies a percentage to use to resize an image, such as 50 or 75 percent, in generated output.
GraphicStyle	Specifies the name of a graphic style defined in a project to apply to an image. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality.

Marker Type	Description
Hypertext	Specifies a link using the <code>newlink</code> and <code>gotolink</code> commands in Adobe FrameMaker. This marker type is a default Adobe FrameMaker marker type ePublisher automatically maps.
ImageAltText	Specifies alternate text for an image. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content.
ImageAreaAltText	Specifies alternate text for clickable regions in an image map. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content.
ImageLongDescByRef	Specifies the path to the file that contains the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content.
ImageLongDescNotReq	Specifies that a long description is not required for an image, which bypasses this accessibility check for the image when you create accessible content.
ImageLongDescText	Specifies the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content.
Keywords	Specifies the keywords to include in the <code>meta</code> tag for the topic. The <code>meta</code> tag improves searchability on the Web.
PageStyle	Specifies the name of a page style defined in the project to apply to a topic. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality.

Marker Type	Description
PassThrough	Specifies that ePublisher place the contents of the marker directly into the generated output without processing the content in any way. For example, you could use a PassThrough marker if you wanted to embed HTML code within your generated output.
Popup	Specifies the start of the content to include in a popup window. The content is displayed in a popup window when you hover over the link. When you click the link in some output formats, the topic where the popup text is stored, such as the glossary, is displayed.
PopupEnd	Marks the end of the content to include in a popup window.
PopupOnly	Specifies the start of the content to include in only a popup window. Browsers display the content in a popup window when you hover over or click the link.
RubiComposite	No longer supported.
SeeAlsoKeyword	Specifies an internal identifier for a topic. SeeAlsoLink markers in other topics can list this identifier to create a link to this topic. Used in conjunction with a See Also paragraph format or character format.
SeeAlsoLink	Identifies an internal identifier from another topic to include in the list of See Also links in this topic. Used in conjunction with a See Also paragraph format or character format.
SeeAlsoLinkDisplayType	Specifies whether to display the target topics on a popup menu or in a window. By default, the links are displayed in the Topics Found window. To display a popup menu, set the value to <code>menu</code> . This marker type is supported only in HTML Help.
SeeAlsoLinkWindowType	Specifies the name of the window defined in the <code>.hhp</code> file, such as TriPane or Main, that the topic opens

Marker Type	Description
	in when the user clicks the link. This marker type is supported only in HTML Help.
TableSummary	Specifies an alternate text summary for a table, which is used when you create accessible content. This text is added to the <code>summary</code> attribute of the <code>table</code> tag in the output. Screen readers read this description when you create accessible content.
TableSummaryNotReq	Specifies that a summary is not required for a table, which bypasses this accessibility check for that table.
TOCIconHTMLHelp	Identifies the image to use as the table of contents icon for a topic in the HTML Help output format.
TOCIconJavaHelp	Identifies the image to use as the table of contents icon for a topic in the Sun JavaHelp output format.
TOCIconOracleHelp	Identifies the image to use as the table of contents icon for a topic in the Oracle Help output format.
TOCIconWWHelp	Identifies the image to use as the table of contents icon for a topic in the WebWorks Help output format.
TopicAlias	Specifies an internal identifier for a topic that can be used to create a context-sensitive link to that topic.
TopicDescription	Specifies a topic description for a context-sensitive help topic in Eclipse help systems. For more information, see "Using Markers to Specify Topic Descriptions for Context-Sensitive Help Topics in Eclipse Help".
WhatIsThisID	Identifies a What Is This help internal identifier for creating context-sensitive What Is This field-level help for Microsoft HTML Help.
WindowType	Specifies the name of the window defined in the help project that the topic should be displayed in. In Microsoft HTML Help, the window names are defined



Marker Type	Description
	in the <code>.hhp</code> file. This marker type is supported in Microsoft HTML Help and Oracle Help.

## Variables and Conditions in FrameMaker

**Variables** allow you to define a phrase once and consistently use that phrase throughout your source documents. Then, if you ever need to change that phrase, you can change it in one location and apply that change throughout your source documents. For example, you can use variables for product names, book titles, and company names.

**Conditions** allow you to single-source information and include or exclude specific sets of information. You can apply a condition to a character, word, sentence, paragraph, or entire sections of content. Then, you can specify whether to show or hide the content with that condition applied to it. This capability allows you to create multiple version of content based on your specific needs. You can also use conditions to include and exclude notes to the writer or reviewer during the content development process. When you combine variables and conditions, you can customize information for multiple versions of a product while reducing your maintenance costs by reducing duplicate information.

When working with conditions, you can customize the appearance of content with a condition applied by using color, underline, overline, and strikethrough formatting for the condition. This formatting helps you maintain and work with the content. However, ePublisher does not display this formatting in the generated output. If you want to apply formatting in the generated output, use paragraph and character formats to define the appearance for the content.

To simplify consistently setting variables and conditions across your source documents, create a standard file with all the variables and conditions defined in it. The file can display the value of each variable and the show/hide state of each condition. Writers can set the variables and conditions as needed in this one file. Then, the writers can import the variables and conditions from this file into all the source documents.

**Note:** Use each variable and condition for the same purpose and value in all source documents. For example, if you want the footer in the preface file to be different from the footer in the chapter files, use a different variable to define the footer in each file. Otherwise, you cannot import variables across all the source documents.

## Page Layouts in FrameMaker

A **master page** defines the layout of one or more pages and includes all design elements, such as headers, footers, background text, and graphics, for every page that uses that master page. A master page allows you to define the layout of multiple pages in one place, and apply that layout to multiple pages. If you want to adjust the page layout, you need change it only in one place. Each template has at least one master page.

You can create multiple master pages, such as one for odd pages, one for even pages, and one for the first page of each chapter. To simplify page management and being able to import master pages across files, do not redefine a master page to have a different layout in different files. For example, if you want odd pages to have a different footer in the front matter than in the chapters, create a master page for each case, such as ChapterOdd and PrefaceOdd.

You may need to create many master pages for special purposes, such as Title, LegalNotice, PrefaceOdd, PrefaceEven, ChapterFirst, ChapterOdd, ChapterEven, AppendixFirst, AppendixOdd, AppendixEven, IndexFirst, IndexOdd, and IndexEven.

## Reference Pages, Table of Contents, and Indexes in FrameMaker

**Reference pages** define default images, lines, heading levels, and formatting for generated table of contents and index files. You can use reference pages to simplify content creation in your source documents. For more information about reference pages and formatting generated table of contents and indexes in your printed content, see the Adobe FrameMaker documentation and help.

Since graphics and lines defined on the reference pages are not in the main flow, ePublisher cannot include these items in the generated output. For images and lines, use anchored frames in your content to include the images by reference.

Since the appearance of online table of contents and indexes often differ from printed versions, you need to be able to deliver customized table of contents and indexes in your online content. Therefore, ePublisher does not need the table of contents and index formatting defined on the reference pages. ePublisher allows you to define the table of contents levels and appearance, as well as the appearance of the index in your generated output. ePublisher uses the index markers throughout your source files to build the online index. This powerful support allows you to deliver the online content you require.

## Implementing Online Features in FrameMaker

Implement online features in your output by preparing your Adobe FrameMaker source documents with custom marker types, paragraph formats, and character

formats defined by the Stationery designer for your Stationery. These markers and styles define the presentation and behavior of your online content. For example, markers can define the name of the file generated for a topic. Formats can define how content displays online.

## Custom Marker Types in FrameMaker

ePublisher projects use the custom marker types to implement online features when generating output. Before you begin using custom marker types, talk to the Stationery designer and verify which online features your Stationery supports. Your Stationery only recognizes the custom marker types defined by the Stationery designer in your Stationery. If you try to implement online features using custom marker types not supported in your Stationery, ePublisher does not recognize these items when generating output. ePublisher correctly converts all standard Adobe FrameMaker marker types. In addition, ePublisher also supports several custom marker types you can use to implement online features in your generated output.

When the Stationery designer creates the Stationery, the Stationery designer can use the default name for a custom marker type or the Stationery designer can use a different name for the custom marker type. The following table lists the default names of custom marker types used to implement online features. Always verify with the Stationery designer the names of the custom marker types you should use when implementing online features before you use these items in your source documents.

Marker Type	Description
AbbreviationTitle marker type	Specifies abbreviation alternate text for browsers to display for abbreviations such as SS# when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. Used in combination with the Abbreviation character format. For more information, see "Assigning Alternate Text to Abbreviations in FrameMaker"
AcronymTitle marker type	Specifies acronym alternate text for browsers to display for acronyms such as HTML when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. Used in combination with the Acronym character format. For more information, see "Assigning Alternate Text to Acronyms in FrameMaker".
Citation marker type	Specifies the source of a quote using a fully qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. Used in combination with the Citation character format. For more information, see "Providing Citations for Quotes in FrameMaker".
Context Plugin marker type	Specifies context plug-ins for Eclipse help systems. Other Eclipse plug-ins can use the context plug-in IDs to call the Eclipse help system. For more information, see "Specifying Context Plug-ins in FrameMaker".
DropDownEnd marker type	Marks the end of an expand/collapse section. Used in conjunction with an Expand/Collapse paragraph format. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in FrameMaker".
Filename marker type	Specifies the name of an output file for a page or an image. For more information, see "Specifying Output File Names in FrameMaker".
GraphicScale marker type	Specifies a percentage to use to resize an image, such as 50 or 75 percent, in generated output. For

Marker Type	Description
	more information, see "Assigning Image Scales in FrameMaker".
GraphicStyle marker type	Specifies the name of a image style defined in a project to apply to an image. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality. For more information, see "Assigning Image Styles in FrameMaker".
Hypertext marker type	Specifies a link using the <code>newlink</code> and <code>gotolink</code> commands in Adobe FrameMaker. This marker type is a default Adobe FrameMaker marker type ePublisher automatically maps.
ImageAltText marker type	Specifies alternate text for an image. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content. For more information, see "Assigning Alternate Text to Images in FrameMaker".
ImageAreaAltText marker type	Specifies alternate text for clickable regions in an image map. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content. For more information, see "Assigning Alternate Text to Image Maps in FrameMaker".
ImageLongDescByRef marker type	Specifies the path to the file that contains the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content. For more information, see "Using Text in External Files to Assign Long Descriptions to Images in FrameMaker".
ImageLongDescNotReq marker type	Specifies that a long description is not required for an image, which bypasses this accessibility check for the image when you create accessible content. For more information, see "Excluding Images from Accessibility Report Checks in FrameMaker".

Marker Type	Description
ImageLongDescText marker type	Specifies the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content. For more information, see "Assigning Long Descriptions to Images in FrameMaker".
Keywords marker type	Specifies the keywords to include in the <code>meta</code> tag for the topic. The <code>meta</code> tag improves searchability on the Web. For more information, see "Creating Meta Tag Keywords in FrameMaker".
PageStyle marker type	Specifies the name of a page style defined in the project to apply to a topic. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality. For more information, see "Assigning Custom Page Styles in FrameMaker".
<u>PassThrough</u>	Specifies that ePublisher place the contents of the marker directly into the generated output without processing the content in any way. For example, you could use a PassThrough marker if you wanted to embed HTML code within your generated output.
Popup marker type	Specifies the start of the content to include in a popup window. The content is displayed in a popup window when you hover over the link. When you click the link in some output formats, the topic where the popup text is stored, such as the glossary, is displayed. For more information, see "Using Markers to Create Popup Windows in FrameMaker".
PopupEnd marker type	Marks the end of the content to include in a popup window. For more information, see "Using Markers to Create Popup Windows in FrameMaker".
PopupOnly marker type	Specifies the start of the content to include in only a popup window. Browsers display the content in a popup window when you hover over or click the link.

Marker Type	Description
	For more information, see "Using Markers to Create Popup Windows in FrameMaker".
RubiComposite marker type	No longer supported.
SeeAlsoKeyword marker type	Specifies an internal identifier for a topic. SeeAlsoLink markers in other topics can list this identifier to create a link to this topic. Used in conjunction with a See Also paragraph format or character format. For more information, see "Creating See Also Links in FrameMaker".
SeeAlsoLink marker type	Identifies an internal identifier from another topic to include in the list of See Also links in this topic. Used in conjunction with a See Also paragraph format or character format. For more information, see "Creating See Also Links in FrameMaker".
SeeAlsoLinkDisplayType marker type	Specifies whether to display the target topics on a popup menu or in a window. By default, the links are displayed in the Topics Found window. To display a popup menu, set the value to <code>menu</code> . This marker type is supported only in HTML Help. For more information, see "Creating See Also Links in FrameMaker".
SeeAlsoLinkWindowType marker type	Specifies the name of the window defined in the <code>.hhp</code> file, such as TriPane or Main, that the topic opens in when the user clicks the link. This marker type is supported only in HTML Help. For more information, see "Creating See Also Links in FrameMaker".
TableStyle marker type	Specifies the name of a table style defined in the project to apply to a table in versions of Microsoft Word that did not support table styles. This marker type is an internal marker type that is not displayed in Stationery Designer. This marker type is supported only for Microsoft Word documents. You cannot create a marker type with a different name and assign it this functionality. For more information, see "Applying Table Styles in Word"

Marker Type	Description
TableSummary marker type	Specifies an alternate text summary for a table, which is used when you create accessible content. This text is added to the <code>summary</code> attribute of the <code>table</code> tag in the output. Screen readers read this description when you create accessible content. For more information, see "Assigning Alternate Text (Summaries) to Tables in FrameMaker".
TableSummaryNotReq marker type	Specifies that a summary is not required for a table, which bypasses this accessibility check for that table. For more information, see "Excluding Tables from Accessibility Report Checks in FrameMaker".
TOCIconHTMLHelp marker type	Identifies the image to use as the table of contents icon for a topic in the HTML Help output format. For more information, see "Customizing Table of Contents Icons in FrameMaker".
TOCIconJavaHelp marker type	Identifies the image to use as the table of contents icon for a topic in the Sun JavaHelp output format. For more information, see "Customizing Table of Contents Icons in FrameMaker".
TOCIconOracleHelp marker type	Identifies the image to use as the table of contents icon for a topic in the Oracle Help output format. For more information, see "Customizing Table of Contents Icons in FrameMaker".
TOCIconWWHelp marker type	Identifies the image to use as the table of contents icon for a topic in the WebWorks Help output format. For more information, see "Customizing Table of Contents Icons in FrameMaker".
TopicAlias marker type	Specifies an internal identifier for a topic that can be used to create a context-sensitive link to that topic. For more information, see "Creating Context-Sensitive Help in FrameMaker".
TopicDescription marker type	Specifies a topic description for a context-sensitive help topic in Eclipse help systems. For more information,



Marker Type	Description
	see "Specifying Context-Sensitive Help Links in FrameMaker".
WhatIsThisID marker type	Identifies a What's This help internal identifier for creating context-sensitive What's This field-level help for Microsoft HTML Help. For more information, see "Opening Topics in Custom Windows in FrameMaker".
WindowType marker type	Specifies the name of the window defined in the Help project that the topic should be displayed in. In Microsoft HTML Help, the window names are defined in the <code>.hhp</code> file. This marker type is supported in Microsoft HTML Help and Oracle Help. For more information, see "Opening Topics in Custom Windows in FrameMaker".

## Paragraph and Character Formats in FrameMaker

ePublisher projects use the paragraph formats and character formats defined by the Stationery designer to implement online features when generating output. Before you begin using paragraph formats and character formats to implement online features, talk to the Stationery designer and verify which online features your Stationery supports. Your Stationery only recognizes the paragraph formats and character formats defined by the Stationery designer in your Stationery. If you try to implement online features using paragraph formats and character formats not supported in your Stationery, ePublisher does not recognize these items when generating output.

When the Stationery designer creates the Stationery, the Stationery designer specifies the names of paragraph format and character formats used to implement an online feature. Consult with the Stationery designer to obtain the names of the paragraph formats and character formats defined by the Stationery designer to support each online feature you want to implement.

The following table lists the default names of paragraph formats and character formats used to implement online features. Always verify with the Stationery designer the names of the paragraph formats and character formats you should use when implementing online features before you use these items in your source documents.

<b>Format</b>	<b>Description</b>
AbbreviationTitle character format	Specifies abbreviation alternate text for browsers to display for abbreviations such as SS# when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. Used in combination with the AbbreviationTitle marker type. For more information, see "Assigning Alternate Text to Abbreviations in FrameMaker".
AcronymTitle character format	Specifies acronym alternate text for browsers to display for acronyms such as HTML when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. Used in combination with the AcronymTitle marker type. For more information, see "Assigning Alternate Text to Acronyms in FrameMaker".
Citation character format	Specifies the source of a quote using a fully qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. Used in combination with the Citation marker type. For more information, see "Providing Citations for Quotes in FrameMaker".
Expand/Collapse paragraph format	Specifies the content you want to include in an expand/collapse section. Used in conjunction with a DropDownEnd marker type. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in FrameMaker".
Popup paragraph format	Specifies the popup content to display in both a popup window and in a standard help topic. Applied to the first paragraph of popup content. For more information, see "Using Paragraph Formats to Create Popup Windows in FrameMaker".
Popup Append paragraph format	Specifies the popup content to display in a popup window and in a standard help topic. Applied to additional popup paragraphs when you have more than one paragraph of popup content. For more

<b>Format</b>	<b>Description</b>
	information, see "Using Paragraph Formats to Create Popup Windows in FrameMaker".
Popup Only paragraph format	Specifies the popup content to display in only a popup window. Applied to the first paragraph of popup content. For more information, see "Using Paragraph Formats to Create Popup Windows in FrameMaker".
Popup Only Append paragraph format	Specifies the popup content to display in only a popup window. Applied to additional popup paragraphs when you have more than one paragraph of popup content. For more information, see "Using Paragraph Formats to Create Popup Windows in FrameMaker".
Related Topic paragraph format	Specifies related topics links. For more information, see "Creating Related Topics in FrameMaker".
See Also character format	Specifies the text you want to include in a See Also button. For more information, see "Creating See Also Links in FrameMaker".
See Also paragraph format	Specifies the text you want to include in a See Also inline text link. For more information, see "Creating See Also Links in FrameMaker".

## Obtaining and Applying the Latest Adobe FrameMaker Template

An efficient, effective, and consistent ePublisher online content generation process relies upon the use of templates. Templates define marker types and paragraph, character, and table formats. Templates may also contain standard conditions, variables, and cross-reference definitions that you can use when creating and working with source documents used to generate online content. Templates help control the look and feel of source documents and generated output across multiple writers, multiple projects, and multiple types of generated output.

The ePublisher content generation process assumes that you use marker types and paragraph, character and table formats defined in an Adobe FrameMaker template prepared by a Stationery designer as you create content and format your source documents. Using Adobe FrameMaker templates and the marker types and paragraph, character, and table formats and other layout formats and

characteristics defined in templates ensures that you format content in your source documents consistently and also ensures ePublisher can use your source documents effectively to generate output.

If your source documents do not use templates or do not use the same marker types, formats, and standards defined in your Stationery by the Stationery designer, your generated output may not conform to the styles and standards defined by the Stationery designer for output. You may also not be able to implement some online features if you do not use the correct templates or the correct marker types and formats defined in the templates.

As a part of preparing your Adobe FrameMaker source documents for output generation, ensure your source documents use the correct Adobe FrameMaker templates from the Stationery designer and you have applied all paragraph, character, and table formats specified in the template correctly.

## **Importing Custom Marker Types in FrameMaker**

Typically the Stationery designer defines custom marker types supported in your ePublisher Stationery in an Adobe FrameMaker template. You then import the custom marker types defined in an Adobe FrameMaker template into your Adobe FrameMaker source documents by importing document properties from the Adobe FrameMaker template. This procedure explains how to import custom marker types from an Adobe FrameMaker template. For more information about creating custom marker types, see “Creating Custom Marker Types in FrameMaker”.

The following procedure provides an example of how to import custom marker types into Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for importing custom marker types in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To import custom marker types from an Adobe FrameMaker template into your source documents

1. In your Adobe FrameMaker source document, on the **File** menu, click **Import > Formats**.
2. In the **Import from Document** field, select the Adobe FrameMaker template that contains the custom marker types you want to import from the list.
3. In the **Import and Update** field, select only the **Document Properties** check box.
4. Click **Import**.
5. Click **OK** to confirm the operation.

## Creating Custom Marker Types in FrameMaker

Typically you should not need to create a custom marker type in an Adobe FrameMaker source document. If you want to use a custom marker type to implement an online feature, use the custom marker type provided in the Adobe FrameMaker template you use for your source documents. If you do not see a custom marker type you want to use to implement an online feature in the Adobe FrameMaker template, verify with the Stationery designer that your Stationery supports the custom marker type before you insert and use the custom marker in a source document.

Occasionally your Stationery may support a custom marker type that is not defined in the Adobe FrameMaker template you use with your source documents. In this situation, first confirm with the Stationery designer that your Stationery supports the custom marker type. After confirming your project supports the custom marker type, you can create the custom marker type in your Adobe FrameMaker source document.

The following procedure provides an example of how to create custom marker types in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating custom marker types in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create a custom marker type in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, on the **Special** menu, click **Marker**.
2. In the **Marker Type** field, select **Edit** from the drop-down list.
3. Type *Custom*`MarkerTypeName` to create a custom marker type, where *Custom*`MarkerTypeName` is the name of the custom marker type you want to create.

**Note:** The custom marker type name you type must match the name of the custom marker type supported in your ePubublisher Stationery. If you specify a name for the custom marker type that is different than the name of the custom marker type supported in your ePubublisher Stationery, ePubublisher will not be able to recognize and use the custom marker type when generating output.

4. Click **Add**.
5. Click **OK** to confirm the operation.
6. Click **Done**. Adobe FrameMaker displays the custom marker type you created in the Marker window in the **Marker Type** field.

## Creating a Passthrough Marker in FrameMaker

A passthrough marker is a marker that allows you to insert content that you do not want ePubublisher to process when you generate output. For example, if you have embedded multimedia files in your source documents, such as Audio Video Interleave files (`.avi`) or Adobe Software Flash files (`.swf`), you can insert a passthrough marker with a value that is set to the HTML code that you do not want ePubublisher to process.

The following example shows `.avi` code to which you could insert using a passthrough marker.

```
<embed src="sample.avi" width="400"
height="300" pluginspage="">
</embed>
```

## To create a passthrough marker in an Adobe FrameMaker source document

1. In Adobe FrameMaker, on the **Special** menu, click **Marker**.
2. In the **Marker Type** field, select **Passthrough** from the drop-down list.
3. **If the Passthrough marker type is not on the list**, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
4. In the **Marker Text** field, type the html code that you would like to not be processed by ePublisher such as the Flash embed code indicated in the previous topic.
5. Click **New Marker**.
6. Save your source document.
7. Generate output for your project. For more information, see “Generating Output”.
8. In Output Explorer, verify ePublisher created the appropriate result for your embedded html code. For more information, see “Viewing Output in Output Explorer”.

## Creating Cross-References and Links in FrameMaker

When you generate output, ePublisher automatically converts all cross-references in your Adobe FrameMaker source documents to links. Typically Stationery designers specify in Stationery how cross-references should display in generated output. For example, Stationery designers typically specify that cross-references that contain page numbers in source documents display without page numbers in generated output, as page numbers are out of context in online output. If you have target setting permissions, you can also customize the cross-reference formats you want to use when you generate output. For more information about customizing cross-reference settings, see “Setting Cross-References in Projects”.

Including cross-references in Adobe FrameMaker source documents is typically the easiest way to produce links in online content. However, in some cases you may not be able to achieve the effect you want by creating links using cross-references. In these cases, you can insert native Adobe FrameMaker Hypertext markers that use the `gotolink` and `message URL` hypertext commands in your Adobe FrameMaker source documents and use the Hypertext markers to create the links you want.

## To create a link using a cross-reference or Hypertext markers in an Adobe FrameMaker source document

1. ***If you want to create a link using a cross-reference***, complete the following steps:
  - a. In your Adobe FrameMaker source document, select the text for which you want to create a link.
  - b. On the **Special** menu, click **Cross-Reference**.
  - c. In the **Document** field, select the document that contains the content to which you want to link.
  - d. In the **Paragraph Tags** field, select the paragraph tag used for the content to which you want to link.
  - e. In the **Paragraphs** field, select the paragraph to which you want to link.
  - f. In the **Format** field, select the appropriate format for the link. For example, if you are creating a link to a glossary term, select a glossary term cross-reference format.
  - g. Click **Replace**.
2. ***If you want to create a link using hypertext markers***, complete the following steps:
  - a. In your Adobe FrameMaker source document, insert your cursor in front of the link target text.
  - b. On the **Special** menu, click **Marker**.
  - c. In the **Marker Type** field, select **Hypertext** from the list.
  - d. In the **Marker Text** field, type `newlink linkname` or `newlink filename:linkname`, where `linkname` is the name of the named destination for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document. To make maintenance easy, create short link names that use alphanumeric, lowercase characters.
  - e. Click **New Marker**.
  - f. Insert your cursor in front of the word or phrase for which you want to create a link.
  - a. On the **Special** menu, click **Marker**.



- b. In the **Marker Type** field, select **Hypertext** from the list.
  - c. In the **Marker Text** field, type `gotolink linkname` or `gotolink filename:linkname`, where `linkname` is the name of the named destination you created for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.
  - d. Click **New Marker**.
  - e. Select the word or phrase for which you want to create a link. The selected area must contain the both text and the hypertext marker you created.
  - f. Apply a link character format to the word or phrase. Applying a link character format to the word or phrase makes the link appear active, or clickable, in the generated output. If you do not know which character format to use for links, consult the Stationery designer.
3. ***If you want to create a link to a PDF file***, complete the following steps:
- a. In the FrameMaker menu, go to **Special > Hypertext**
  - b. From the Command dropdown menu, chose **Message Client**
  - c. In the Syntax text box, type message `openfile relative_path` where `relative_path` is the relative directory where you have your PDF located and then you add `example.pdf` to this path
4. Save your Adobe FrameMaker source document.

## Working with Tables in FrameMaker

This section explains how to prepare tables in source documents for output generation. Obtain your latest templates and apply the latest table formats from the template to tables in your source documents. If your tables do not have header rows, create a header row for each table. If your tables do not have footer rows, create a footer row for each table.

## Applying Table Formats in FrameMaker

Table formats define the appearance of your tables, and ePublisher uses table formats to define the appearance of tables in generated output. When you work with tables in your Adobe FrameMaker source documents, ensure you apply the correct table formats to your tables. The Stationery designer defines the table formats you can use in your Adobe FrameMaker source documents in the Adobe FrameMaker templates you associate with your Adobe FrameMaker source

documents. If you want to specify a different table format for sets of tables in your generated output, first ensure the different table format you want to apply is available in your Adobe FrameMaker source document. Then apply the different table format to tables in your Adobe FrameMaker source documents as appropriate.

For example, you may have a small set of tables that contain information about a specific component in a product. If you decide you want to modify the appearance of these tables in your generated output by specifying that the tables associated with this component display with a yellow background in your generated output, apply a table format available in your Adobe FrameMaker source document that the Stationery designer created to meet this requirement. When you generate output, the Stationery designed by the Stationery designer specifies that any tables created with a table format configured to display tables with a yellow background display in your output with a yellow background.

## **Creating Table Header Rows in FrameMaker**

Most tables in Adobe FrameMaker source documents include header rows, because by default Adobe FrameMaker allows you to quickly and easily specify the number of header rows in a table when you create a table. However, if your tables do not have header rows, consider adding table header rows to tables in your Adobe FrameMaker source documents. Using table header rows allows you to more tightly control the appearance of tables when you generate output. For example, if you use header rows, you can specify one appearance for header rows in your generated output, and a different appearance for body rows in your generated output.

The following procedure provides an example of how to create table header rows in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating table header rows in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

### To create a table header row in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the table for which you want to create a table header row.
2. Insert your cursor in the top row of the table.
3. On the **Table** menu, click **Add Rows or Columns**.
4. Click **Add 1 Row**.
5. Select **To Heading** from the list.
6. Click **Add**. Adobe FrameMaker inserts a header row into the table.
7. Type the text for the header row into the table.
8. Delete any existing rows in the text that contain the text you typed into the new table header row as needed.

## Creating Table Footer Rows in FrameMaker

Most tables in Adobe FrameMaker source documents include footer rows, because by default Adobe FrameMaker allows you to quickly and easily specify the number of footer rows in a table when you create a table. However, if your tables do not have footer rows, consider creating footer rows in your source documents in order to quickly and easily specify the appearance that you want for your table footer rows in your generated output. For example, if you use footer rows in conjunction with header rows, you can specify one appearance for footer rows in your generated output, and then different appearances for header rows and table body rows in your generated output.

The following procedure provides an example of how to create table footer rows in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating table footer rows in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create a table with a footer row in Adobe FrameMaker

1. In your Adobe FrameMaker source document, locate the table for which you want to create a table footer row.
2. Insert your cursor in the bottom row of the table.
3. On the **Table** menu, click **Add Rows or Columns**.
4. Click **Add 1 Row**.
5. Select **To Footing** from the list.
6. Click **Add**. Adobe FrameMaker inserts a footer row into the table.

## Working with Images in FrameMaker

Many writers include images when producing documents using Adobe FrameMaker. Most writers typically insert images into Adobe FrameMaker source documents in one of the following ways:

- Copying images directly into in Adobe FrameMaker source documents, also known as embedding images
- Importing images by reference, which creates a link to the source image in the Adobe FrameMaker source documents

If you copy an image into an Adobe FrameMaker source document, Adobe FrameMaker copies, or embeds, the image in the Adobe FrameMaker source document, and the image becomes a part of the document.

If you import an image by reference in Adobe FrameMaker source documents, Adobe FrameMaker creates a link to the image and displays the image in the Adobe FrameMaker source document. The link becomes a part of the document, but the actual image file is not inserted into the document, although the actual image files is displayed in the document. If you update the image file referenced by the link, Adobe FrameMaker displays the updated image referenced by the link automatically.

There are benefits and drawbacks to copying images directly into Adobe FrameMaker source documents and importing images by reference.

For example, if you copy images into Adobe FrameMaker source documents, you do not have worry about breaking the reference, or link, between the Adobe FrameMaker source documents and the image files. If you import the image by reference into Adobe FrameMaker source documents, you must ensure that you keep the same file structure for the image files in order to not break the references, or links, between the Adobe FrameMaker source document and the image file.

However, importing images by reference in Adobe FrameMaker source documents, rather than copying images into the source documents, provides the following benefits:

- You can update image files without recopying the image into your Adobe FrameMaker source documents.
- If you have one image used in multiple places, you can update the image in one place, rather than recopying the image into multiple places.
- You can manage your documentation files and image files separately, which makes organizing images easier.
- Source documents with images imported by reference in Adobe FrameMaker are smaller than source documents with copied images.

When you work with Adobe FrameMaker source documents that you will use to generate output, ensure you follow the guidelines specified by the Stationery designer for the following items:

- Method used to insert images
- Correct DPI to use for inserted images
- Correct image file format to use for inserted images

## Inserting Images in FrameMaker

Before you insert images into Adobe FrameMaker source documents you plan to use to generate output, review image considerations. For more information, see “Working with Images in FrameMaker”.

When you insert images into Adobe FrameMaker source documents, insert the image into an anchored frame. The anchored frame allows you to specify the image alignment and position. For more information about anchored frame options, see the Adobe FrameMaker documentation.

The following procedure provides an example of how to use an anchored frame to insert an image by reference in an unstructured Adobe FrameMaker source document using Adobe FrameMaker 7.2. Steps for inserting an image by reference in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To insert an image in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, insert your cursor on a blank line below the paragraph where you want to insert your image.

**Note:** Inserting an image on a blank line allows you to customize the paragraph tag applied to the line. Many Adobe FrameMaker templates have a special paragraph tag for you to use when you insert graphics. This paragraph tag specifies the space above and below the paragraph and the alignment of the inserted image.

2. Apply the appropriate paragraph format for images to the blank anchored frame line. For more information about the correct paragraph format to use for image anchored frame lines, consult with the Stationery designer.
3. On the **Special** menu, click **Anchored Frame**.
4. Specify the position, alignment, and size of the anchored frame, and then click **New Frame**. Adobe FrameMaker inserts an empty anchored frame into the source document.

For more information about anchored frame options, see the Adobe FrameMaker documentation.

5. On the **File** menu, click **Import > File**.
6. Click **OK** to continue.
7. Browse to the location of the file you want to import and select the file.
8. Click **Import by Reference**, and then click **Import**.
9. Specify the size of the graphic.

**Note:** Most writers do not select the **Fit in Selected Rectangle** option. This option resizes the image to fit inside the selected anchor or graphic frame. When you select this option, Adobe FrameMaker sets the DPI to unknown, and the imported image is usually distorted.

- **If you want to use the DPI from the graphic**, do not change the setting in the **Custom dpi** field. The number in the **Custom dpi** field is the DPI of the imported graphic.
- **If you want to change the size of the graphic**, click the button for the dpi setting you want to specify.

**Note:** If you do not use the same DPI setting as the source image, the image in your output may be distorted.

**10.** Click **Set**. Adobe FrameMaker imports the image into the source document.

**11. *If you have white space between the graphic and the anchored frame,*** you can shrink-wrap the frame by completing the following steps:

**Shrinking-wrapping** an anchored frame removes the white space between the graphic and the anchored frame and changes the anchoring position of the frame to **At Insertion Point** and displays the frame 0 points above the baseline of the text. If the anchored frame is on the same line as the text, the 0 point baseline can cause the image to cover the text of the preceding lines. For this reason, many writers prefer to insert images on a separate line below the text. The image may also be distorted if you don't shrink wrap the image.

- a.** Click the anchored frame or the image in the anchored frame.
- b.** Press **ESC+M+P**. Adobe FrameMaker shrinks or expands the anchored frame to fit the contents of the anchored frame and positions the anchored frame according to the paragraph pagination settings.

After you insert an image, you can assign alternate text or a long description to the image. For more information, see "Assigning Alternate Text to Images and Image Maps in FrameMaker" and "Assigning Long Descriptions to Images in FrameMaker".

## Creating Image Links in FrameMaker

You can create image links that allow users who click the image to link to content in another location. For example, if you include your company logo in a source document, you can define a link for the logo so that when users click the logo, they link to your company home page.

The following procedure provides an example of how to create an image link in Adobe FrameMaker source documents using Adobe FrameMaker 7.2. Steps for creating an image link in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create an image link in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, insert the image for which you want to create an image link. For more information, see “Inserting Images in FrameMaker”.
2. ***If you want to link to content in a different location in your source document***, create a named destination for the link by completing the following steps:

**Note:** You do not need to perform these steps if you want to link to content on a Web site.

- a. Locate the link target in your source document.
- b. On the **Special** menu, click **Marker**.
- c. In the **Marker Type** field, click **Hypertext**.
- d. In the **Marker Text** field, `newlink linkname` or `newlink filename:linkname`, where `linkname` is the name of the named destination for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.

**Note:** To make maintenance easy, create short link names that use alphanumeric, lowercase characters.

- e. Click **New Marker**.
3. In the anchored frame that contains the image for which you want to create a link, draw a text frame that covers the entire clickable region by completing the following steps:
    - a. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
    - b. Click the **Text Frame** icon.
    - c. Drag the cursor across the image to draw a text frame over the image.
    - d. In the Create New Text Frame window, in the **Number** field, type `1`, and then click **Set**.
    - e. Click outside of the image, and then insert your cursor in the text frame.
  4. In the text frame, insert a Hypertext marker that specifies the destination of the link by completing the following steps:
    - a. On the **Special** menu, click **Marker**.



- b. In the **Marker Type** field, select **Hypertext** from the list.
- c. ***If you want link to content in a different location in your source document***, use the named destination link you created in step 2. In the **Marker Text** field, type `gotolink linkname` or `gotolink filename:linkname`, where `linkname` is the name of a link target you created previously, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.
- d. ***If you want to link to a page on a Web site***, in the **Marker Field**, type `message URL web address`, where `web address` is the URL of the web page you want to open when users click the image.

**Note:** For more information about using the `gotolink` and `message URL` commands, see the Adobe FrameMaker documentation.

- e. Click **New Marker**.
5. Save your Adobe FrameMaker source document.
  6. Generate output for your project. For more information, see “Generating Output”.
  7. In Output Explorer, verify ePublisher created the image link using the link information you specified on the page by clicking on the image. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

## Creating Clickable Regions for Image Maps in FrameMaker

An image map can be a single image separated with clickable regions or a composite image made up of multiple images grouped together, yet still separated with clickable regions. For example, you could create an image of the countries of Europe and then define an image map for the image that allows users to link to a topic about each country when they click on an area of the image. User can click France to see information about France, Italy to see information about Italy, and so on.

When you define an image map, you can also define alternate text for each clickable region. For example, you might define alternate text for the Italy region as “Click here for more information about Italy.” For more information about assigning alternate text to image maps, see “Assigning Alternate Text to Images and Image Maps in FrameMaker”.

## Creating Image Maps for Single Images in FrameMaker

You create image maps for single images in Adobe FrameMaker source documents using text frames and hyperlinks. In Adobe FrameMaker, a hyperlink consists of a link and a link target, or named destination. A **named destination** is a unique identifier for a location in the document.

You can also create an image map for a composite image in an Adobe FrameMaker source document. For more information about creating composite images, see “Creating Image Maps for Composite Images in FrameMaker”.

The following procedure provides an example of how to create an image map for a single image in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating an image map for a single image in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

**To create an image map for a single image in an Adobe FrameMaker source document:**

1. In your Adobe FrameMaker source document, insert the image you want to use for your image map into an anchored frame. For more information, see “Inserting Images in FrameMaker”.
2. ***If you want to link to content in a different location in your source document***, create a named destination for the link for each area of the image map by completing the following steps:

**Note:** You do not need to perform these steps if you want to link to content on a Web site.

- a. Locate the link target in your source document.
- b. On the **Special** menu, click **Marker**.
- c. In the **Marker Type** field, click **Hypertext**.
- d. In the **Marker Text** field, `newlink /linkname` or `newlink filename:linkname`, where `linkname` is the name of the named destination for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.

**Note:** To make maintenance easy, create short link names that use alphanumeric, lowercase characters.

- e. Click **New Marker**.
3. In the anchored frame that contains the image for which you want to create an image map, draw a text frame that covers each region of the image where you want users to be able to click by completing the following steps:
    - a. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
    - b. Click the **Text Frame** icon.
    - c. Drag the cursor over the portion of the image for which you want to create a clickable area.
    - d. In the Create New Text Frame window, in the **Number** field, type `1`, and then click **Set**.
  4. In the text frame, insert a Hypertext marker that specifies the destination for the clickable region by completing the following steps:
    - a. Insert your cursor into the text frame.

- b. On the **Special** menu, click **New Marker**.
- c. ***If you want to link to content in a different location in your source document***, use a named destination link you created in step 2. In the **Marker Text** field, type `gotolink linkname` or `gotolink filename:linkname`, where `linkname` is the name of a link target you created previously, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.
- d. ***If you want to link to a page on a Web site***, in the **Marker Text** field, type `message URL web address`, where `web address` is the URL of the web page you want to open when users click the image.

**Note:** For more information about using these commands, see the Adobe FrameMaker documentation.

- e. Click **New Marker**.
5. Save your Adobe FrameMaker source document.
  6. Generate output for your project. For more information, see “Generating Output”.
  7. In Output Explorer, verify ePublisher created the image map using the link information you specified by clicking on the page that contains the image map and then clicking on each area of the image where you created a link. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

## Creating Image Maps for Composite Images in FrameMaker

You can create composite images by inserting the composite images into an anchored frame and then inserting text frames that contain the link you want users to go to when they click an area of a composite image.

The following procedure provides an example of how to use Hotspots in Adobe FrameMaker to create image maps for composite images in source documents.

**To create an image map for a composite image using Hotspots in an Adobe FrameMaker source document:**

1. In your Adobe FrameMaker source document, insert each image you want to use for your image map into an anchored frame. For more information, see “Inserting Images in FrameMaker”.
2. ***If you want to link to content in a different location in your source document***, create a named destination for the link for each area of the image map by completing the following steps:

**Note:** You do not need to perform these steps if you want to link to content on a Web site.

- a. Locate the link target in your source document.
- b. On the **Special** menu, click **Marker**.
- c. In the **Marker Type** field, click **Hypertext**.
- d. In the **Marker Text** field, `newlink linkname` or `newlink filename:linkname`, where `linkname` is the name of the named destination for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.

**Note:** To make maintenance easy, create short link names that use alphanumeric, lowercase characters.

- e. Click **New Marker**.
3. In the anchored frame that contains the image for which you want to create an image map, draw a text frame that covers each region of the image where you want users to be able to click by completing the following steps:
    - a. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
    - b. Click the **Text Frame** icon.
    - c. Drag the cursor over the portion of the image for which you want to create a clickable area.
    - d. In the Create New Text Frame window, in the **Number** field, type `1`, and then click **Set**.
  4. For each text frame, set the Hotspot properties that specifies the destination for the clickable region and optionally set its tooltip text by completing the following steps:

- a. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
  - b. Click the **Select Object** icon.
  - c. Click on the text frame so that you can set its hotspot properties.
  - d. With the text frame selected, right-click and select **Hotspot Properties...** from the menu.
  - e. ***If you want to link to content in a different location in your source document***, under **Destination** select the **Document** radio button. Then select either **Current** or the name of your target source document. Then click the marker with the named destination you created previously in step 2.
  - f. ***If you want to link to a page on a Web site***, under **Destination** select the **URL** radio button. Then type the complete web address (including protocol such as http://) in the adjacent text box of the web page you want to open when users click the image.
  - g. ***If you want tooltip text to be available in the output***, under the **Tooltip Text** label, enter the text you want to appear as the tooltip.
  - h. Click the **Save** button on the **Hotspot** dialog.
5. Save your Adobe FrameMaker source document.
  6. Generate output for your project. For more information, see “Generating Output”.
  7. In Output Explorer, verify ePublisher created the image map using the link information you specified by clicking on the page that contains the image map and then clicking on each area of the image where you created a link. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

If your version of Adobe FrameMaker does not provide for Hotspot properties, you can still create image maps for composite images by following this alternate example of how to create image maps for composite images in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating image maps for composite images in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

**To create an image map for a composite image in an Adobe FrameMaker source document using hypertext markers:**

1. In your Adobe FrameMaker source document, insert each image you want to use for your image map into an anchored frame. For more information, see “Inserting Images in FrameMaker”.
2. ***If you want to link to content in a different location in your source document***, create a named destination for the link for each area of the image map by completing the following steps:

**Note:** You do not need to perform these steps if you want to link to content on a Web site.

- a. Locate the link target in your source document.
- b. On the **Special** menu, click **Marker**.
- c. In the **Marker Type** field, click **Hypertext**.
- d. In the **Marker Text** field, `newlink /linkname` or `newlink filename:linkname`, where `linkname` is the name of the named destination for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.

**Note:** To make maintenance easy, create short link names that use alphanumeric, lowercase characters.

- e. Click **New Marker**.
3. In the anchored frame that contains the image for which you want to create an image map, draw a text frame that covers each region of the image where you want users to be able to click by completing the following steps:
    - a. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
    - b. Click the **Text Frame** icon.
    - c. Drag the cursor over the portion of the image for which you want to create a clickable area.
    - d. In the Create New Text Frame window, in the **Number** field, type `1`, and then click **Set**.
  4. In each text frame, insert a Hypertext marker that specifies the destination for the clickable region by completing the following steps:
    - a. Insert your cursor into the text frame.

- b. On the **Special** menu, click **Marker**.
- c. ***If you want to link to content in a different location in your source document***, use the named destination link you created in step 2. In the **Marker Text** field, type `gotolink linkname` or `gotolink filename:linkname`, where `linkname` is the name of a link target you created previously, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.
- d. ***If you want to link to a page on a Web site***, in the **Marker Text** field, type `message URL web address`, where *web address* is the URL of the web page you want to open when users click the image.

**Note:** For more information about using these commands, see the Adobe FrameMaker documentation.

- e. Click **New Marker**.
5. Save your Adobe FrameMaker source document.
  6. Generate output for your project. For more information, see “Generating Output”.
  7. In Output Explorer, verify ePublisher created the image map using the link information you specified by clicking on the page that contains the image map and then clicking on each area of the image where you created a link. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

## Assigning Image Scales in FrameMaker

When ePublisher converts images inserted into your source documents, it can scale images to make them display larger or smaller in your generated output. By default, ePublisher uses the scaling factor applied to images as specified by the image format you apply to each image. For example, if you apply an image format to images and the Stationery designer defined the image format to scale images to 80% of their original size, all images that have this image format applied to them will be scaled to 80% in the generated output.

Typically, using the standard scaling factor specified in the image format is sufficient. Occasionally, however you may want to override the scaling factor for an individual image. For example, while most `.gif` images scale to 80%, you may have one large image that you want scaled to 60% in your generated output. You can manually override the standard scaling factor specified in your Stationery for a specific image by using the `GraphicScale` marker.

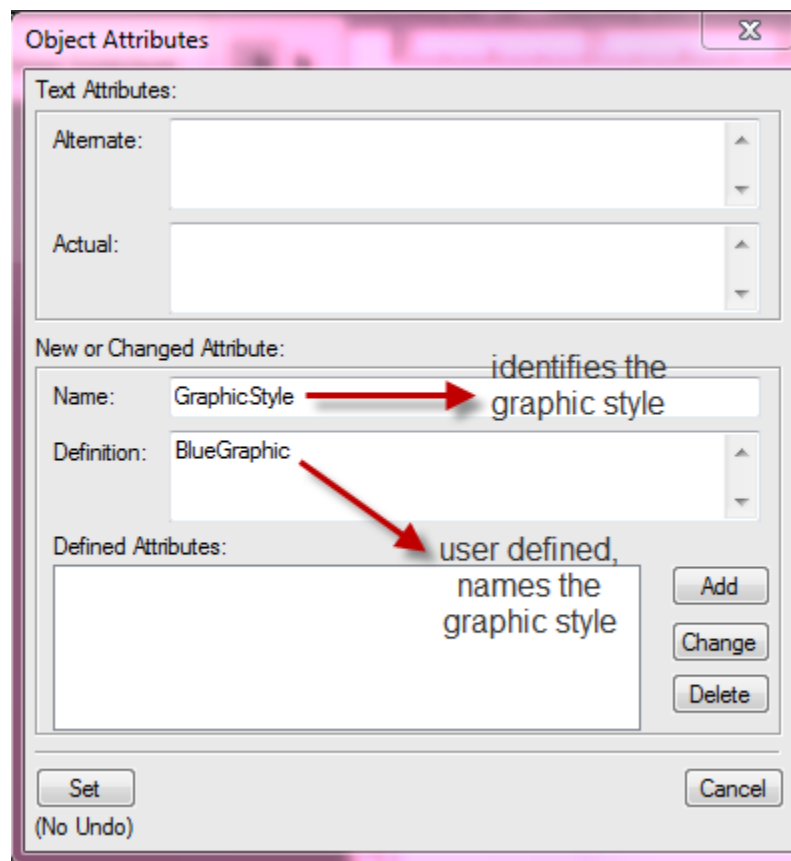


To assign a scale to a specific image, your Stationery and template must have the GraphicScale marker type configured. Your output format must also support scaling by image.

The following procedure provides an example of how to specify image scaling for an image in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying image scaling for an image in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To specify an image scale for an image in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the anchored frame for the image for which you want to specify image scaling.
2. Right-click on the image anchor. Make sure this is the entire anchor, not just the graphic itself
3. Click on **Object Properties** and then click **Object Attributes**. Identify the **GraphicScale** according to the box below. Assign the desired style name in the attribute value box.



4. Click **Add**, then **Set**, and then **Set**. Adobe FrameMaker may prompt you to approve the change as the operation cannot be undone
5. Save your Adobe FrameMaker source document.
6. Scan this document in ePublisher so that the **GraphicScale** marker will show up under "Marker Styles". This will configure the correct marker behavior for processing.
7. Generate output for your project. For more information, see "Generating Output".

8. In Output Explorer, verify ePublisher created the image using the image scale you specified in the **GraphicScale** marker by clicking on the page that contains the image for which you specified image scaling. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

## Assigning Image Styles in FrameMaker

Typically you do not need to specify an image style for images when you generate output. By default, each image generated by ePublisher is associated with the default image style defined in the Stationery. However, if you want to change the image style of one image or a small set of images, you can specify the image style you want to use for an image in your source document using the **GraphicStyle** marker type.

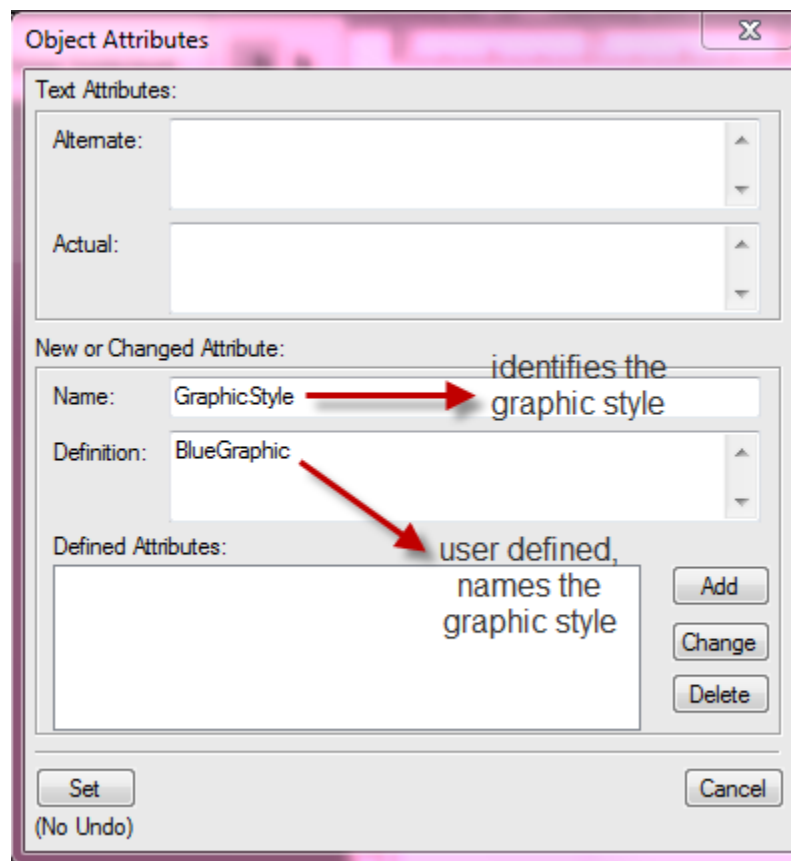
For example, if you want to specify a yellow border around a set of screen shot images that illustrate a particular piece of product functionality, you can specify that each of the screen shots images in the set have a yellow border around them through the use of the **GraphicStyle** marker type.

To assign a style to a specific image, your Stationery and FrameMaker template must have the **GraphicStyle** marker type configured. Your output format must also support specifying image styles.

The following procedure provides an example of how to specify image styles for images in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying image styles for images in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To specify an image style for an image in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the anchored frame for the image for which you want to specify an image style.
2. Right-click on the image anchor. Make sure this is the entire anchor, not just the graphic itself
3. Click on **Object Properties** and then click **Object Attributes**. Identify the **GraphicStyle** according to the box below. Assign the desired style name in the attribute value box.



4. Click **Add**, then **Set**, and then **Set**. Adobe FrameMaker may prompt you to approve the change as the operation cannot be undone
5. Save your Adobe FrameMaker source document.
6. Scan this document in ePublisher so that the **GraphicStyle** marker will show up under "Graphic Styles"
7. Generate output for your project. For more information, see "Generating Output".

8. In Output Explorer, verify ePublisher created the image using the image style you specified by clicking on the page that contains the image for which you specified an image style and verifying ePublisher applied the image style you specified in the generated output. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

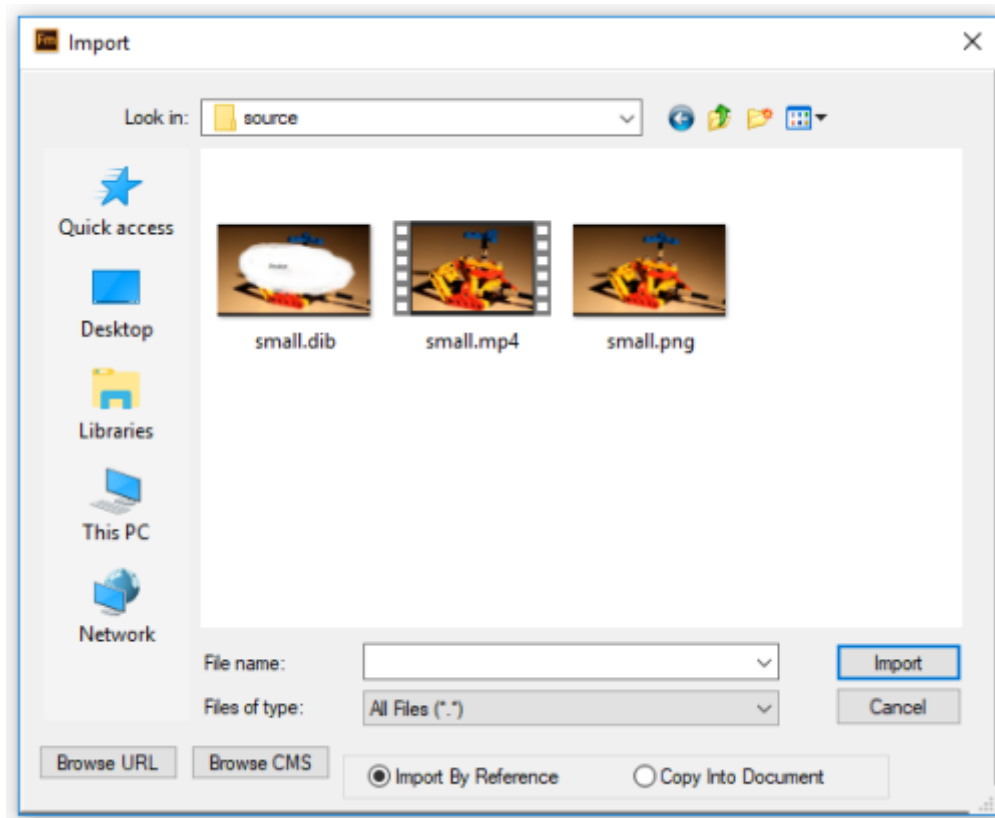
## Working with Videos in FrameMaker

Occasionally, writers include videos when producing documents using Adobe FrameMaker.

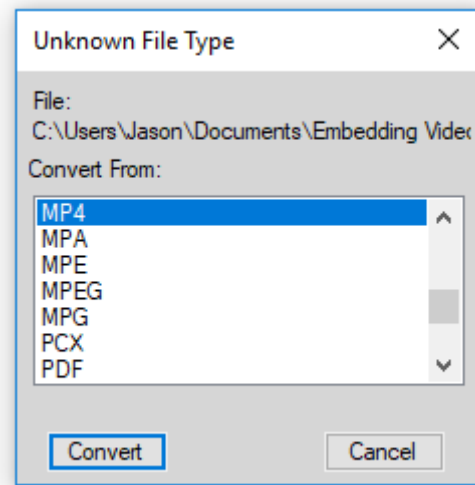
**Note:** If the video is not found, ensure the correct path to the video is specified in the FrameMaker document.

## To include a video in an Adobe FrameMaker source document

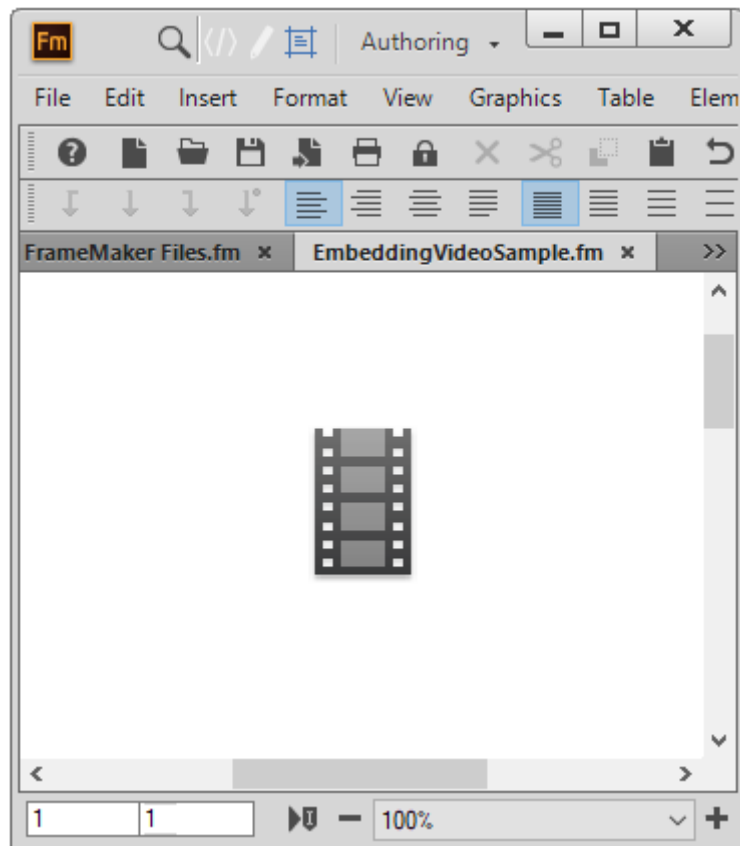
1. In your Adobe FrameMaker source document, click on File from the tool bar, select Import File, then a dialog box will appear.



2. Locate the video file in the dialog box.
3. Select the video file, click the import button.
4. Select which file type to convert from. This example uses an MP4 formatted video selected, so MP4 format should be selected in the dialog.

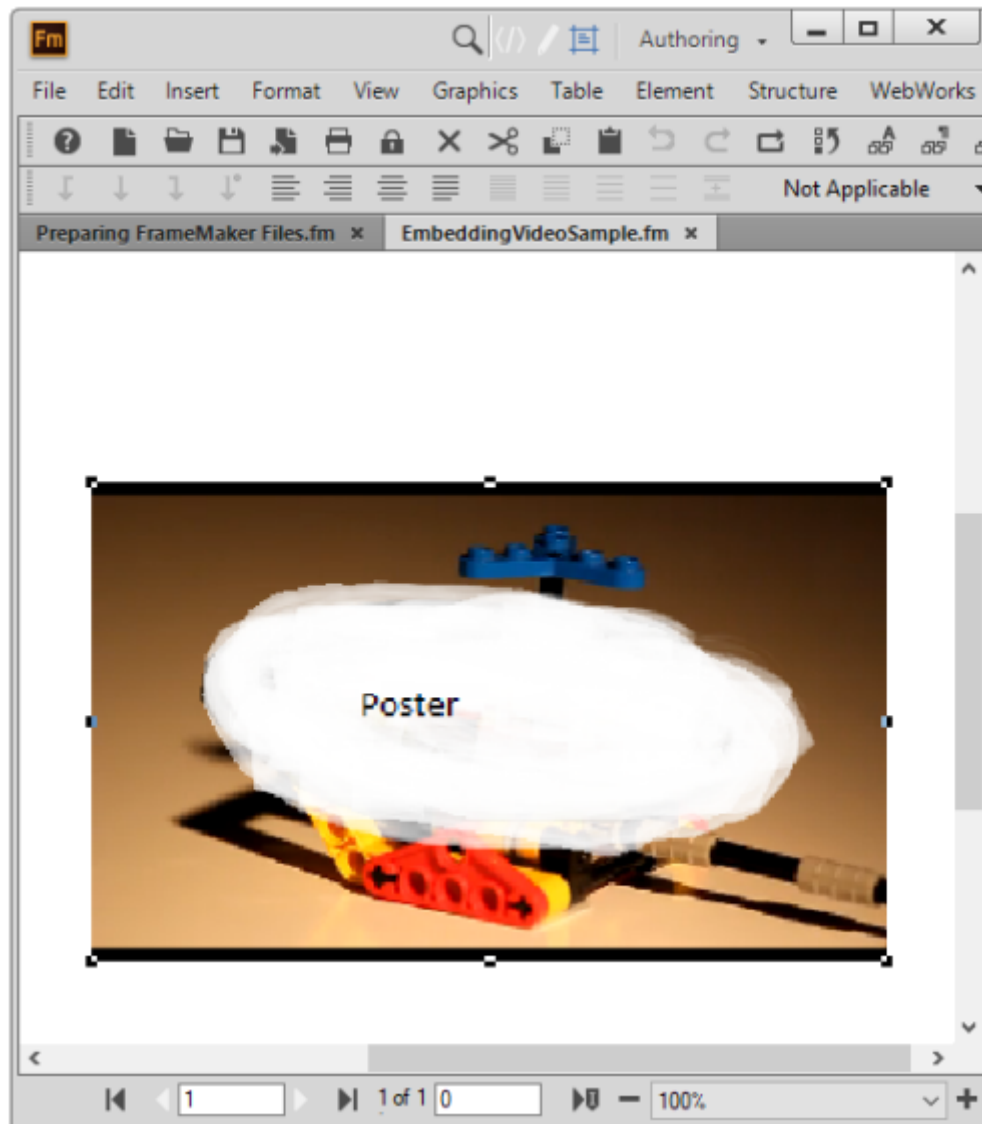


5. Select the Imported Graphic Scaling. For this example 72 dpi was used. After making the selection, the video will appear in the FrameMaker document.



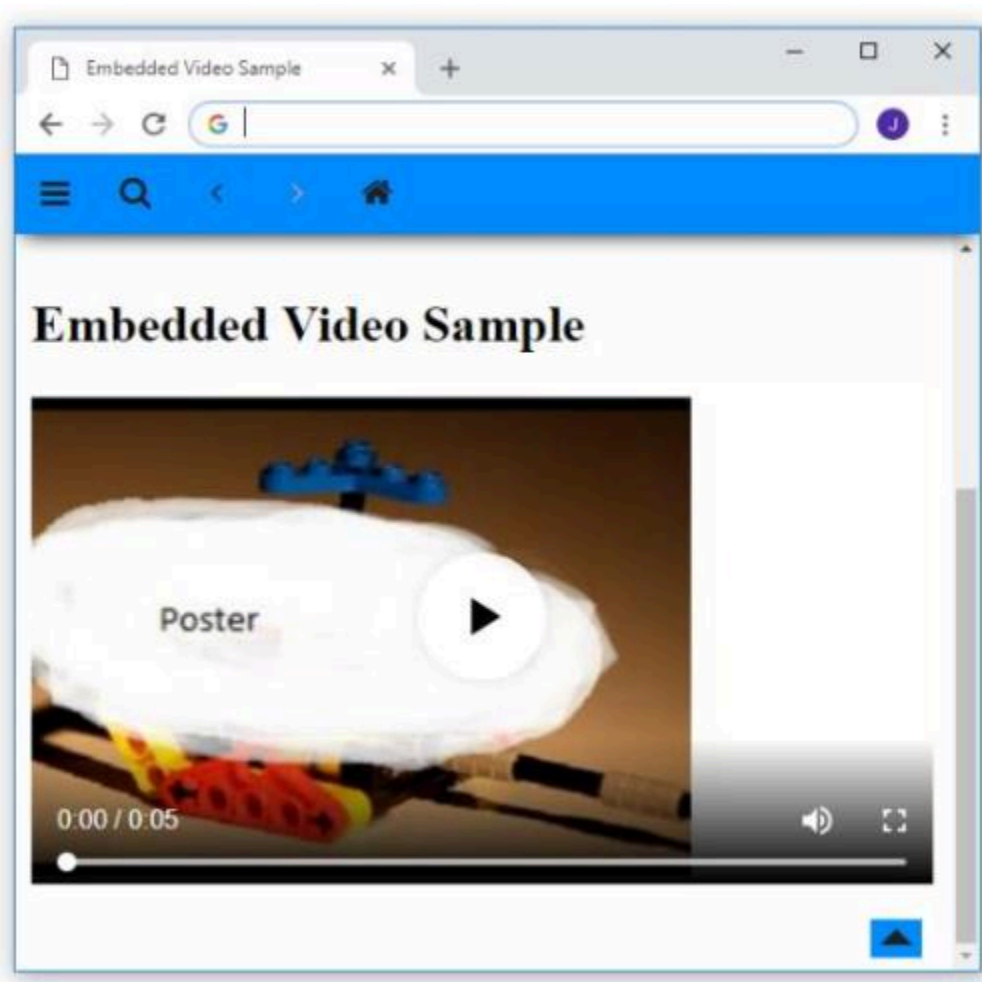
6. Add a poster image to the video file. This step is optional.

- a. Click on File, then select Import File. Select the image to use as the poster file image.
- b. Click the Replace button.
- c. Select the Imported Graphic Scaling. 72 dpi was used in the example.



- d. Position the video with your mouse.
  - e. Save the File
7. Add the document to your Reverb 2.0 project and generate.





## Creating Index Entries in FrameMaker

An index lists the terms and topics discussed in a document and the page or pages on which they appear. An online index provides the user with a point-and-click resource for quickly navigating online content.

ePublisher uses the same native index entry features used in source documents to create a printed index to create an online index. If you include index entries in your source documents, ePublisher detects the index entries and uses the index entries to create an online index in your generated output.

Adobe FrameMaker inserts index entries as Index markers. To create index entries in an Adobe FrameMaker source document, insert Index markers into your Adobe FrameMaker source document. ePublisher then uses the Index markers to create an online index when you generate output.

Before you insert index entries, verify with the Stationery designer that your Stationery is configured to support online index generation. By default, ePublisher enables online index generate for output, but this functionality can be disabled in your Stationery by the Stationery designer. Also confirm that your output format supports online index creation.

Talk with the Stationery designer and other writers about the standard location and method you should use when you insert Index markers into your Adobe FrameMaker source documents. For example, some writers prefer to insert index entries into topic headings, while other writers prefer to insert index entries on the first line of the paragraph that contains the indexed term or terms. Some writers prefer to create one Index marker for each term, while other writers prefer to create one Index marker and then type all index terms associated with a paragraph into one Index marker.

The following procedure provides an example of how to inset index entries in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for inserting index entries in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To insert an index entry in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, insert your cursor in the location where you want to create an index entry.
2. On the **Special** menu, click **Marker**.
3. In the **Marker Type** field, select **Index** from the drop-down list.
4. In the **Marker Text** field, type the text you want to specify for the index entry.

**Note:** Following are some common ways writers can create index entries in Adobe FrameMaker. For more information about creating index entries, see the Adobe FrameMaker Help.

- ***If you want to specify multiple index entries in the marker,*** separate each index entry with a semicolon (;) character.

For example, type `index; table of contents; headings; footers`

- ***If you want to create a subentry,*** separate the primary and secondary entry with a colon (:).

For example, type `index:creating; index:generating;`

- ***If you want to create See references,*** insert the entry but use the `<$npage>` command to suppress the page number.

For example, type `document, See source document <$npage>`

- ***If you want to create see references with the word See italicized,*** use a character tag inside the Index marker and the `<Default Para Font>` tag to turn off the character tagging.

For example, type `document, <Emphasis>See <Default Para Font>source documents<$npage>`

- ***If you want to create See also references,*** use alternate text that specifies how Adobe FrameMaker sorts the see also reference.

For example, type `document, <Emphasis>See also<Default Para Font>source documents<$npage>[document:aa]`

The text in brackets at the end of the entry controls where Adobe FrameMaker displays the text in the entry. In this example, the `aa` text ensures Adobe FrameMaker displays the entry as the first subentry under document.

5. Click **New Marker**.
6. After you insert you index entries, save your Adobe FrameMaker source document.
7. Generate output for your project. For more information, see “Generating Output”.
8. In Output Explorer, verify ePublisher created the index correctly by clicking on the page or tab that displays the index and then clicking on the index entries. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

## Using Variables in FrameMaker

A variable serves as a placeholder for information that may change frequently. Using variables in source documents allows you to quickly and easily control the content in your generated output. When you change the value of a variable in an ePublisher project, it changes the value in only your generated output. The variable value does not change in your source document.

Once you insert variables into your source documents, whenever the value of a item defined by a variable needs to change, you can make the change in a single location, rather than searching and replacing for all instances of the item. For example, you can use variables in the following ways:

- If you have publication dates or release dates in your source documents that you need to update periodically, you can set up the date as a variable.
- If you work with products that have names or versions that frequently change, you can set up variables for product names and versions.
- If you need to produce documentation sets for a product with multiple brands, you can use variables to help you produce documentation for each different brand using the same set of source files.

## Importing or Creating Variables in FrameMaker

When you work with Adobe FrameMaker source documents, typically you import variables into your Adobe FrameMaker source documents from an Adobe FrameMaker template. The Adobe FrameMaker template contains variables defined by the Stationery designer.

Typically you should not need to create variables in your Adobe FrameMaker source files if you use an Adobe FrameMaker template created by a Stationery designer. However, in some cases you may need to create a variable in an Adobe FrameMaker

source document if you do not have an Adobe FrameMaker template that includes a variable you need for your project.

The following procedure provides an example of how to import or create variables in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for importing or creating variables in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To import variables or create a variable in an Adobe FrameMaker source document

1. Open your Adobe FrameMaker source file.
2. ***If you want to import variables into your Adobe FrameMaker source file from an Adobe FrameMaker template,*** complete the following steps:
  - a. Open the Adobe FrameMaker template that contains the variables you want to import.
  - b. On the **File** menu, click **Import > Formats**.
  - c. In the **Import from Document** field, select the Adobe FrameMaker template that contains the variables you want to import from the list.
  - d. In the **Import and Update** field, select only the **Variable Definitions** check box.
  - e. Click **Import**.
  - f. Click **OK** to confirm the operation.
3. ***If you want to create a variable in your Adobe FrameMaker source file,*** complete the following steps:
  - a. On the **Special** menu, click **Variables**.
  - b. Click **Create Variable**.
  - c. In the **Name** field, type a name for the variable. Variable names are case sensitive. For example, VariableName and variablename are different variables.
  - d. Insert your cursor in the **Definition** field.
  - e. In the **Character Formats** field, select a character format for the variable and then type a value for the variable. For more information about specifying character formats for variables, see the Adobe FrameMaker Help.
  - f. Click **Add**. Adobe FrameMaker adds the variable to the list of variables. The variable is the value that Adobe FrameMaker displays in your Adobe FrameMaker source document.
  - g. Click **Done**.
4. Save your Adobe FrameMaker source file.

# Inserting Variables into FrameMaker

You can insert a variable into a source document after you import the variables into your source document. If you want to use a variable that is not defined in an Adobe FrameMaker template, you must create the variable in your source document before you can insert it. For more information about importing or creating variables, see “Importing or Creating Variables in FrameMaker”.

The following procedure provides an example of how to insert variables in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for inserting variables in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

### To insert a variable into an Adobe FrameMaker source document

1. Open the Adobe FrameMaker source document into which you want to insert a variable.
2. Place your cursor in the location where you want to insert the variable.
3. On the **Special** menu, click **Variable**.
4. In the **Variable** field, select the variable you want to insert from the list, and then click **Insert**. Adobe FrameMaker inserts the variable.

## Changing Variable Values in FrameMaker

You can change the value assigned to a variable in an Adobe FrameMaker source document.

The following procedure provides an example of how to change variable values in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for changing variable values in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.



### To change a variable value in an Adobe FrameMaker source document

1. Open the Adobe FrameMaker source document that contains the variable with a value you want to change.
2. On the **Special** menu, click **Variable**.
3. In the **Variable** field, select the variable with the value you want to change.
4. Click **Edit Definition**.
5. In the **Definition** field, edit the variable value.
6. Click **Done**. Adobe FrameMaker updates the variable value in each place in your source document where you inserted the variable.
7. Click **Done** again to close the window.

## Deleting Variables in FrameMaker

Delete a variable in an Adobe FrameMaker source document when you no longer want to use the variable. Before you delete a variable, ensure you search for the variable and delete or replace all references to the variable. If your source document still contains a reference to a variable after you delete it, Adobe FrameMaker prompts you to convert references to the variable in your source document to editable text.

The following procedure provides an example of how to delete variables in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for deleting variables in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To delete a variable in an Adobe FrameMaker source document

1. Open the Adobe FrameMaker source document that contains the variable you want to edit.
2. Search for and replace all references to the variable you want to delete in the source document by completing the following steps:
  - a. On the **Edit** menu, click **Find/Change**.
  - b. In the **Find** field, select **Variable of Name** from the list.
  - c. In the field next to the **Find** field, type the name of the variable you want to delete.
  - d. Click **Find**.
  - e. Delete each variable you find or replace the variable with a different variable as appropriate.
3. On the **Special** menu, click **Variable**.
4. In the **Variable** field, select the variable you want to delete.
5. Click **Edit Definition**.
6. In the **User Variables** field, ensure the variable you want to delete is selected, and then click **Delete**.
7. Click **Done**.
8. Click **OK** to confirm the operation.

## Using Conditions in FrameMaker

Conditions allow you to show or hide information in your source documents and in your online output. You apply conditions to the content in your source documents, and then you set the visibility for those conditions either in your source documents or in your ePubublisher project.

For example, your source documents might contain some content that should be displayed in only the printed version and other content that should be displayed in only the online version. You can use the same set of source documents for both printed and online versions through the use of conditions. You can create one condition called **PrintOnly** specifically for printed content, and then you can create another condition called **OnlineOnly** specifically for online content. After you create the **PrintOnly** and **OnlineOnly** conditions, you can apply them to the appropriate content in your source documents.

After you apply conditions in your source documents, ePublisher can use the conditions defined in your source document to control the visibility of content when it generates output. You can also change the visibility specified for any condition in your ePublisher project. Changing the visibility specified for any condition in your ePublisher project does not change the visibility specified for the condition in your source documents.

## **Creating Conditions in FrameMaker**

When you work with Adobe FrameMaker source documents, typically you import conditions into your Adobe FrameMaker source documents from an Adobe FrameMaker template. The Adobe FrameMaker template contains conditions defined by the Stationery designer.

Typically you should not need to create conditions in your Adobe FrameMaker source files if you use an Adobe FrameMaker template created by a Stationery designer. However, in some cases you may need to create a condition in your Adobe FrameMaker source documents if you do not have an Adobe FrameMaker template that includes a condition you need for a project. If you need to create a condition that is not available in your Adobe FrameMaker template, use native Adobe FrameMaker functionality to create the condition.

The following procedure provides an example of how to create conditions in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating conditions in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create a condition in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, on the **Special** menu, click **Conditional Text**.
2. Click **Edit Condition Tag**.
3. In the **Tag** field, type a name for the condition.

For example, if you want to create a condition for content that you want to display in only online content, type `OnlineOnly`. If you want to create a condition for content that you want to display in only printed content, type `PrintOnly`.

4. ***If you want to specify a style for the condition***, in the **Style** field, select the style you want to use for the condition from the drop-down list. Specifying a style for the condition allows you to more easily see the content tagged with the condition in your Adobe FrameMaker source documents. If you do not want to use a style for the condition, select **As Is**.
5. ***If you want to specify a color for the condition***, in the **Color** field, select a color for the condition from the drop-down list. Specifying a color for the condition allows you to more easily see the content tagged with the condition in your Adobe FrameMaker source documents. If you do not want to use a color for the condition, select **As Is**.
6. Click the **Set** to create the condition.

## Applying Conditions in FrameMaker

After you have imported conditions from your Adobe FrameMaker template or created conditions in your Adobe FrameMaker source document, you can apply conditions to content in your Adobe FrameMaker source documents. For more information about creating conditions in Adobe FrameMaker source documents, see “Creating Conditions in FrameMaker”.

The following procedure provides an example of how to apply conditions in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for applying conditions in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

### To apply a condition to content in an Adobe FrameMaker source document

1. In your Adobe Framemaker source document, select the content to which you want to apply the condition.
2. On the **Special** menu, click **Conditional Text**.
3. In the **Not In** list, select the condition you want to apply to the content.
4. Click the left arrow to move the condition from the **Not In** list to the **In** list.
5. Click **Apply** button to apply the condition.

## Removing Conditions in FrameMaker

If you no longer want to apply a condition to content in an Adobe FrameMaker source document, you can remove the applied condition from the content.

The following procedure provides an example of how to remove conditions from content in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for removing conditions from content in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

### **To remove a condition from content in an Adobe FrameMaker source document**

1. In your Adobe FrameMaker source document, select the content with the condition you want to remove.
2. On the **Special** menu, click **Conditional Text**.
3. Click **Unconditional**.
4. Click **Apply**.

## **Modifying Conditions in FrameMaker**

You can edit the name of a condition and change the color or style assigned to a condition in an Adobe FrameMaker source document.

The following procedure provides an example of modify conditions in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for modifying conditions in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To modify a condition in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, on the **Special** menu, click **Conditional Text**.
2. Select the condition you want to modify.
3. Click **Edit Condition Tag**.
4. *If you want to modify the name of the condition*, in the **Tag** field, type the new name for the condition.
5. *If you want to modify the style specified for the condition*, in the **Style** field, select the style you want to apply to the condition from the drop-down list. If you do not want to use a style, select **As Is**.
6. *If you want to modify the color specified for the condition*, in the **Color** field, select a color for the condition from the drop-down list. If you do not want to use a color, select **As Is**.
7. Click **Set** to modify the condition.

## Showing and Hiding Conditions in FrameMaker

You can show and hide conditions you applied in your Adobe FrameMaker source document. You can also show all of the conditions you applied in your Adobe FrameMaker source document. Showing all of the conditions applied allows you to see where all of the conditional content is in your Adobe FrameMaker source document.

The following procedure provides an example of how to show and hide conditions in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for showing and hiding conditions in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To show and hide conditions in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, on the **Special** menu, click **Conditional Text**.
2. Click **Show/Hide**.
3. **If you want to show all conditions**, click **Show All**. Showing all conditions is helpful when you are working with a document and you want to be sure you can see all of the content in the document.
4. **If you want to show a specific condition**, select it, and then click the left arrow to move it to the **Show** list on the left.
5. **If you want to hide a specific condition**, select it, and then click the right arrow to move it to the **Hide** list on the right.
6. Click **Set**.

## Using Passthrough Conditions in FrameMaker

A passthrough condition is a condition you apply to content that you do not want ePublisher to process when you generate output. For example, if you have embedded multimedia files in your source documents, such as Audio Video Interleave files (`.avi`) or Adobe Software Flash files (`.swf`), you can apply a passthrough condition to the code so that ePublisher does not process the code.

The following example shows `.avi` code to which you can apply a passthrough condition.

```
<embed src="sample.avi" width="400"
  height="300" pluginspage=";>
</embed>
```

The following example shows `.swf` code to which you can apply a passthrough condition.

```
<embed src="sample.swf" width="400"
  height="300" pluginspage="
http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=ShockwaveFlash";>
</embed>
```

If you have code in your Adobe FrameMaker source documents that you do not want ePublisher to process, create a passthrough condition and then apply the passthrough condition to the code.



Typically you use a passthrough condition defined in an Adobe FrameMaker template by a Stationery designer. You import this condition into your Adobe FrameMaker source document from an Adobe FrameMaker template. After you import the passthrough condition into your source document from the template, you apply the passthrough condition to the content as appropriate.

Typically you should not need to create a passthrough condition in your Adobe FrameMaker source file if you use an Adobe FrameMaker template created by a Stationery designer. However, in some cases you may need to create a passthrough condition in your Adobe FrameMaker source document if you do not have an Adobe FrameMaker template that includes a passthrough condition you need for a project. If you need to create a passthrough condition that is not available in your Adobe FrameMaker template, use native Adobe FrameMaker functionality to create the condition. For more information about creating a condition and applying a condition, see “Creating Conditions in FrameMaker” and “Removing Conditions in FrameMaker”.

You can also use Passthrough markers and the Passthrough paragraph styles and character styles options to insert content directly into your output without being transformed and coded for your output.

## **Deleting Conditions in FrameMaker**

Delete a condition in an Adobe FrameMaker source document when you no longer want to apply the condition to content in the source document.

The following procedure provides an example of how to delete conditions in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for deleting conditions in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

### To delete a condition in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, on the **Special** menu, click **Conditional Text**.
2. Select the condition you want to delete.
3. Click **Edit Condition Tag**.
4. Click **Delete**.

## Conditional Output Using Expressions in FrameMaker

Adobe FrameMaker 8.0 introduced ways to use Boolean Expressions (using the terms AND, OR, or Not) in conditional text, for example WebHelp AND PDF. To do this, you first create the conditions, and then use the Build Expression button to create this.

The following procedure provides an example of how to use Expressions in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 9. Steps for deleting conditions in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

### To build an Expression in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, on the **Special** menu, click **Conditional Text -> Show/Hide Conditional Text**
2. Click button **Build Expression**
3. Click desired condition and the arrow button to add it to the Expression
4. Separate the conditions by clicking the buttons for "AND" "OR" or "NOT"
5. When finished select Set so that this Expression will be able to be selected in the **Expression** dropdown located in the **Show/Hide Conditional Text** window
6. Go to **Special -> Conditional Text -> Show/Hide Conditional Text** and select the Show as per Expression radio button
7. Select the desired text you want to apply the expression and hit **Apply**
8. Save your source document.
9. Generate output for your project. For more information, see "Generating Output".

10. In Output Explorer, verify ePublisher created an output file using the file name you specified. For more information, see “Viewing Output in Output Explorer”.

## Specifying Output File Names in FrameMaker

By default, ePublisher automatically assigns file names to your generated output files for topics (pages) and for embedded image (graphic) output files.

**Note:** If you insert your images by reference in Adobe FrameMaker, ePublisher preserves the original file names. For more information, see “Working with Images in FrameMaker”.

You can customize this naming convention using one of the following methods:

- Inserting Filename markers into your source documents
- Specifying the topic (page) and image (graphic) naming patterns for ePublisher to use in the target settings for your output

This section explains how you can specify output file names in your FrameMaker source documents using Filename markers. For more information about using target settings to specify output file names using page and image naming patterns, see “Specifying Page, Image, and Table File Naming Patterns”.

To specify a file name for a page or image output file using Filename markers, your Stationery and FrameMaker template must have the Filename marker type configured.

The following procedure provides an example of how to specify page and embedded image output file names in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying page and embedded image output file names in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To specify page and embedded image output file names in an Adobe FrameMaker source document

1. ***If you want to insert a Filename marker for a page output file,*** complete the following steps:
  - a. In your Adobe FrameMaker source document, locate the page for the topic to which you want to assign a specific filename. For more information about creating pages using page breaks, see “Specifying Page Breaks Settings”.
  - b. Insert your cursor at the beginning of the first paragraph on the page.
2. ***If you want to insert a Filename marker for an embedded image output file,*** complete the following steps:
  - a. In your Adobe FrameMaker source document, locate the embedded image for which you want to assign the output graphic file.
  - b. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
  - c. Click the **Text Frame** icon.
  - d. Drag the cursor across the image to draw a text frame over the image.
  - e. In the Create New Text Frame window, in the **Number** field, type **1**, and then click **Set**.
  - f. Click outside of the image, and then insert your cursor in the text frame.
3. In Adobe FrameMaker, on the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select **Filename** from the drop-down list.
5. ***If the Filename marker type is not on the list,*** check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
6. In the **Marker Text** field, type the file name you want to assign to the output file. Do not include the output file extension when you type the Filename marker text.
7. Click **New Marker**.
8. Save your source document.

9. Generate output for your project. For more information, see “Generating Output”.
10. In Output Explorer, verify ePublisher created an output file using the file name you specified. For more information, see “Viewing Output in Output Explorer”.

## Creating Context-Sensitive Help in FrameMaker

This section explains how you can use ePublisher to create links to context-sensitive help content in Adobe FrameMaker source documents.

### Context-Sensitive Help in FrameMaker

Context-sensitive help links provide content based on the context of what the user is doing. In many cases, this help content is based on the window that is open and active. For example, the Help button on a window in a software product can open a specific Help topic that provides important information about the window:

- What the window allows you to do
- Brief concepts needed to understand the window
- Guidance for how to use the window
- Descriptions about each field on the window, valid values, and related fields
- Links to related topics, such as concepts and tasks related to the window

The Help topic can also be embedded in the window itself, such as an HTML pane that displays the content of the Help topic. Providing this content when and where the user needs it, without requiring the user to search through the help, keeps the user productive and focused. This type of help also makes the product more intuitive by providing answers when and where needed.

There are several methods for creating context-sensitive Help. In addition, output formats use different mechanisms to support context-sensitive Help. You can reference a topic in the following ways:

#### File name

Use a Filename marker to assign a file name to a topic. Each topic can have no more than one Filename marker by default. However, you can create a custom mapping mechanism using file names. Then, you can open the specific topic with that file name. However, if your file naming changes, you need to change the link to the topic. This file naming approach delivers context-

sensitive help capabilities in output formats that do not provide a mapping mechanism.

### **Internal identifier (topic alias)**

Use a TopicAlias marker to define an internal identifier for each topic. The benefit of using an internal identifier is that it allows file names to change without impacting the links from the product. The writer inserts this marker in a topic and specifies a unique value for that topic. Then, the mapping mechanism of your output format determines how that internal identifier is supported. Some output formats, such as HTML Help, use a mapping file that defines these topic aliases.

To simplify the coding of your source documents, the Stationery designer can also configure your Stationery to define both the file name and the topic alias for each topic file.

Before you begin to insert Filename markers or TopicAlias markers into your source documents, consult with your Stationery designer. Confirm that your Stationery supports context-sensitive help links, and discuss with your Stationery designer the type of marker you should use to define context-sensitive help link in your source documents.

***If you generate Eclipse Help output,*** you also can choose the topic description you want to display for each context-sensitive link. When you use a TopicAlias marker to create context-sensitive links, Eclipse creates a `contexts.xml` file that lists all of the context IDs for the Eclipse Help system you created using TopicAlias markers. In the `contexts.xml` file, Eclipse also provides a description of the context-sensitive link. By default, the description Eclipse provides for the context-sensitive link is the text of the first paragraph of the topic. However, if you want to specify a different description for the context-sensitive link, you can do this by using the TopicDescription marker. For more information about using the TopicDescription marker, see “Specifying Context-Sensitive Help Links in FrameMaker”.

## **Planning for Context-Sensitive Help in FrameMaker**

Creating context-sensitive help requires you to collaborate with application developers. Because topic IDs and map numbers must be embedded in both the software application and in your source documents, you and the application developers must agree in advance on the values to use.

Before you create context-sensitive help topics, first confirm with your application developers that the application supports context-sensitive help. Then work with your application developers to decide how to choose the topic ID for each context-sensitive help topic:

### **You choose the topic IDs**

You can choose a set of topic IDs and embed them in your source documents using TopicAlias markers. When you generate output, ePublisher can generate a mapping file using those topic IDs and assign a unique number to each topic ID. You can provide the generated mapping file to your application developers, who can embed the topic IDs in the application code. You can then manually maintain this mapping file, or you can allow ePublisher to generate a new file each time you generate the help. Remember to give the updated help system and mapping file to your application developers each time.

### **Your developers choose the topic IDs**

Your application developers can choose a set of topic IDs and embed them in the application code. Then, you can get a copy of the mapping file from your application developers, specify this mapping file in your project settings, and embed the topic IDs in your source documents using TopicAlias markers. In this case, ePublisher does not generate the mapping file.

Before you begin to implement context-sensitive help, meet with your application developers to select one of these methods for assigning the topic IDs to use for context-sensitive help links. Once you choose a set of topic IDs, embed them in your source documents using TopicAlias markers and do not change them.

## **Specifying Context-Sensitive Help Links in FrameMaker**

You can use TopicAlias markers that contain topic IDs, or Filename markers that specify file names, to create context-sensitive help. If your output format supports the use of mapping files and topic IDs, typically you use TopicAlias markers to create context-sensitive help. If your output format does not support the use of mapping files and topic IDs, typically you use Filename markers to create context-sensitive help.

***If you are generating Eclipse Help***, you can also choose to specify a topic description for each context-sensitive help link you created using a TopicAlias marker by using a TopicDescription marker in conjunction with the TopicAlias marker. For more information about how TopicAlias markers and TopicDescription markers can work together when generating Eclipse Help, see “Context-Sensitive Help in FrameMaker”.

To specify a context-sensitive help link, your Stationery and template must have a TopicAlias or Filename marker type configured. If you are generating Eclipse Help and you want to be able to specify topic descriptions for your context-sensitive help links, your Stationery and template must also have a TopicDescription marker type configured. Consult with the Stationery designer to determine which marker type you should use to create context-sensitive help links and topic descriptions in your source documents.

The following procedure provides an example of how to create context-sensitive help links and topic descriptions in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating context-sensitive help links in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.



## To create a context-sensitive help link in an Adobe FrameMaker source document

1. Open the Adobe FrameMaker source document that contains the context-sensitive topic you want to link to when users click a help button or help icon from within an application.
2. Insert your cursor at the beginning of the topic or paragraph in which you want to link.
3. On the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select the marker type the Stationery designer configured your Stationery to support from the drop-down list. For example, select **TopicAlias** or **Filename**.
5. ***If the TopicAlias or Filename marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
6. In the **Marker Text** field, type the topic ID or file name you want to use for the context-sensitive help link. Specify topic IDs or file names that met the following guidelines:
  - Must be unique
  - Must begin with an alphabetical character
  - May contain alphanumeric characters
  - May not contain special characters or spaces, with the exception of underscores ( \_ )
7. Click **New Marker**.
8. ***If you are generating Eclipse Help and you want to specify topic descriptions for each context-sensitive help link you are creating***, complete the following steps:
  - a. Insert your cursor in the topic after the TopicAlias marker you inserted for the Eclipse context-sensitive help topic.
  - b. On the **Special** menu, click **Marker**.
  - c. In the **Marker Type** field, select **TopicDescription** marker type from the drop-down list.

- a. ***If the TopicDescription marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
  - b. In the **Marker Text** field, type the topic description you want to use.
  - c. Click **New Marker**.
- 9. Save your source Adobe FrameMaker source document.
- 10. Generate output for your project. For more information, see “Generating Output”.
- 11. In Output Explorer, complete the following steps:
  - a. Verify that ePublisher inserted the topic ID into the map file when it generated output.
  - b. ***If you generated Eclipse Help and specified topic descriptions for your context-sensitive help topics***, verify that the `contents.xml` file for your Eclipse Help system contains the topic descriptions you specified for context-sensitive help topics.
  - c. Test the generated output using the application and verify that the application links to the appropriate context-sensitive help topic. This testing ensures the context-sensitive help link you created displays correctly within the application.

## Creating Popup Windows in FrameMaker

A popup window is a window that is smaller than standard windows and typically does not contain some of the standard window features such as tool bars or status bars. Popup windows display when users hover over or click on a link. The popup window closes automatically as soon as the users click somewhere else.

A typical use of popup windows is to display glossary terms. For example, in printed documentation, terms and definitions are typically grouped in a separate glossary document. However, in online content, you can display glossary definitions in popup windows. With glossary popup windows, users can choose whether or not they want to view the definition of a term.

You create popup windows by creating a link between the word or phrase in a topic and the content you want to display in the popup window. After you create the

link, you then insert Popup markers or apply Popup paragraph styles to define the content you want to display in the popup window.

If the Stationery designer configured the Stationery to support popup windows using markers, you use the following Popup markers to create popup windows:

### **Popup**

Specifies the start of the content to include in a popup window. The content displays in a popup window when users hover over or click on the link. In some output formats users can also view the content in a standard help topic window in addition to viewing the content in a popup window. For example, if you insert a Popup marker in front of a glossary definition, the glossary definition displays in both a popup window and in a glossary topic that contains the definition.

### **PopupEnd**

Specifies the end of the content to display in the popup window.

### **PopupOnly**

Specifies that the popup content displays only through a popup window. For example, if you insert a PopupOnly marker in front of a glossary definition, the glossary definition displays only in a popup window.

If the Stationery designer configured the Stationery to support popup windows using paragraph formats, you use the following paragraph formats to create popup windows:

### **Popup and Popup Append paragraph behaviors**

Specifies that content displays both in popup windows and in standard help topics. You apply the Popup paragraph format to the first paragraph of content you want displayed in the popup window. If you have more than one paragraph of content you want to display, you apply the Popup Append format to the additional paragraphs.

For example, if you apply a glossary term and glossary definitions format for a glossary using the Popup and Popup Append format, the terms and definitions in your output display in both a popup window and in a glossary topic that contains the definitions.

### **Popup Only and Popup Only Append paragraph behaviors**

Specifies that content displays only in popup windows. You apply the Popup Only paragraph format to the first paragraph of content you want displayed in the popup window. If you have more than one paragraph of content you want to display, you apply the Popup Only Append format to the additional paragraphs.

For example, if you apply a glossary term and glossary definition format for a glossary using the Popup Only and Popup Only Append paragraph format, the terms and definitions in your output display in only popup windows. The content is not displayed in an additional glossary topic that contains the definitions.

## **Creating Popup Window Links in FrameMaker**

Your first step in creating a popup window is to create a link between a word or phrase in a topic and the popup content you want to display when users hover over or click the link. Use native Adobe FrameMaker functionality to create a link between the word or phrase in a topic and the content you want to display in a popup window. You can create a link in Adobe FrameMaker by using a cross-reference or by using hypertext markers.

Before you create popup window links, verify that your output format supports this feature.

The following procedure provides an example of how to create a popup window link in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating links in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create a link between a word or phrase and popup content in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the text you want to create a link to and display in the popup window.
2. ***If you want to create a link that includes the link target text***, create the link using a cross-reference by completing the following steps:
  - a. Select the text for which you want to create a link.
  - b. On the **Special** menu, click **Cross-Reference**.
  - c. In the **Document** field, select the document that contains the content to which you want to link.
  - d. In the **Paragraph Tags** field, select the paragraph tag used for the content to which you want to link.
  - e. In the **Paragraphs** field, select the paragraph to which you want to link.
  - f. In the **Format** field, select the appropriate format for the link. For example, if you are creating a link to a glossary term, select a glossary term cross-reference format.
  - g. Click **Replace**.
3. ***If you want to create a link that does not include the link target text***, create the link using a hypertext marker by completing the following steps:
  - a. Insert your cursor in front of the link target text.
  - b. On the **Special** menu, click **Marker**.
  - c. In the **Marker Type** field, click **Hypertext**.
  - d. In the **Marker Text** field, `newlink linkname` or `newlink filename:linkname`, where `linkname` is the name of the named destination for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document. To make maintenance easy, create short link names that use alphanumeric, lowercase characters.
  - e. Click **New Marker**.
  - f. Insert your cursor in front of the word or phrase for which you want to create a link.
  - a. On the **Special** menu, click **Marker**.

- b. In the **Marker Type** field, select **Hypertext** from the list.
- c. In the **Marker Text** field, type `gotolink` *linkname* or `gotolink` *filename:linkname*, where `linkname` is the name of the named destination you created for the link, and `filename` is the name of the file that contains the link, if the link is in a different Adobe FrameMaker source document.
- d. Click **New Marker**.
- e. Select the word or phrase for which you want to create a link. The selected area must contain the both text and the hypertext marker you created.
- f. Apply a link character format to the word or phrase. If you do not know which character format to use for links, consult the Stationery designer.

4. Save your Adobe FrameMaker source document.

After you create a link between a word or phrase in a topic and the popup content you want to display in the popup window, define the content you want to display in the popup window using one of the following methods:

- Create popup windows using Popup markers. For more information, see “Using Markers to Create Popup Windows in FrameMaker”.
- Create popup windows using Popup paragraph formats. For more information, see “Using Paragraph Formats to Create Popup Windows in FrameMaker”.

## Using Markers to Create Popup Windows in FrameMaker

You can insert Popup markers into your Adobe FrameMaker source documents to create popup windows. To use Popup markers to create popup windows, your Stationery and FrameMaker template must have the following items configured:

- Popup marker type
- PopupEnd marker type
- PopupOnly marker type

Your output format must also support this feature.

The following procedure provides an example of how to insert Popup markers in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for inserting Popup markers in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

**Note:** Popup content is created from whole paragraphs. You cannot include a subset of a paragraph in a popup.

## To use popup markers to create popup windows in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, create a link between a word or phrase in the topic and the content you want to display in the popup window. For more information, see “Creating Popup Window Links in FrameMaker”.
2. Insert your cursor in front of the text you want to display in the popup window.
3. On the **Special** menu, click **Marker**.
4. ***If you want the popup content to display in both a popup window and in a standard help topic,*** in the **Marker Type** field select **Popup** from the drop-down list.
5. ***If you want the popup content to display only in a popup window,*** in the **Marker Type** field select **PopupOnly** from the drop-down list.
6. ***If the Popup or PopupOnly marker type is not on the list,*** check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
7. In the **Marker Text** field, do not enter any text. You do not need to enter any text in this field when you insert a Popup or PopupOnly marker.
8. Click **New Marker**.
9. Specify where you want the popup content to end by completing the following steps:
  - a. Insert your cursor at the end of the content you want to display in the popup window.
  - b. On the **Special** menu, click **Marker**.
  - c. In the **Marker Type** field, select **PopupEnd** from the drop-down list.
  - d. ***If the PopupEnd marker type is not on the list,*** check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
  - e. In the **Marker Text** field, do not enter any text. You do not need to enter any text in this field when you insert a PopupEnd marker.



- f.** Click **New Marker**.
- 10.** Save your Adobe FrameMaker source document.
- 11.** Generate output for your project. For more information, see “Generating Output”.
- 12.** In Output Explorer, go to the page where you created the popup window and verify that ePublisher created the popup window and that the popup window displays the content you specified. For more information, see “Viewing Output in Output Explorer”.

# Using Paragraph Formats to Create Popup Windows in FrameMaker

You can use Popup paragraph formats in your Adobe FrameMaker source documents to create popup windows. To use Popup paragraph formats to create popup windows, your Stationery and FrameMaker template must have the following items configured:

- **Popup and Popup Append** paragraph formats if you want your content to display both in popup windows and in standard help topics.
- **Popup Only and Popup Only Append** paragraph formats if you want your content to display only in popup windows.

Your output format must also support this feature.

The following procedure provides an example of how to use Popup paragraph formats to create popup windows in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for using Popup paragraph formats to create popup windows in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## **To create popup windows using Popup paragraph formats in an Adobe FrameMaker source document**

1. In your Adobe FrameMaker source document, create a link between a word or phrase in the topic and the content you want to display in the popup window. For more information, see “Creating Popup Window Links in FrameMaker”.
2. Apply the appropriate Popup paragraph format to the popup content you want to display in the popup window.
3. Save your Adobe FrameMaker source document.
4. Generate output for your project. For more information, see “Generating Output”.
5. In Output Explorer, go to the page where you created the popup window and verify that ePublisher created the popup window and that the popup window displays the content you specified. For more information, see “Viewing Output in Output Explorer”.

## **Creating Expand/Collapse Sections (Drop-Down Hotspots) in FrameMaker**

You can create sections of content that expand and collapse when you click a link or hot spot. This structure allows you to create items, such as tasks with numbered procedures, bulleted lists, or definitions, that are easy to scan. Users can then expand individual items to display additional information.

Hot spots for expand/collapse sections initially display in one of the following states:

- The content is initially collapsed and will expand beneath the hotspot when the user clicks the hotspot. Clicking the hotspot a second time causes the expanded content to return to its original collapsed state.
- The content is initially expanded and will collapse or disappear from beneath the hotspot when the user clicks the hotspot.

You use an Expand/Collapse paragraph or table format to start expand/collapse sections and a DropDownEnd marker to specify where the content in the expand/collapse section ends. The Stationery defines whether the sections should initially be expanded (shown) or collapsed (hidden) and the image used to show the state of the section.

To create expand/collapse sections, your Stationery and FrameMaker template must have the following items configured:

- An Expand/Collapse paragraph or table format

- A DropDownEnd marker

Your output format must also support this feature.

The following procedure provides an example of how to create expand/collapse sections in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating expand/collapse sections in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create an expand/collapse section in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, identify a topic that contains text for which you want to create an expand/collapse section.
2. Apply an Expand/Collapse paragraph format to the text you want users to click to expand or collapse content.

For example, in the following sample procedure, you could apply the Expand/Collapse paragraph format to the *To open a project* text.

*To open a project*

- a. On the **File** menu, click **Open**.
  - b. Browse to the location of the project on your local computer.
  - c. Select the project, and then click **Open**.
3. Insert your cursor at the end of the content you want to display in the expand/collapse section.

For example, in the following sample procedure, you would insert your cursor after the period in the last sentence of the procedure, *Select the project, and then click Open*.

*To open a project*

- a. On the **File** menu, click **Open**.
  - b. Browse to the location of the project on your local computer.
  - c. Select the project, and then click **Open**.
4. On the **Special** menu, click **Marker**.
5. In the **Marker Type** field, select **DropDownEnd** from the drop-down list.
6. ***If the DropDownEnd marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
7. In the **Marker Text** field, do not enter any text. You do not need to enter any text in this field when you insert a DropDownEnd marker.
8. Click **New Marker**.

9. Save your Adobe FrameMaker source document.
10. Generate output for your project. For more information, see “Generating Output”.
11. In Output Explorer, go to the page where you created the expand/collapse section and verify that ePublisher created the expand/collapse section and that the expand/collapse section displays the content you specified. For more information, see “Viewing Output in Output Explorer”.

## Creating Related Topics in FrameMaker

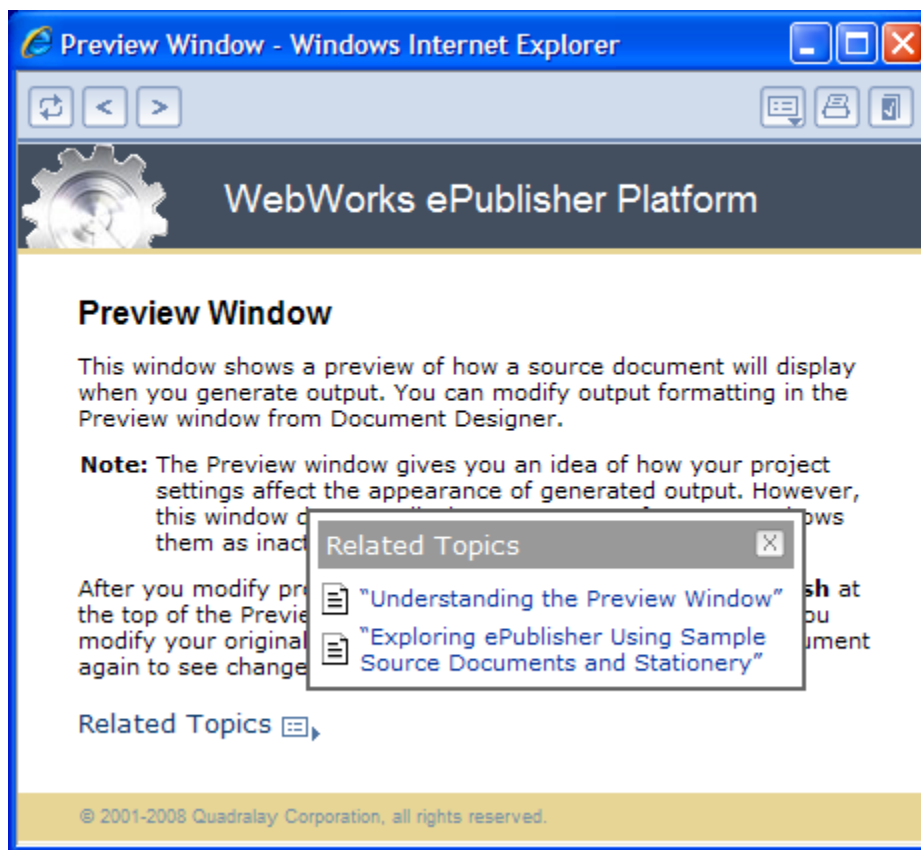
Related topics provide a list of other topics that may be of interest to the user viewing the current topic. For example, you could have a section called Creating Web Pages in your help. You may also have many other topics, such as HTML Tags and Cascading Style Sheets, that related to creating Web pages. Identifying these related topics for users can help them find the information they need and identify additional topics to consider. However, providing these types of links as cross-references within the content itself may not be the most efficient way to present the information. By utilizing related topics links, you combine the capabilities of cross-references with the efficiency of a related topics button.

Related topics and See Also links provide similar capabilities, but there are several important differences:

- Related topics can link to headings in a Help system that do not start a new page.
- Related topics links are static and defined in the source documents as links. You must have all the source documents to create the link and generate the output.
- If a related topics list contains a broken link in the source document, that link is broken in the generated output. In a See Also link list, the broken link is not included in the output.

The Stationery designer can configure related topics to display in the following ways:

- Included as a list in the topic itself.
- Displayed in a popup window when the user clicks a button, as show in the following figure.



**Note:** If a related topic link is broken in the source document, in most cases that link is broken in the generated output. WebWorks Help and WebWorks Reverb provide an additional feature by removing broken links from related topics lists that are displayed in a popup window when a user clicks the Related Topics button.

To create related topics links, your Stationery and FrameMaker template must have a Related Topics paragraph style configured. Your output format must also support this feature.

The following procedure provides an example of how to create related topics links in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating related topics links in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create a related topics list in an Adobe FrameMaker source document

1. Identify the topic in which you would like to insert a related topics list.
2. Identify the different topics you want to link to from this topic.

**Note:** Generally, you should only create one related topics list for each section of your source document that corresponds to a help topic. For example, if the Stationery designer specified in your Stationery that there will be a page break at each Heading 1 section, then you should only create one related topics list for each Heading 1 section within your source document.

3. Create a cross-reference to each topic you want to include in the related topics list by completing the following steps:
  - a. Insert your cursor in the location in your Adobe FrameMaker source document where you want to insert the link to the related topic.
  - b. On the **Special** menu, click **Cross-Reference**.
  - c. In the **Document** field, select the source document that contains the topic to which you want to link.
  - d. In the **Source Type** field, click **Paragraphs**.
  - e. In the **Paragraph Tags** field click the paragraph tag used by the topic to which you want to link.
  - f. In the **Paragraphs** field, select the topic to which you want to link.
  - g. In the **Format** field, select the format you want to use for the cross-reference.
  - h. Click **Insert**.
4. Apply the Related Topic paragraph format to the cross-references in your related topics list.
5. ***If you want to display the list of related topics in only your generated output***, apply an OnlineOnly condition to the list of related topics. For more information about applying conditions, refer to "Applying Conditions in FrameMaker".
6. Save your Adobe FrameMaker source document.
7. Generate output for your project. For more information, see "Generating Output".

8. In Output Explorer, go to the page where you created the related topics list and verify that ePublisher created the related topics and that the related topics list displays the topics you specified. For more information, see “Viewing Output in Output Explorer”.

## Creating See Also Links in FrameMaker

See Also links, also known as ALinks, or associative links, are links that may be of interest to the user viewing the current topic. These links use internal identifiers to specify the links and the link list is built dynamically based on the topics available when the user clicks to display the links. See Also links are important to use with larger help sets and merged help sets.

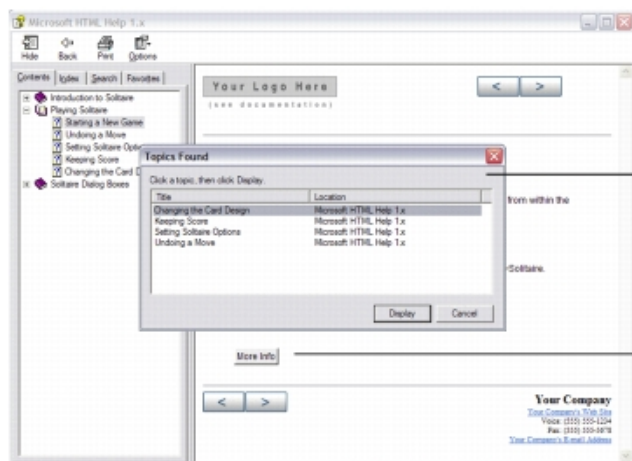
Related topics and See Also links provide similar capabilities, but there are several important differences:

- See Also links must link to formats that start a new topic, such as a heading.
- See Also links are dynamic and the lists of links are built at display time instead of during help generation.
- Since see Also link lists are dynamically built, they do not include links to topics that are not available when the user displays the links. If a related topics list contains a broken link in the source document, that link is broken in the generated output for most output formats.

See Also links are useful if you plan to merge help systems. For example, if you have a multiple help systems that you merge into one main help system at run time and if your topics in the merged help systems contain See Also keywords that are also used in the main help system, links to those topics are included in the See Also lists in the main project.

You can create See Also links as buttons or as inline text links in Microsoft HTML Help and WebWorks Help. The following example shows how the two different types of See Also links display in a Microsoft HTML Help system.





Dialog box  
listing See  
Also links

See Also  
button labeled  
**More Info**

Create See Also links by applying the See Also paragraph format or character format to text in your Adobe FrameMaker source documents and inserting markers into your Adobe FrameMaker source documents. To create See Also links, your Stationery and template must have the following items configured:

- See Also paragraph format if you want to create See Also links with buttons
- See Also paragraph format if you want to create see Also links as inline text links
- SeeAlsoKeyword marker type
- SeeAlsoLink marker type
- SeeAlsoLinkDisplay marker type if you generate Microsoft HTML Help and you want to display the target topics in a popup menu
- SeeAlsoLinkWindowType marker type if you generate Microsoft HTML Help and you want to display the target topics in a custom window

The following procedure provides an example of how to create See Also links in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating See Also links in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create a See Also link in an Adobe FrameMaker source document

1. Identify each topic to which you want to link from a See Also link, and then complete the following steps for each topic:
  - a. Insert your cursor into the topic to which you want to link.
  - b. On the **Special** menu, click **Marker**.
  - c. In the **Marker Type** field, select **SeeAlsoKeyword** from the drop-down list.
2. ***If the SeeAlsoKeyword marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
3. In the **Marker Text** field, type a text string that is a unique identifier for the topic.

For example, if you have a unique topic called About WebWorks Help, type `AboutWebWorks` help in the **Marker Text** field.

4. Click **New Marker**.
5. Identify the topic where you want to insert a list of See Also links.
6. Enter the text you want to display for the See Also button or for the See Also inline text link on a separate line in the source document where you want the See Also button or inline text link to display.

For example, if you want to create a button with the text See Also on the button, type `See Also`. If you want to create inline text with the text Additional Information for the link, type `Additional Information`.

7. ***If you want to create a See Also button for your See Also links***, apply the See Also paragraph format to the text you want to display in the See Also button.
8. ***If you want to create a See Also inline text link for your See Also links***, apply the See Also character format to the text you want to display for the See Also inline text link.
9. Apply an OnlineOnly condition to the See Also text. Applying an OnlineOnly condition to the See Also button or See Also inline text displays the See Also link in your generated output, but does not display the See Also button or link in your printed content.

**10.** Insert your cursor inside the text you specified for the See Also button or See Also inline text link.

**11.** On the **Special** menu, click **Marker**.

**12.** In the **Marker Type** field, select **SeeAlsoLink** from the drop-down list.

**13.** *If the **SeeAlsoLink** marker type is not on the list*, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".

**14.** In the **Marker Text** field, type the text string that is a unique identifier for the topic to which you want to link. This text string is the text string you typed when you created the SeeAlsoKeyword marker for the topic.

For example, if you have a unique topic called About WebWorks Help, type `AboutWebWorks` help in the **Marker Text** field.

**15.** Click **New Marker**.

**16.** Continue to insert SeeAlsoLink markers for each topic you want display when users click the See Also button or inline text link.

**17.** *If you generate Microsoft HTML Help output and you want to display the target topics in a popup menu*, complete the following steps:

**a.** Insert your cursor inside the text you specified for the See Also button or inline text link.

**b.** In the **Marker Type** field, select **SeeAlsoLinkDisplayType** from the drop-down list.

**Note:** This marker type is supported only in Microsoft HTML Help.

**c.** *If the **SeeAlsoLinkDisplayType** marker type is not on the list*, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".

**d.** In the **Marker Text** field, type `menu`. By default, Microsoft HTML Help displays See Also links in the Topics Found window. To display See Also links in a popup menu, specify menu for the marker value.

**e.** Click **New Marker**.

**18. If you generate Microsoft HTML Help output and you want to display the target topics in a custom window**, complete the following steps:

- a. Insert your cursor inside the text you specified for the See Also button or inline text link.
- b. In the **Marker Type** field select **SeeAlsoLinkWindowType** from the drop-down list.

**Note:** This marker type is supported only in Microsoft HTML Help.

- c. **If the SeeAlsoWindowType marker type is not on the list**, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker"
- d. In the **Marker Text** field, type `menu`. By default, Microsoft HTML Help displays See Also links in the Topics Found window. To display See Also links in a popup menu, specify menu for the marker value.
- e. Click **New Marker**.

**19.** Save your Adobe FrameMaker source document.

**20.** Generate output for your project. For more information, see "Generating Output".

**21.** In Output Explorer, go to the page where you created the See Also links and verify that ePublisher created the See Also button or See Also inline text and that the See Also button or inline text displays the links you specified. For more information, see "Viewing Output in Output Explorer".

## Creating Meta Tag Keywords in FrameMaker

Meta tags are lines of code placed between the `<head>` and `</head>` tags in HTML pages. Meta tags give web search engines information about the content of the web page and how search engines should treat the web page. Users viewing web pages do not see the meta tags, but meta tags can be used to influence the way web pages on a web site appear in web search engine results. Users also see the text you specify for meta tags right following the title of your page when your page comes up in search results.

In help systems, search ranking works like ranking in an Internet search engine. If you generate help system output, you can use meta tag keywords to specify terms for pages for help topics where you want to improve searchability. For example,

assume that in your help system you have a topic called See Also links. However, you know that See Also links are also sometimes referred to as ALinks, and you think that some users of your help system may search for information about See Also links by typing `ALinks` into the **Search** field for your help system. In this example, you can insert ALinks as a meta tag keyword for each page that discusses See Also links, so users who search your system for information about ALinks can find the information they are looking for in your See Also link topics.

To assign meta tag keywords, your Stationery and template must have the Keywords marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to create meta tag keywords in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for creating meta tag keywords in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To create meta tag keywords for a page in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, find the first paragraph in the page for the page for which you want to create a meta tag keyword.
2. Insert your cursor into the paragraph.
3. On the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select **Keywords** from the drop-down list.
5. **If the Keywords marker type is not on the drop-down list**, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
6. In the **Marker Text** field, type the comma-delimited list of keywords that you want search engines to use.

For example, type `keyword1, keyword2, keyword3`, where *keyword* is the keyword you want search engines to use.

7. Click **New Marker**.
8. Save your Adobe FrameMaker source document.
9. Generate output for your project. For more information, see "Generating Output".
10. In Output Explorer, verify that ePublisher inserted your meta tag keywords correctly by completing the following steps:
  - a. On the **View** menu, click **Output Explorer**.
  - b. In the *TargetName\ProjectName* folder, open the page to which you assigned meta tag keywords in Notepad, where *TargetName* is the name of your target and *ProjectName* is the name of your project.
  - c. Verify that the text you specified for your meta tag displays in the `meta name` attribute between in the `<head>` and `</head>` tags section of your web page. For example, if you typed `keyword1, keyword2, keyword3`, for your meta tag keywords, your meta tags in for the page should be similar to the following entry:

```
<meta name="keywords" content="keyword1, keyword2, keyword3" />
```

# Assigning Custom Page Styles in FrameMaker

By default, each page generated by ePublisher is associated with the default page style defined in the Stationery used by your ePublisher project. This means that typically you do not need to specify a page style for pages when you generate output. However, if you want to change the page style of one page or a smaller set of pages, you can specify the page style you want to use for a page in your Adobe FrameMaker source document using the PageStyle marker.

For example, you may want to use one page style in your help system for all concept and procedure topic pages, and another page style for all context-sensitive window description topic pages in your help system. In this example, you can use the default page style for all of your concept and procedure topic pages, and then you can use a second custom page style defined in your Stationery for all context-sensitive window description topic pages in your help system.

To assign custom page styles, your Stationery and template must have the following items configured:

- Custom page styles defined for your Stationery by the Stationery designer
- PageStyle marker type

Your output format must also support this feature.

The following procedure provides an example of specifying custom page styles for pages in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying custom page styles for pages in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To specify a custom page style for a page in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the page for the topic to which you want to assign a page style.
2. Insert your cursor in the location on the page where you want to insert the PageStyle marker.
3. On the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select **PageStyle** from the drop-down list.
5. ***If the PageStyle marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
6. In the **Marker Text** field, type the name of the custom page style the Stationery designer configured for your Stationery.

For example, if the Stationery designer configured an page style called BluePage in your Stationery, type `BluePage`.

7. Click **New Marker**.
8. Save your Adobe FrameMaker source document.
9. Generate output for your project. For more information, see "Generating Output".
10. In Output Explorer, verify ePublisher created the page using the page style you specified by clicking on the page and verifying ePublisher applied the page style you specified in the generated output. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Opening Topics in Custom Windows in FrameMaker

You can open topics in custom windows in Microsoft HTML Help and Oracle Help. By default, Microsoft HTML Help displays content in the standard Microsoft HTML Help tri-pane window. The Stationery designer can modify the size, position, and other characteristics of the tri-pane window in your Microsoft HTML Help project. The Stationery designer can also define custom windows for you to use in a Microsoft HTML Help project. If the Stationery designer defines custom windows in a Microsoft



HTML Help project, you can specify which topics you want to display in the custom window using the WindowType marker.

By default, Oracle Help displays content in the standard Oracle Help viewer. The Stationery designer can modify the size, position, and other characteristics of Oracle Help windows. The Stationery designer can also define custom windows for you to use in an Oracle Help project. If the Stationery designer defines custom windows in an Oracle Help project, you can specify which topics you want to display in the custom window using the WindowType marker.

For example, if you want your context-sensitive help topics to display in a different type of window than other content, after you create a context-sensitive help topic you can use the WindowType marker to specify that you want the context-sensitive help topics to display in a custom window. After you assign a custom window to a topic using the WindowType marker, the help system displays the topic in your generated output in the custom window whenever users access the topic from the table of contents, index, a standard hyperlink, a related topics list, or a See Also link.

To open topics in custom windows, your Stationery and template must have the following items configured:

- Custom window styles defined for your Stationery by the Stationery designer
- PageStyle marker type

Your output format must also support this feature.

The following procedure provides an example of how to specify topics open in custom Microsoft HTML Help or Oracle Help windows in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying topics open in custom Microsoft HTML Help or Oracle Help windows in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

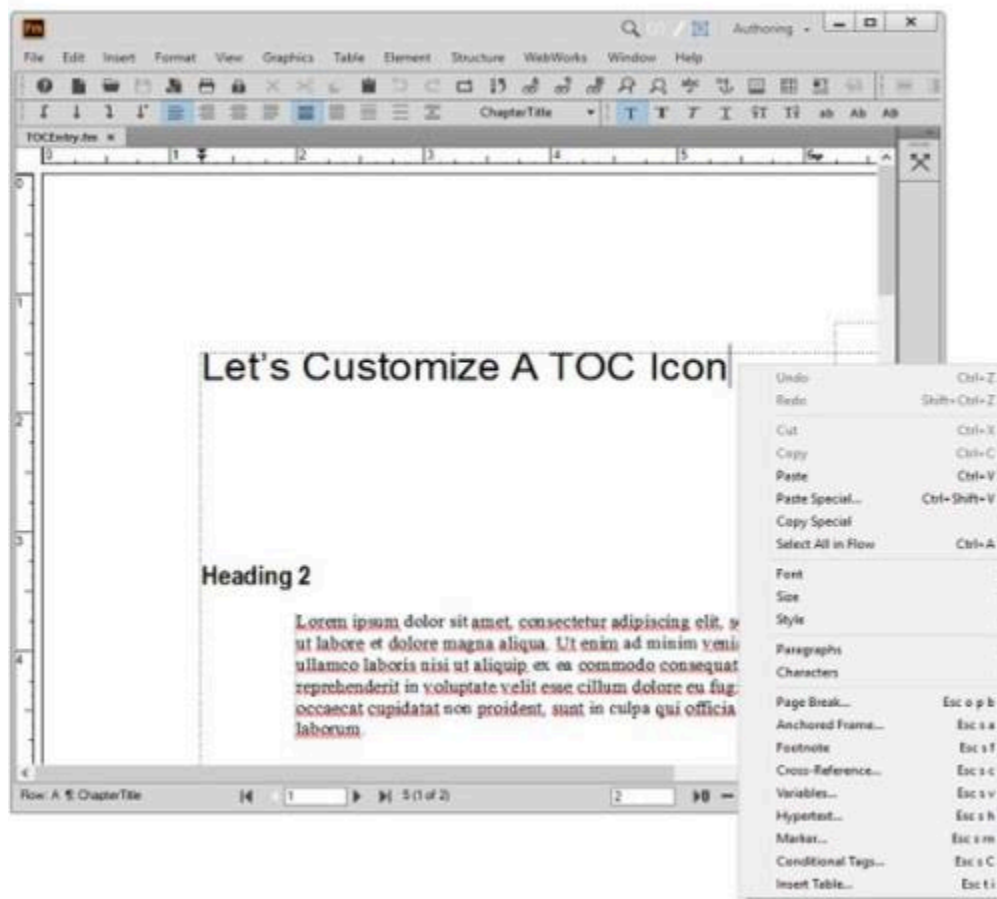
## To specify topics open in a custom window in an Adobe FrameMaker source document

1. Obtain the names of custom windows configured in the Stationery you use for your ePublisher project from the Stationery designer.
2. In your Adobe FrameMaker source document, locate the topic that you want to open in a custom window.
3. Insert your cursor into the topic.
4. On the **Special** menu, click **Marker**.
5. In the **Marker Type** field, select **WindowType** from the drop-down list.
6. *If the WindowType marker type is not on the list*, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
7. In the **Marker Text** field, type the name of the custom window configured by the Stationery designer that you want to specify for the topic.
8. Click **New Marker**.
9. Save your Adobe FrameMaker source document.
10. Generate output for your project. For more information, see "Generating Output".
11. In Output Explorer, verify the topic displays in the custom window you specified for the topic. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

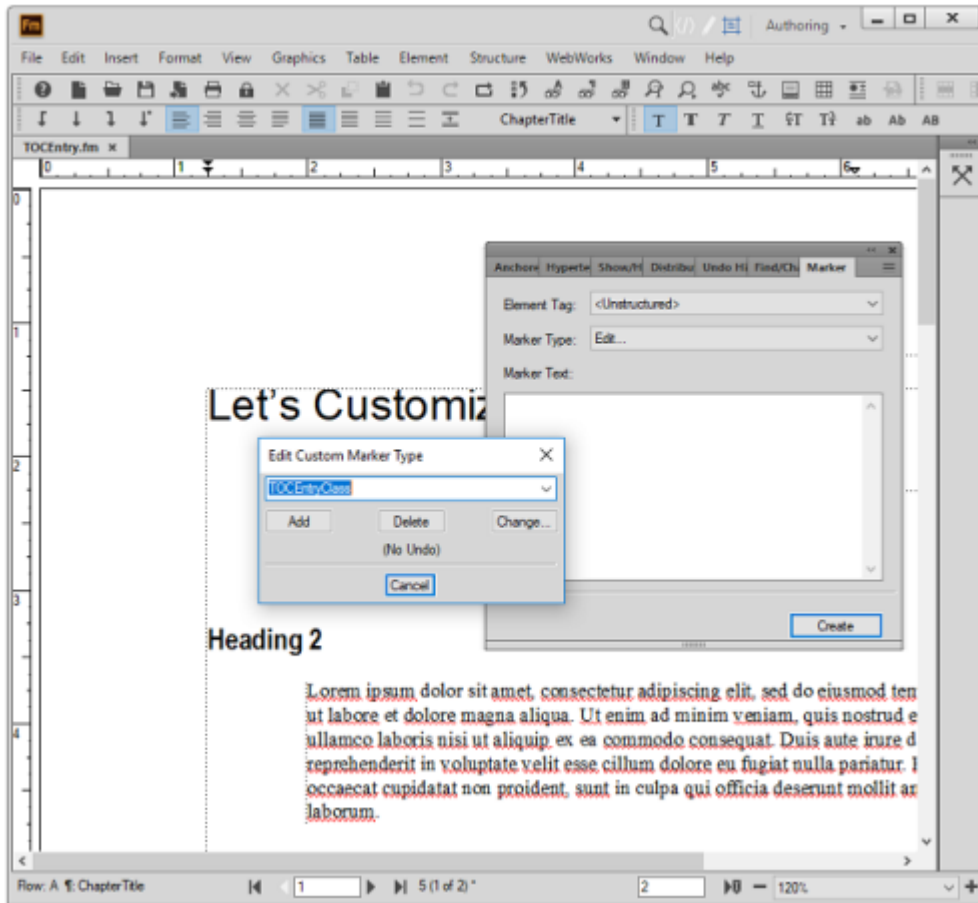
## Customizing TOC Entry in FrameMaker

Use these steps to customize a TOC entry in your **Reverb 2.0** output. Your FrameMaker file must have a nested heading structure for TOC Icons to appear.

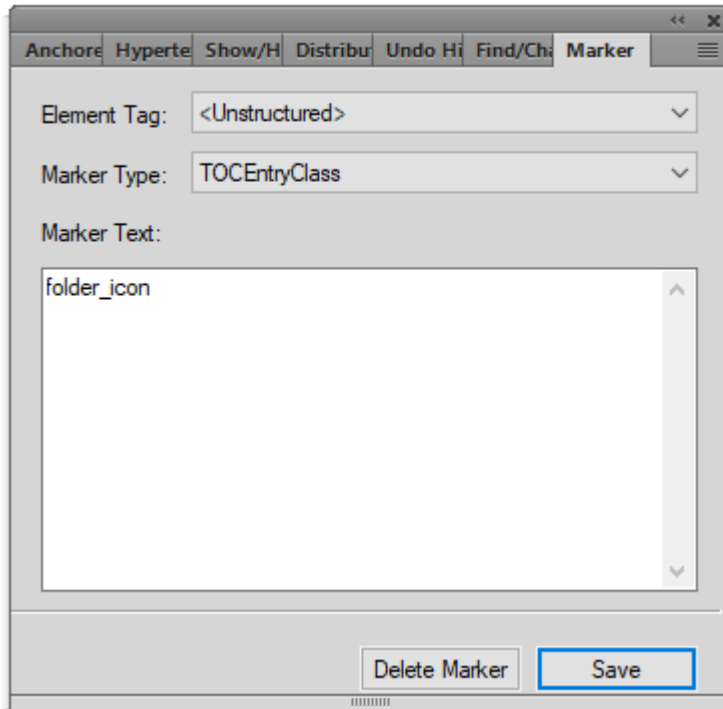
1. Right-click at the end the heading. Select Marker.



2. Select Edit from the Marker Type dropdown.
3. Type in TOCEntryClass. Click Add, then click Done.



4. In the Marker Text window, type in the name of your custom class. In the example, folder\_icon is the class name.



5. Save the FrameMaker document.
6. Scan the document in **ePublisher Designer**.
7. Open the **Style Designer**.
8. Open **Marker Styles**.
9. Locate the **Marker Type Option** from the **Options** tab and set its value to `TOC Entry Class`.
10. In this example, the assigned class for the Menu TOC entry will be the value of the marker: `folder_icon`.
11. Add the following to a target override of `_icons.scss`. Notice how the CSS class is the name of the value given in the Marker Text Window. In this example we change the icon color and the icon of the TOC entry. You are able to make other customizations such as adding a border, or changing the background color.

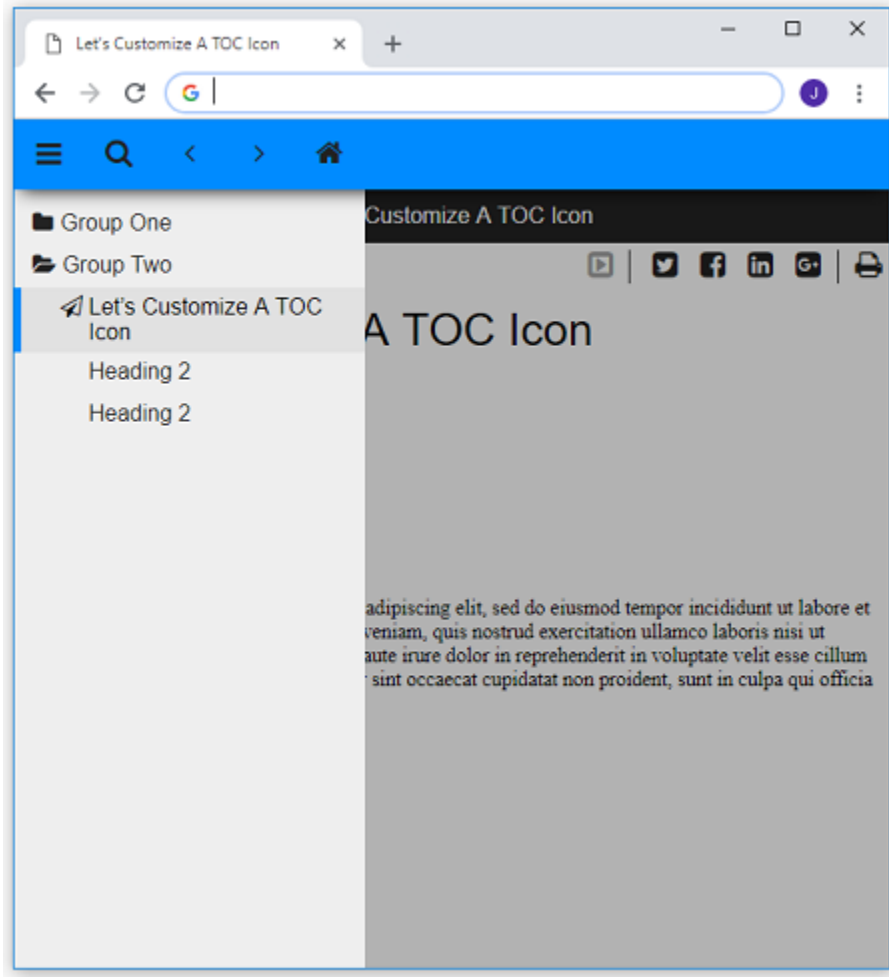
```

.folder_icon {
  > div > span > i {
    color: black;

    &:before {
      content: $folder_icon;
    }
  }
}

```

**12.** Save your project and generate the output.



# Customizing Table of Contents Icons in FrameMaker

By default, the **Contents** tab in a Microsoft HTML Help, Oracle Help, and WebWorks Help uses book and page icons to identify entries. By default, the **Contents** tab in Sun JavaHelp uses folder and page icons to identify entries. You can also customize the table of contents icons.

For example, if you want to make new topics stand out by using a unique icon specific to new books, pages, or folders, you can insert a marker into a topic and specify the icon you want to display for the book, page, or folder in your help system table of contents.

To customize a table of contents icon, your Stationery and template must have the following items configured:

- TOCIconHTMLHelp for Microsoft HTML Help
- TOCIconOracleHelp for Oracle Help
- TOCIconJavaHelp for Sun JavaHelp
- TOCIconWWHelp for WebWorks Help

You can customize the appearance of table of contents icons for topics in Microsoft HTML Help, Sun JavaHelp, Oracle Help, and WebWorks help.

The following procedure provides an example of how to customize table of contents icons for topics in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for customizing table of contents icons for topics in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To specify a custom table of contents icon in an Adobe FrameMaker source document

1. ***If you want to specify a custom table of contents icon for Microsoft HTML Help***, identify the number of the image you want to use for the table of contents image for the topic in the `.hhp` file for your Microsoft HTML Help project by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. Open the `ProjectName` folder, where *ProjectName* is the name of your project.
  - c. Open the `ProjectName.hhp` file where *ProjectName* is the name of your project.
  - d. On the **Contents** tab, select a table of contents entry, and then click the **Pencil** icon.
  - e. On the **Advanced** tab, in the **Image index** field, use the up and down arrows to identify the table of contents image you want to use for the topic.
  - f. Note the number of the image you want to use for the table of contents image for the topic.

For example, if you want to use a question mark icon with a red star for the table of contents icon for new topics, note that the number for this icon is 10.

- g. Close HTML Help Workshop.
2. ***If you want to specify a custom table of contents icon for Oracle Help or Sun JavaHelp***, create the graphic file for the custom table of contents icon in `.gif` format. The default graphics used as Sun JavaHelp or Oracle Help table of contents icons are 17 x 17 pixels. The custom graphics you create for Sun JavaHelp or Oracle Help table of contents icons should also be 17 x 17 pixels. You can assign any name to the graphic files.
3. ***If you want to specify a custom table of content icon for WebWorks help***, create graphics files containing the collapsed and expanded versions of the icons you want to use, then save the graphic files in `.gif` format. The default graphics used as WebWorks Help table of contents icons are 17 x 17 pixels. The custom graphics you create for WebWorks Help table of contents icons should also be 17 x 17 pixels. You can assign any name to the graphic files.
4. Copy the graphic files you want to use as icons in the table of contents into the following folder:



**Note:** If the folder does not exist, first create the folder using the specified folder structure and then copy the graphic files you want to use as icons into the folder. You do not need to perform this step when specifying custom table of contents icons for Microsoft HTML Help.

- **If you are generating Oracle Help**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Oracle Help\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating Sun JavaHelp 1.1.3**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Sun Java Help 1.1.3\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating Sun JavaHelp 2.0**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Sun Java Help 2.0\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating WebWorks Help**, in your `ProjectName\Files` folder, where *ProjectName* is the name of your project, create a `wwhelp\images` subfolder and copy the graphic files you want to use into this folder. Your project file structure should be similar to the following structure:

`ProjectName\Files\wwhelp\images`, where *ProjectName* is the name of your project.

5. In your Adobe FrameMaker source document, locate the topic where you want to use the custom table of contents icon.
6. Insert your cursor into the heading for the topic.
7. On the **Special** menu, click **Marker**.
8. In the **Marker Type** field, select the appropriate TOCIcon marker type from the drop-down list.
9. **If the appropriate TOCIcon marker type is not on the list**, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
10. In the **Marker Text** field, type the following text:

- **If you are generating Microsoft HTML Help**, type the number of the icon that you want to use for the table of contents image.

For example, if you want to use a question mark icon with a red star for the table of contents icon for new topics, type `10`.

- **If you are generating Oracle Help or Sun JavaHelp**, type the following text:

```
images/TOCIcon.gif
```

where `TOCIcon.gif` is the name of the table of contents icon you want to display for the topic.

- **If you are generating WebWorks Help**, type the following text:

```
c="collapsed.gif" e="expanded.gif"
```

where `collapsed.gif` is the name of the icon you want to use when the table of contents entry is collapsed, and `expanded.gif` is the name of the icon you want to use when the table of contents entry is expanded. If the table of contents entry is for a page instead of a book, the entry will never be expanded, so you can omit the `e="expanded.gif"` portion of the entry for pages.

For example, you might create a special icon to highlight books that are new for a particular release of your WebWorks Help system. If you named these icons `newbookopen.gif` and `newbookclosed.gif`, you would type the following text into the **Value** field:

```
c="newbookclosed.gif" e="newbookopen.gif"
```

11. Click **New Marker**.
12. Save your Adobe FrameMaker source document.
13. Generate output for your project. For more information, see "Generating Output".
14. In Output Explorer, verify ePublisher created the table of contents using the table of contents icon you specified for the topic. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Specifying Context Plug-ins in FrameMaker

You can specify Eclipse Help context plug-ins by using Context Plugin markers in your source documents. ePublisher places the context plug-ins you specify in your source documents in the `plugin.xml` file generated for each source document group you have in Document Manager. You can then have developers use the context plug-ins defined in `plugin.xml` files to call your Eclipse Help system as appropriate from Eclipse plug-ins.

For example, assume you have the following three top-level groups in Document Manager for your Eclipse Help system target:

- Component A group - contains the source documents for ComponentA Feature1 and ComponentA Feature2
- Component B group - contains the source documents for ComponentB Feature1 and ComponentB Feature 2
- Component C group - contains the source documents for ComponentC Feature1 and ComponentC Feature 2

You insert the following Context Plugin markers into the source documents for each group:

- ComponentAFeature1 and ComponentAFeature2 Context Plugin markers in source documents contained in the ComponentA group
- ComponentBFeature1 and ComponentBFeature2 Context Plugin markers in source documents contained in the ComponentB group
- ComponentCFeature1 and ComponentCFeature2 Context Plugin markers in source documents contained in the ComponentC group

When you generate your Eclipse Help system, ePublisher creates the following folder structure in the *ProjectName*\Output\TargetName folder, where *ProjectName* is the name of your ePublisher project, and *TargetName* is the name of your target:

- `ComponentA` folder, which contains a `plugin.xml` file with the following entries:

```
plugin="ComponentAFeature1ContextPlugin"
```

```
plugin="ComponentAFeature2ContextPlugin"
```

- `ComponentB` folder, which contains a `plugin.xml` file with the following entries:

```
plugin="ComponentBFeature1ContextPlugin"
```

```
plugin="ComponentBFeature2ContextPlugin"
```

- `ComponentC` folder, which contains a `plugin.xml` file with the following entries:

```
plugin="ComponentCFeature1ContextPlugin"
```

```
plugin="ComponentCFeature2ContextPlugin"
```

You can then provide the context plug-in IDs in your `plugin.xml` files to the appropriate Eclipse developers to use. The Eclipse developers use the context plug-ins defined in `plugin.xml` files to call your Eclipse Help system as appropriate from Eclipse plug-ins.

## To specify a context plug-in in an Adobe FrameMaker source document

1. Identify a topic in a source document where you want to insert the context plug-in.
2. On the **Special** menu, click **Marker**.
3. In the **Marker Type** field, select **Context Plugin** from the drop-down list.
4. **If the Context Plugin marker type is not on the list**, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
5. In the **Marker Text** field, type the appropriate ID for the context plug-in.

**Note:** If you are responsible for defining the ID, ensure you supply the context plug-in ID to your developers to use as appropriate for their Eclipse plug-ins. If your developers define the ID, use the context plug-in ID you obtained from your developers.

6. Click **New Marker**.
7. Save your Adobe FrameMaker source document.
8. Generate output for your project. For more information, see "Generating Output".
9. In Output Explorer, verify ePublisher generated a `plugin.xml` file that contains the context plug-in IDs you specified by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. Open the *ProjectName* folder, where *ProjectName* is the name of your project.
  - c. Open the group folder for a group that contains the source documents where you specified your context plug-in ID.
  - d. Open the `plugin.xml` file in Notepad and verify that the context plug-in IDs you specified in your source documents are listed in the `plugin.xml` file. Your context plug-in IDs should be listed in the Contexts area of the file. Following is an example of the how the context plug-in IDs you specified in your source documents should be displayed in the `plugin.xml` file:

```
<!-- Contexts -->
```

```
<!-- -->

<extension point="org.eclipse.help.contexts">

<contexts file="contexts.xml"
plugin="ComponentAFeature1ContextPlugin" />

</extension>

<extension point="org.eclipse.help.contexts">

<contexts file="contexts.xml"
plugin="ComponentAFeature2ContextPlugin" />

</extension>
```

## Creating Accessible Online Content in FrameMaker

**Accessible content** is content that can be easily accessed by users with certain disabilities. This section explains how you can prepare your Adobe FrameMaker source documents to ensure your content is accessible to users using assistive technologies.

### Accessible Content in FrameMaker

Images and tables are helpful ways to convey information to end users. However, users with disabilities often cannot access the important information provided by images and table layouts in online content. You should document images and other non-text items such as table layouts so that users using assistive technologies to access online content can access the information these items provide.

Content that must easily be accessed by people with disabilities must conform to certain guidelines published by both the W3C and the United States government in order to produce accessible online output, also known as Section 508 compliant output. These guidelines are intended to help writers produce accessible content.

You can use ePublisher to help you produce online content that conforms to the W3C Web Content Accessibility Guidelines 1.0 (WCAG), Section 508 of the U.S. Rehabilitation Act of 1998, and the Americans with Disabilities Act (ADA). If you are required to generate accessible content, typically you provide the following items in your online content:

- Alternate text and descriptions for all images and image maps. For more information, see “Assigning Alternate Text to Images and Image Maps in FrameMaker”.
- Long descriptions for all images. For more information, see “Assigning Long Descriptions to Images in FrameMaker”.
- Summaries for all tables. For more information, see “Assigning Alternate Text (Summaries) to Tables in FrameMaker”.

You may also choose to provide the following items in your online content:

- Alternate text for abbreviations. For more information, see “Assigning Alternate Text to Abbreviations in FrameMaker”.
- Alternate text for acronyms. For more information, see “Assigning Alternate Text to Acronyms in FrameMaker”.
- Citations for quotes. For more information, see “Providing Citations for Quotes in FrameMaker”

You must prepare source documents and configure your ePublisher project in order to create accessible content. You prepare your source documents by inserting markers into your source documents and by applying character formats and paragraph formats. You configure accessibility settings in your ePublisher project. ePublisher uses the information in your source documents and your ePublisher project to generate accessible online output.

For more information about producing accessible content and to check your content further for compliance, see the following Web sites:

- For the complete W3C note on the WCAG, visit <http://www.w3c.org/TR/WCAG10-CORE-TECHS>.
- For information about the related Web Accessibility Initiative, visit <http://www.w3.org/WAI>.
- For information about Section 508 of the U.S. Rehabilitation Act of 1998, visit <http://www.w3.org/WAI/Policy/#508>.

## Accessible Content Navigation in FrameMaker

Users can navigate through the accessible content using keys on the keyboard. The following output formats support navigation keys:

- Dynamic HTML

- Microsoft HTML Help
- Oracle Help
- WebWorks Help

**Note:** For the Dynamic HTML, navigation key behavior may vary based on the browser the user uses. For example, in Netscape and Mozilla, users must hold down the `Alt` key while pressing the navigation keys. In Internet Explorer, users must first hold down the `Alt` key while pressing the navigation key, and then press `Enter`.

The following table lists the how each output format supports navigation keys.



Navigation Key	Function	Format
1	Display the TOC	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• WebWorks Help 5.0</li> </ul>
2	Display the Index	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• WebWorks Help 5.0</li> </ul>
3	Display the Search tab	WebWorks Help 5.0
4	Go to the previous page	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• Microsoft HTML Help</li> <li>• Oracle Help</li> <li>• WebWorks Help 5.0</li> </ul> <p><b><i>If you are using Microsoft HTML Help,</i></b> <code>Alt+4</code> works only if the topic pane has the focus. If the topic pane does not have the focus, you must press <code>Alt+0</code> and then <code>Alt+4</code>.</p> <p><b><i>If you are using Oracle Help,</i></b> you must press Enter after pressing <code>Alt+4</code>.</p>
5	Go to the next page	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• Microsoft HTML Help 1.x</li> <li>• Oracle Help</li> <li>• WebWorks Help 5.0</li> </ul>

Navigation Key	Function	Format
		<p><b><i>If you are using Microsoft HTML Help,</i></b> the <b>Alt+5</b> key works only if the topic pane has the focus. If the topic pane does not have the focus, you must press <b>Alt+ 0</b> and then <b>Alt +5</b>.</p> <p><b><i>If you are using Oracle Help,</i></b> you must press <b>Enter</b> after pressing <b>Alt +5</b>.</p>
6	Shift the focus to the related topics list displayed at the bottom of the current page	<p>WebWorks Help 5.0</p> <p>After you press the <b>6</b> key, you can press <b>Tab</b> to cycle through the entries in the related topics list.</p>
7	Display a blank feedback e-mail (equivalent to clicking the e-mail button in the toolbar frame)	WebWorks Help 5.0
8	Print the current page (equivalent to clicking the Print button in the toolbar frame)	WebWorks Help 5.0
9	Bookmark the current page (equivalent to clicking the Bookmark button in the toolbar frame)	WebWorks Help 5.0
10	Shift the focus to the topic frame (equivalent to clicking within the topic frame)	WebWorks Help 5.0

# Validating Accessible Content in FrameMaker

After you configure your source documents and configure the appropriate settings, ePublisher uses Accessibility conformance reports to perform the following checks to verify that the generated output conforms to accessibility standards:

- Alternate text for all images
- Alternate text for all clickable regions in all image maps
- Long descriptions for all images
- Summaries for all tables

ePublisher does not verify that you have provided alternate text for abbreviations or acronyms or verify that you have included citations for quotes. For more information about understanding and using the Accessibility conformance reports ePublisher provides, see “Configuring Reports”, and “Generating Reports”.

## Assigning Alternate Text to Images and Image Maps in FrameMaker

This section provides information about how to create accessible images and image maps in your generated output by assigning alternate text to images.

### Image and Image Map Alternate Text in FrameMaker

One of the largest accessibility challenges with online content today is the lack of alternative text for images and image maps. Sight-impaired users often use screen readers or refreshable Braille devices to read online content. However, when these assistive technologies come across images or image maps without alternative text, also known as alternate text, they are unable to provide users with information about the image or image map and its meaning.

The Web Content Accessibility Guidelines require that alternate text be provided for all images and image maps in online content. The alternate text is an image label that describes the image or each area of the image map. Online content should display alternate text for images and image maps when users perform the following actions:

- The user hovers the mouse pointer over an image or section of an image map.
- The user browser has been configured to disable display of images and image maps.

- The user browser is a text-only browser such as Lynx.
- The user uses assistive technology such as a screen reader.

The alternate text you assign to an image or sections of an image map should be as accurate and as succinct as possible and provide users with a brief description of the image and how the image relates to the page they are viewing. Make sure that your alternate text conveys all of the important information related to the image or image map section, but do not burden users with excessively long alternative text. Screen readers or refreshable Braille devices always read the alternative text, so if your page has several images or complex image maps with long descriptions, it can take a long time for the assistive devices to read image-heavy pages with long descriptions. If you need to provide a description of the image or image map section that is more than a few words or a few short sentences, you should provide a brief alternate text description of the image or image map section and then assign a longer description the image using either the `longdesc` attribute or a description. Once you specify a long description using the `longdesc` attribute, you can also optionally display a D link next to the image. For more information about assigning long descriptions to images, see “Assigning Long Descriptions to Images in FrameMaker”.

## Assigning Alternate Text to Images in FrameMaker

To assign alternate text to an image, your Stationery and template must have the ImageAltText marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to assign alternate text to images in the most current version of FrameMaker. The Object Attribute method can be used with newer versions of FrameMaker.

## To assign alternate text to an image in an Adobe FrameMaker source document using object attributes

1. In your Adobe FrameMaker source document, locate the anchored frame for the image to which you want to assign alternate text.
2. In the anchored frame that contains the image to which you want to assign alternate text, and complete the following:
  - a. Select the anchored frame that contains the image to which you want to assign an alternate name.
  - b. On the **Graphics** menu, click **Object Properties**.
  - c. Click **Object Attributes**.
  - d. In the **New or Changed Attribute** area, in the **Name** field, type **ImageAltText**.
  - e. In the **Definition** field, type the alternate text you want to assign to the image. Your text cannot exceed 255 characters.
  - f. Click **Add**.
  - g. Click **Set**.
  - h. Click **Set** again to close the window.

## To assign alternate text to an image in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the anchored frame for the image to which you want to assign alternate text.
2. In the anchored frame that contains the image to which you want to assign alternate text, insert a text frame by completing the following steps:
  - a. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
  - b. Click the **Text Frame** icon.
  - c. Drag the cursor over the portion of the image where you want to insert the text frame that will contain the ImageAltText marker.
  - d. In the Create New Text Frame window, in the **Number** field, type **1**, and then click **Set**.
  - e. Click outside the image, and then insert your cursor in the text frame.
3. On the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select **ImageAltText** from the drop-down list.
5. ***If the ImageAltText marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
6. In the **Marker Text** field, type the alternate text you want to assign to the image.
7. Click **New Marker**.
8. Save your Adobe FrameMaker source document.
9. Generate output for your project. For more information, see "Generating Output".
10. Verify ePublisher assigned the alternate text you specified to the image when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the image to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.

- c. Verify that the alternate text you specified is included in the `alt` tag for the image.

## **Assigning Alternate Text to Image Maps in FrameMaker**

To assign alternate text to an image, your Stationery and template must have the ImageAreaAltText marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to assign alternate text to an image map in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for assigning alternate text to an image map in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To assign alternate text for an image map in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the anchored frame for the image map to which you want to assign alternate text.
2. In the anchored frame that contains the image map to which you want to assign alternate text, complete the following steps for *each* area of an image map:
  - a. Insert your cursor into the text frame that defines a clickable region on the image map.
  - b. On the **Special** menu, click **Marker**.
  - c. In the **Marker Type** field, select **ImageAreaAltText** from the drop-down list.
  - d. ***If the ImageAreaAltText marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
  - e. In the **Marker Text** field, type the alternate text you want to assign to the image map area.
  - f. Click **New Marker**.
3. Save your Adobe FrameMaker source document.
4. Generate output for your project. For more information, see “Generating Output”.
5. Verify ePublisher assigned the alternate text you specified to each area of the image map when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the image map to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the alternate text you specified is included in the `alt` tag for each area of the image map.

## Assigning Long Descriptions to Images in FrameMaker



This section explains how to create accessible images in your generated output by assigning long descriptions to images.

## Image Long Descriptions in FrameMaker

The Web Content Accessibility Guidelines and Section 508 guidelines require you to include long descriptions for each image in an HTML document. You can use the `longdesc` attribute and a long descriptions stored in an external `.txt` file to assign a long description to an image. When you use this approach, the long descriptions are referenced in the HTML `<img>` tag in the `longdesc` attribute as shown in the following example:

```

```

The `longdesc` attribute in the `<img>` tag provides a link to a separate page where a long description is available. The link is invisible to sighted users, but when a conformant screen reader application reads the `longdesc` attribute, it loads the file referenced in the `longdesc` attribute and reads it. In the previous example, the screen reader would load and read the `mission.txt` file.

ePublisher provides the following options for assigning long descriptions to images:

- You can use the `ImageLongDescText` marker to assign a long description to an image. With this method, you assign a long description to an image using a description you include in a marker you insert into your source document. For more information, see “Assigning Long Descriptions to Images in FrameMaker”.
- You can use the `ImageLongDescByRef` marker to assign a long description to an image by referencing a long description saved in an external text (`.txt`) file. With this method, you specify the path to the external text file in a marker. For more information, see “Using Text in External Files to Assign Long Descriptions to Images in FrameMaker”.

If you assign long descriptions to some, but not all of your images, you can use the `ImageLongDescNotReq` marker. Use this marker when you use accessibility reports to verify that all images have long description but you have certain images in your source document that do not require a long description. For more information, see “Excluding Images from Accessibility Report Checks in FrameMaker”.

Although using the `longdesc` attribute is recommended in the Web Content Accessibility Guidelines and in 508 guidelines, older screen readers and many current browsers do not support this attribute and few online content developers use this attribute. As a result, the `longdesc` attributed benefits a only a small number of users. Only users who use modern screen readers can access the `longdesc` attribute easily. Older screen readers did not support this attribute. In addition, even users who use the latest version of screen reader may be unfamiliar

with the `longdesc` attribute and may not know how to access long descriptions using their screen reader because the `longdesc` attribute is used so infrequently in online content.

If you use the ImageLongDescText marker to assign long descriptions to images, as an interim solution ePublisher allows you to display a D link immediately after the image. The D link is an upper case letter D link that directs users to another page that contains the text you specified in the ImageLongDescText marker. Although a D link is not required for accessible Web pages, it can be used in addition to the `longdesc` attribute. The D link technique works in all browsers, but it is less elegant than using the `longdesc` attribute. Some users may be confused when they see a D link on the page, while other users will ignore the D link.

If you want to use D links in addition to the `longdesc` attribute when you generate output, your Stationery must have the D link option enabled. If you have permissions to modify target settings in ePublisher Express, you can enable the D link option setting in an ePublisher Express project. For more information about enabling the D link option in an ePublisher project, see “Specifying Accessibility Settings”. For more information about permissions required to modify target settings using ePublisher Express, see “Working with Target Settings”.

## Specifying Long Descriptions for Images in FrameMaker

To assign a long description to an image, your Stationery and template must have the ImageLongDescText marker type configured. Your output format must also support this feature.

When you use the ImageLongDescText marker to assign long descriptions to images, ePublisher generates an external text file that contains the long description you specify. When a conformant screen reader application reads the generated page, it loads the `.txt` file referenced in the `longdesc` attribute on the page and reads the file.

You can use the ImageLongDescText marker to assign long descriptions to images if your image descriptions do not exceed 255 characters. If your image long descriptions are longer than 255 characters, you must use an external `.txt` file and the ImageLongDescByRef marker to assign long descriptions to images, because Adobe FrameMaker limits the length of marker text to 255 characters. For more information, see “Using Text in External Files to Assign Long Descriptions to Images in FrameMaker”.

In addition, Adobe FrameMaker ignores carriage returns in marker text when generating MIF files. As result, if you use ImageLongDescText markers, each long description will be generated as a single paragraph.

The steps you use to assign long descriptions to images varies based on the version of Adobe FrameMaker you use. If you use Adobe FrameMaker 6.0, you use the ImageLongDescText marker. If you use Adobe FrameMaker 7.0 or later, you can

use either the ImageLongDescText marker or an object attribute you create for the anchored frame to assign a long description to an image. For more information about long descriptions and D links, see “Specifying Long Descriptions for Images in FrameMaker”.

## To assign a long description to an image using marker text in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the anchored frame for the image to which you want to assign a long description.
2. ***If you are using Adobe FrameMaker 6.0***, complete the following steps:
  - a. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
  - b. Click the **Text Frame** icon.
  - c. Drag the cursor over the portion of the image where you want to insert the text frame that will contain the ImageLongDescText marker.
  - d. In the Create New Text Frame window, in the **Number** field, type **1**, and then click **Set**.
  - e. Click outside the image, and then insert your cursor in the text frame.
  - f. On the **Special** menu, click **Marker**.
  - g. In the **Marker Type** field, select **ImageLongDescText** from the drop-down list.
  - h. ***If the ImageLongDescText marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
  - i. In the **Marker Text** field, type the long description you want to assign to the image. Your description cannot exceed 255 characters.
  - j. Click **New Marker**.
3. ***If you are using Adobe FrameMaker 7.0***, complete the following steps:
  - a. Select the anchored frame that contains the image to which you want to assign a long description using an external file.
  - b. On the **Graphics** menu, click **Object Properties**.
  - c. Click **Object Attributes**.
  - d. In the **New or Changed Attribute** area, in the **Name** field, type **ImageLongDescText**.

- e. In the **Definition** field, type the long description you want to assign to the image. Your description cannot exceed 255 characters.
  - f. Click **Add**.
  - g. Click **Set**.
  - h. Click **Set** again to close the window.
4. Save your Adobe FrameMaker source document.
  5. Generate output for your project. For more information, see “Generating Output”.
  6. Verify ePublisher assigned the long description to the image by completing the following steps:
    - a. On the **View** menu, click **Output Directory**.
    - b. In the *TargetName*\images folder, verify that ePublisher created a .txt file that contains the long description you specified in the ImageLongDescText marker, where *TargetName* is the name of your target.  
  
For example, if you specified a long description for *ImageName*.png, verify that ePublisher created an *ImageName*.txt file in the images folder, where *ImageName* is the name of the image to which you assigned a long description.
    - c. In the *TargetName*\ProjectName folder, open the page that contains the image to which you assigned the long description in Notepad and verify that the longdesc attribute references the *ImageName*.txt file ePublisher created for the image, where *TargetName* is the name of your target, *ProjectName* is the name of your project, and *ImageName* is the name of the image to which you assigned a long description.
    - d. **If you used the ImageLongDescText marker and the Stationery designer configured your Stationery to support D links**, open the page in a browser, verify that the D link displays in the browser, and then click the D link and verify that a page opens that displays the long description that you specified in the **ImageLongDescText** marker.

## Using Text in External Files to Assign Long Descriptions to Images in FrameMaker

Assign long descriptions to images using external files when you have image descriptions that exceed 255 characters or if you want to use image descriptions in external text files to assign long descriptions to images.

To assign a long description to an image, your Stationery and template must have the ImageLongDescText marker type configured. Your output format must also support this feature.

The steps you use to assign long descriptions to image varies based on the version of Adobe FrameMaker you use. If you use Adobe FrameMaker 6.0, you use the ImageLongDescByRef marker. If you use Adobe FrameMaker 7.0 or later, you use an object attribute you create for the anchored frame to assign a long description to an image.

## To assign a long description to an image using an external file in an Adobe FrameMaker source document

1. Create a `.txt` file that contains each image long description.
2. Place each image long description text file in a folder in the `ProjectName\Formats\TargetName\Files` folder for your project, where *ProjectName* is the name of your ePublisher project and *TargetName* is the name of your target.

For example, place the each image long description in the following location:

```
ProjectName\Formats\TargetName\Files  
\longdescriptions\imagelongdescription.txt
```

where *ProjectName* is the name of your ePublisher project, *TargetName* is the name of your target, *longdescriptions* is the name of the folder where you placed the image long description, and *imagelongdescription* is the name of the `.txt` file that contains the image long description.

3. In your Adobe FrameMaker source document, locate the anchored frame for the image to which you want to assign a long description.
4. Complete the following steps:
  - a. Select the anchored frame that contains the image to which you want to assign a long description using an external file.
  - b. On the **Graphics** menu, click **Object Properties**.
  - c. Click **Object Attributes**.
  - d. In the **New or Changed Attribute** area, in the **Name** field, type **ImageLongDescByRef**.
  - e. In the **Definition** field, type the path to the `.txt` file that contains the long description you want to assign to the image.

For example, type:

```
./longdescriptions/imagelongdescription.txt
```

where *longdescriptions* is the name of the folder where you placed the image long description, and *imagelongdescription* is the name of the `.txt` file that contains the image long description.

- f. Click **Add**.
- g. Click **Set**.

- h.** Click **Set** again to close the window.
- 5.** Save your Adobe FrameMaker source document.
- 6.** Generate output for your project. For more information, see “Generating Output”.
- 7.** In Output Explorer, verify ePublisher assigned the long description to the image using the long description in the external file when it generated output by completing the following steps:
  - a.** On the **View** menu, click **Output Directory**.
  - b.** In the *TargetName*\ProjectName folder, open the page that contains the image to which you assigned the long description using an external file in Notepad and verify that the `longdesc` attribute references the external text file that contains the long description for the image, where *TargetName* is the name of your target, and *ProjectName* is the name of your project.

## Excluding Images from Accessibility Report Checks in FrameMaker

In some instances, alternate text is sufficient for an image, and assigning a long description to an image in addition to alternate text would be redundant. However, you may have configured Accessibility reports to check for images without long descriptions and notify you when an image does not have a long description.

In this scenario, while you want an Accessibility report to notify you when you have an image without a long description, you do not want to be notified when you deliberately did not assign a long description to an image because assigning a both a long description and alternative text would be redundant. To address this issue, you can use the `ImageLongDescNotReq` marker to exclude an image that deliberately does not have a long description from validation when you generate Accessibility reports. For more information about Accessibility reports and configuring and generating Accessibility reports, see “Accessibility Reports”, “Configuring Reports”, and “Generating Reports”.

To exclude images without long descriptions from Accessibility reports, your Stationery and template must have the ImageLongDescNotReq marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to exclude images without long descriptions from Accessibility report checks in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for excluding images without long descriptions from Accessibility report checks in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.



## To exclude an image without a long description from Accessibility report checks in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the anchored frame for the image without a long description that you want to exclude from an Accessibility report check.
2. On the **Graphics** menu, click **Tools** to display the graphic tools palette.
3. Click the **Text Frame** icon.
4. Drag the cursor over the portion of the image where you want to insert the text frame that will contain the ImageLongDescNotReq marker.
5. In the Create New Text Frame window, in the **Number** field, type **1**, and then click **Set**.
6. Click outside the image, and then insert your cursor in the text frame.
7. On the **Special** menu, click **Marker**.
8. In the **Marker Type** field, select **ImageLongDescNotReq** from the drop-down list.
9. ***If the ImageLongDescNotReq marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
10. In the **Marker Text** field, do not enter any text. You do not need to enter any text in this field when you insert a ImageLongDescNotReq marker.
11. Save your Adobe FrameMaker source document.
12. Generate output for your project. For more information, see “Generating Output”.
13. Generate an Accessibility report and confirm that ePublisher did not generate an `Image is missing a long description` message for the image. For more information about generating Accessibility reports and Accessibility report messages, see “Generating Reports” and “Accessibility Report Messages”.

## Assigning Alternate Text (Summaries) to Tables in FrameMaker

Tables, just like images, are a way to visually display information. Although tables typically contain text, the purpose of the table is often not evident from text alone. The organization and display of the table may contain information that is not evident to assistive technologies. However, through the use of table summaries, assistive technologies can convey useful information to users about tables. The Web Content Accessibility Guidelines recommend that you provide summary text for each table in an HTML document. Table alternate text, or table summaries, provide users with information about what type of information the table contains.

You can create accessible tables by typing the table summary into a TableSummary marker. When ePublisher generates content, ePublisher puts the table summary you specify into the table in the `summary` attribute.

To assign alternate text to tables, your Stationery and template must have the TableSummary marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to assign alternate text to tables in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for assigning alternate text to tables in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To assign table summaries in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the table to which you want to assign a table summary.
2. Insert your cursor in front of the table.
3. On the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select **TableSummary** from the drop-down list.
5. **If the *TableSummary* marker type is not on the list**, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
6. In the **Marker Text** field, type the table summary you want to assign to the table.
7. Click **New Marker**.
8. Save your Adobe FrameMaker source document.
9. Generate output for your project. For more information, see "Generating Output".
10. Verify ePublisher assigned the table summary you specified to the table when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the table to which you assigned a table summary in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the table summary you specified is included in the `summary` attribute for the table.

## Excluding Tables from Accessibility Report Checks in FrameMaker

Tables used specifically for layout may not need a table summary. For example, if you use a table for layout, you probably would not assign a table summary to the table. However, you may have configured Accessibility reports to check for tables without table summaries and notify you when a table does not have a table summary.

In this scenario, while you want an Accessibility report to notify you when you have a table without a table summary, you do not want to be notified when you deliberately did not assign a table summary to a table because a table summary is not required. To address this issue, you can use the `TableSummaryNotReq` marker to exclude a table that deliberately does not have a table summary from validation when you generate Accessibility reports. For more information about Accessibility reports and configuring and generating Accessibility reports, see “Accessibility Reports”, “Configuring Reports”, and “Generating Reports”.

To exclude tables from Accessibility report checks, your Stationery and FrameMaker template must have the `TableSummaryNotReq` marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to exclude tables without table summaries from Accessibility report checks in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for excluding tables without table summaries from Accessibility report checks in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To exclude a table without a table summary from Accessibility report checks in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the table without a table summary that you want to exclude from an Accessibility report check.
2. Insert your cursor in front of the table.
3. On the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select **TableSummaryNotReq** from the drop-down list.
5. ***If the TableSummaryNotReq marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
6. In the **Marker Text** field, do not enter any text. You do not need to enter any text in this field when you insert a TableSummaryNotReq marker.
7. Click **New Marker**.
8. Save your Adobe FrameMaker source document.
9. Generate output for your project. For more information, see “Generating Output”.
10. Generate the Accessibility report and confirm that ePublisher did not generate an `Table is missing a table summary` message for the table. For more information about generating Accessibility reports and Accessibility report messages, see “Generating Reports” and “Accessibility Report Messages”.

## Assigning Alternate Text to Abbreviations in FrameMaker

Abbreviations are often used in written communication. Using an Abbreviation character format and an AbbreviationTitle marker, you can specify alternate text for abbreviations. For example, if your source document includes an abbreviation such as SS#, you can specify Social Security Number as alternate text for the abbreviation. When you use an AbbreviationTitle marker and Abbreviation character format to specify alternate text for an abbreviation, ePublisher adds the abbreviation alternate text you specify to the `title` attribute of the `abbr` tag in the output.

Following is an example of the HTML code produced when you specify Social Security Number as alternate text for SS#.

```
<th>First name</th>  
<th><abbr title="Social Security Number">SS#</abbr></th>
```

To assign alternate text to abbreviations, your Stationery and template must have the following items configured:

- Abbreviation character format
- AbbreviationTitle marker type

Your output format must also support this feature.

The following procedure provides an example of how to specify alternate text for abbreviations in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying alternate text for abbreviations in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To specify alternate text for an abbreviation in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the abbreviation for which you want to specify alternate text.
2. Apply the Abbreviation character format to the abbreviation text.
3. Insert your cursor anywhere inside the abbreviation.
4. On the **Special** menu, click **Marker**.
5. In the **Marker Type** field, select **AbbreviationTitle** from the drop-down list.
6. ***If the AbbreviationTitle marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
7. In the **Marker Text** field, type the abbreviation alternate text.
8. Click **New Marker**.
9. Save your Adobe FrameMaker source document.
10. Generate output for your project. For more information, see “Generating Output”.
11. Verify ePublisher assigned the abbreviation alternate text you specified when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the abbreviation to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the alternate text you specified for the abbreviation is included in the `abbr` tag in the `title` attribute.

## Assigning Alternate Text to Acronyms in FrameMaker

Acronyms are often used in written communication. Using an Acronym character format and an AcronymTitle marker, you can specify alternate text for acronyms. For example, if your document includes an acronym like NATO you can specify North Atlantic Treaty Organization as alternate text for the acronym. When you

use an AcronymTitle marker and an Acronym character format to specify alternate text for an acronym, ePublisher adds the acronym alternate text you specify to the `title` attribute of the `acronym` tag in the output.

Following is an example of the HTML code produced when you specify North Atlantic Treaty Organization as alternate text for NATO.

```
<p><acronym title="North Atlantic Treaty Organization">NATO</acronym>  
is a military alliance.</p>
```

To assign alternate text to acronyms, your Stationery and template must have the following items configured:

- Acronym character format
- AcronymTitle marker type

Your output format must also support this feature.

The following procedure provides an example of how to specify alternate text for acronyms in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying alternate text for acronyms in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.



## To specify alternate text for an acronym in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the acronym for which you want to specify alternate text.
2. Apply the Acronym character format to the acronym text.
3. Insert your cursor anywhere inside the acronym.
4. On the **Special** menu, click **Marker**.
5. In the **Marker Type** field, select **Acronym** from the drop-down list.
6. ***If the Acronym marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to “Implementing Online Features in FrameMaker”.
7. In the **Marker Text** field, type the acronym alternate text.
8. Click **New Marker**.
9. Save your Adobe FrameMaker source document.
10. Generate output for your project. For more information, see “Generating Output”.
11. Verify ePublisher assigned the acronym alternate text you specified when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the acronym to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the alternate text you specified for the acronym is included in the `[acronym]` tag in the `[title]` attribute.

## Providing Citations for Quotes in FrameMaker

A **citation** is a reference or footnote to a book, article, or other material that specifies the source from which a quotation was borrowed. A citation contains all the information necessary to identify and locate the work. Using a Citation character format and the Citation marker, you can specify citations for quotes that

enable users to go to a Web site that contains additional information about the quote.

Following is an example of the HTML code produced when you specify a citation for a quote.

```
<blockquote cite="http://shakespeare.mit.edu/lll/full.html">  
  <p>Remuneration! O! that's the Latin word for three farthings.  
  --- William Shakespeare (Love's Labor Lost).</p> </blockquote>
```

To provide citations for quotes, your Stationery and template must have the following items configured:

- Citation character format
- Citation marker type

Your output format must also support this feature.

The following procedure provides an example of how to specify citations for quotes in Adobe FrameMaker source documents using unstructured Adobe FrameMaker 7.2. Steps for specifying citations for quotes in Adobe FrameMaker may be different in other versions of Adobe FrameMaker.

## To specify citations for quotes in an Adobe FrameMaker source document

1. In your Adobe FrameMaker source document, locate the quotation for which you want to specify a citation.
2. ***If the quotation is a phrase within a paragraph***, complete the following steps:
  - a. Apply the Citation character format to the quotation phrase.
  - b. Insert your cursor anywhere inside the quotation phrase.
  - c. On the **Special** menu, click **Marker**.
3. ***If the quotation is a full paragraph***, complete the following steps:
  - a. Insert your cursor into the paragraph.
  - b. On the **Special** menu, click **Marker**.
4. In the **Marker Type** field, select **Citation** from the drop-down list.
5. ***If the Citation marker type is not on the list***, check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in FrameMaker".
6. In the **Marker Text** field, type the URL for the citation.
7. Click **New Marker**.
8. Save your Adobe FrameMaker source document.
9. Generate output for your project. For more information, see "Generating Output".
10. Verify ePublisher created the citation you specified when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the quotation for which you specified a quotation in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the citation you specified for the quotation is included in the `cite` attribute.

# Troubleshooting FrameMaker issues

Occasionally there might be issues with the source documents you are using. Below is a list linking to the wiki solutions website that will help you troubleshoot each one:

<b>Issue</b>	<b>For more information, see...</b>
If you are trying to use smart quotes	<a href="#"><u>Character Mapping in mapentrysets.xml</u></a>
If you are trying to turn off auto-hyphenation	<a href="#"><u>Turning off Auto-hyphenation</u></a>
If you are using autonumbering in anchored frames	<a href="#"><u>Autonumbered Paragraphs</u></a>
If you receive a "Cannot Duplicate Document" upon generation	<a href="#"><u>Cannot Duplicate Document</u></a>
If you receive a "Cannot Initialize API" message opening FrameMaker	<a href="#"><u>Cannot Initialize API</u></a>
If you are working with Change Bars	<a href="#"><u>Change Bars</u></a>
If you receive the "Error Communicating with FrameMaker" message upon generating	<a href="#"><u>Error Communicating with FrameMaker</u></a>
If you are not seeing your filename Markers in ePubublisher	<a href="#"><u>Filename markers are not working</u></a>
If you are seeing unwanted files in the temp directory upon generating	<a href="#"><u>Files being created in Temp directory</u></a>
If you are trying to use an image as a bullet	<a href="#"><u>Using an image and text together as a paragraph bullet</u></a>

<b>Issue</b>	<b>For more information, see...</b>
If you are trying to import an XLS file into FrameMaker	<a href="#">Problems with Imported Excel files</a>
If you are trying to link to an external PDF	<a href="#">Creating links outside of project structure</a>
If you are using multiple TOC files in FrameMaker	<a href="#">FrameMaker books with multiple TOC, Index, or Front-matter files</a>
If your title page material is not ordered properly	<a href="#">Title page material not ordered properly</a>
If you are having unexpected paragraph ordering with multiple text flows	<a href="#">FrameMaker paragraphs in different text flows have unexpected order in HTML</a>
If you are wanting to turn unbulleted text to bulleted text in the output	<a href="#">How do I turn regular FrameMaker text into bulleted text in my output?</a>
If you are working with structured content in text-boxes	<a href="#">Structured Elements in Text Boxes</a>
If you notice that some paragraphs in your output are formatting differently than others bearing the same style name	<a href="#">Unexpected changes in font size or other formatting for certain paragraphs</a>
If you notice a few lines of code that are very small at the end of the	<a href="#">Unexpected Code from FrameMaker plugin</a>

<b>Issue</b>	<b>For more information, see...</b>
HTML page from the FM document,	

# Microsoft Word

- Microsoft Word Templates and Standards
- Implementing Online Features in Word
- Working with the WebWorks Transit Menu for Word
- Working with Tables in Word
- Working with Images in Word
- Creating Index Entries in Word
- Using Variables in Word
- Using Conditions in Word
- Specifying Output File Names in Word
- Creating Context-Sensitive Help in Word
- Creating Popup Windows in Word
- Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word
- Creating Related Topics in Word
- Creating Links to PDF in Word
- Creating See Also Links in Word
- Creating Meta Tag Keywords in Word
- Assigning Custom Page Styles in Word
- Creating What's This (Field-Level) Help in Word
- Opening Topics in Custom Windows in Word
- Customizing TOC Entry in Word
- Customizing Table of Contents Icons in Word
- Specifying Context Plug-ins in Word
- Creating Accessible Online Content in Word
- Troubleshooting Word issues

If you want to implement online content features in your generated output, you need to prepare your Microsoft Word source documents for output generation. This section explains how to prepare your Microsoft Word source documents.

## Microsoft Word Templates and Standards

Microsoft Word is a powerful authoring tool that allows you to quickly create professional content. You can develop templates with the styles and other standard elements you need to deliver polished technical documentation. Microsoft Word provides many automation features to streamline the content development process. The new XML-based formats allow Microsoft Word to integrate content with other Microsoft Office products.

This section describes the design considerations for a Microsoft Word template. By effectively designing your Microsoft Word template, and by consistently applying styles throughout your source documents, you can streamline single-sourcing processes and reduce your production and maintenance costs. This section does not describe all Microsoft Word processes, but it focuses on the design considerations related to ePublisher.

## Word Standards to Support Single-Sourcing



To define your Microsoft Word standards, create a template `.dot` file with all the elements you need in it, including styles, variables, autotext, toolbars, macros, and page layouts. You can use this file to create all new source documents and to update the styles, autotext, toolbars, and macros in existing source documents. You can also use this file to create a source document to include in your Stationery design project.

The following sections describe various template areas and considerations, and how to effectively design your Microsoft Word template to support single-sourcing with ePublisher.

## Microsoft Word Template File

The template `.dot` file defines the default styles, autotext, toolbars, and macros for documents attached to that template. You can also define the starting variables, properties, content, sections, and page layouts for new documents created from the template. The existing bulleted and numbered list items in the template file preload the bullet and number gallery in Microsoft Word, which can cause inconsistencies on multiple computers when these items are not properly defined in the template.

When you create a new document based on a template, the contents of the template file, including text, paragraphs, sections, headers, and footers are copied to the new document. In addition, the new document is attached to the template and uses the styles, autotext, toolbars, and macros defined in the template.

You can also attach an existing document to a template. The document can then use the autotext, toolbars, and macros defined in the template. If you select the **Automatically update document styles** option, the style definitions in the template overwrite the style definitions in the document. The existing document content is not changed, including the headers, footers, and page layout. Style modifications on individual paragraphs, such as the Keep with Next setting, are also not changed or reset. You can create a macro to find each style and reset the paragraph to the default style, which removes the modifications.

## Creating a Clean Base Template File

When you create custom styles and macros in Microsoft Word, these customizations are often stored in the default `Normal.dot` file on your computer. In addition, when you use Microsoft Word as your email editor in Microsoft Outlook, some other customizations can be stored in the default `Normal.dot` file. When you create a new Microsoft Word template, you want to use a clean `Normal.dot` file as the starting point for your template. Save the template file with a new name, and then customize the template as needed. To simplify your template use and maintenance, delete the unneeded styles, variables, autotext, and macros.

## To create a clean base template file

1. On the **Tools** menu in Microsoft Word, click **Options**.
2. On the **File Locations** tab, find where the User Templates are stored.
3. Close Microsoft Word and all other Microsoft Office products.
4. Open the folder where the User Templates are stored and rename the `Normal.dot` file in that location in case you later need any customizations stored in that file.
5. Open Microsoft Word.
6. On the **File** menu, click **Save As**.
7. In **Save as type**, select **Document Template (\*.dot)**.
8. Specify the name and location for your new template, and then click **Save**.

You can customize the new template to meet your specific needs. To create a new tab in the Template Selection window in Microsoft Word, you can create a folder within the User Templates location. Then, you can store your new template in that folder.

## Paragraph Styles in Word

Create paragraph styles for items based on function, not based on formatting. This approach allows you to modify formatting over time and the style names continue to apply. It also prepares you for structured writing in the future.

Name your paragraph styles starting with naming conventions that group styles by function. For example, group procedure-related styles together by starting the style names with Procedure, such as ProcedureIntro, ProcedureStep, and ProcedureSubstep.

**Note:** Style names should not include a period in their name. The period can cause display issues when ePublisher creates the cascading style sheet entry that defines the appearance of the style.

To simplify formatting and save time for future maintenance and customization, set the default paragraph font and spacing for a base style, such as Normal. Then, base other styles on this base style to inherit the default formatting settings. This process allows you to quickly modify fonts and spacing across styles by modifying only the base style. You can customize settings for each style as needed. The customized settings are not affected when you modify those settings in the base style. To simplify maintenance for heading styles, which often use a different font

than your content styles, you may want to base all heading styles on the Heading 1 style to define the font for all headings.

In ePublisher, you can scan the source documents to list all the paragraph styles. Then, you can organize them in ePublisher to allow property inheritance and to streamline the customization process for your generated output.

You may need multiple paragraph styles to define functions that support pagination settings, such as a BodyListIntro format that has **Keep with next** set. To reduce the number of paragraph styles, you can customize paragraphs to add the **Keep with next** setting as needed. Customizing this setting on a paragraph does not affect the ability for the paragraph to receive the other formatting settings from the style definition.

To automate and simplify template use, define the paragraph style that follows each paragraph style. This process allows the writer to press Enter after writing a paragraph and the template creates the next paragraph with the style most commonly used next. For example, after a Heading style, the writer most often writes a body paragraph of content.

Common paragraph styles include:

- Figure paragraphs. You may need multiple indents, such as Figure, FigureInList, and FigureInList2.
- Body paragraphs. You may need multiple indents, such as Body, BodyInList, and BodyInList2. To reduce training needs, you can use the default style names, such as Body Text, Body Text Indent, and Body Text Indent 2.
- Headings, such as Heading 1, Heading 2, Heading 3, and Heading 4. You may also need specialized headings, such as Title, Subtitle, FrontMatterHeading1, FrontMatterHeading2, and FrontMatterHeading3. The cross-reference feature in Microsoft Word allows you to create cross references to headings that use the default Heading styles named Heading X, where X is a number. To create cross references to other styles, use bookmarks. Do not paste content at the beginning of a heading. Existing cross references to that heading may include the pasted content when the cross references are updated.
- Bulleted lists. You may need multiple bullet levels, such as Bullet, Bullet2, Bullet3. You may also need a bullet item within a procedure, such as a ProcedureBullet and a bullet item within a table, such as a CellBullet. For more information, see "Bulleted and Numbered Lists in Word".
- Numbered lists. You may need multiple levels, such as ProcedureStep that uses numbers and ProcedureSubstep that uses lowercase letters. You may also need numbered list items in tables, such as CellStep. Be sure to consider related supporting formats, such as ProcedureIntro. For more information, see "Bulleted and Numbered Lists in Word".

- Examples, such as code or command syntax statements, usually in a fixed font. To keep the lines of a code example together, you can set the **Keep with next** setting for the Example style and use an ExampleLast style to identify the end of the example. You may also need multiple example levels, such as ExampleInList and ExampleInListLast.
- Paragraphs in tables, such as CellHeading, CellBody, CellBody2, CellStep, and CellBullet.
- Legal notice and copyright or trademark styles for inside the cover page.
- Table of contents and Index styles.
- Definition lists, such as term and definition or description. You can use a two-column table for this purpose, but a definition list allows long terms, such as field labels in a user interface, to run across the page without wrapping. Then, the definition or description are indented below the term.
- Header and footer styles to control formatting.
- Notes, cautions, tips, and warnings.

ePublisher projects use custom field code markers, paragraph styles, and character styles to define online features. You need to give the list of markers and styles to the writers so they know how to implement each online feature. The writers use the markers and styles you create to define online features.

The Stationery defines the custom markers and styles. To reduce complexity, you can use the style names defined in the documentation, or you can define the online feature to a different style. The following list identifies additional paragraph styles you may need to support ePublisher online content features:

- Paragraph or character styles to support multiple languages, such as bidirectional languages and text.
- Dropdown paragraph style that identifies the start of an expand/collapse section. You can end the section with a paragraph style defined to end the section, or with a DropDownEnd marker.
- Popup paragraph styles that define several aspects of popup window content:
  - Popup paragraph style identifies the content to display in a popup window and in a standard help topic. This style is applied to the first paragraph of popup content.
  - Popup Append paragraph style identifies the content to display in a popup window and in a standard help topic. This style is applied to additional popup paragraphs when you have more than one paragraph of content to include in a popup window.

- Popup Only paragraph style identifies the content to display only in a popup window. This style is applied to the first paragraph of popup content.
- Popup Only Append paragraph style identifies the content to display only in a popup window. This style is applied to additional popup paragraphs when you have more than one paragraph of content to include in a popup window.
- Related topics paragraph style that identifies a link to a related topic, such as a concept topic related to a task or a task related to a concept.
- See Also paragraph style that identifies the text you want to include in an inline See Also link.

For more information about enabling a specific online feature, see “Designing, Deploying, and Managing Stationery”.

## Character Styles in Word

Create character styles for items based on function, not based on formatting or appearance. This approach allows you to modify formatting over time and the style names continue to apply. It also prepares you for structured writing in the future.

You can apply only one character style to a set of text. If you apply a second character style to that text, it replaces the initial character style. Therefore, you may need more character styles to address all the possible combinations, such as variables in a paragraph and variables in a code sample.

Common character styles include:

- Book titles in cross references
- Emphasized text
- Command names
- File and folder names
- User interface items
- Optional steps or if clauses used to introduce optional steps
- Links
- New terms
- Step numbers, which allows you to apply formatting to the step number defined with a sequence field code

- Text the user must type
- Variables

ePublisher projects use custom field code markers and styles to define online features. You need to give the list of markers and styles to the writers so they know how to implement each online feature. The writers use the markers and styles you create to define online features.

The Stationery defines the custom markers and styles. To reduce complexity, you can use the style names defined in the documentation, or you can define the online feature to a different style. The following list identifies additional character styles you may need to support ePublisher online content features:

- Multiple language support, such as bidirectional languages and text, can require a paragraph or character style with Bidi support enabled.
- Abbreviation character style identifies abbreviation alternate text for browsers to display for abbreviations, such as SS#, when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. This character style is used in combination with the AbbreviationTitle marker type.
- Acronym character style identifies acronym alternate text for browsers to display for acronyms, such as HTML, when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. This character style is used in combination with the AcronymTitle marker type.
- Citation character style identifies the source of a quote using a fully-qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. This character style is used in combination with the Citation marker type.
- See Also character style identifies the text you want to include in a See Also button. This style controls the appearance of the text on the button.

For more information about enabling a specific online feature, see “Designing, Deploying, and Managing Stationery”.

## Bulleted and Numbered Lists in Word

ePublisher uses a table-like structure with two columns to display any paragraph style with a hanging indent, such as bulleted and numbered list items, in generated output. ePublisher uses the numbers, characters, and fonts from the source documents for the bullets or numbers. Since some fonts are not available on all computers, you should use character styles in ePublisher to override the formatting of the bullets or numbers. You can also use an image in ePublisher for bullets.

Bulleted and numbered list items can have inconsistent formatting from one computer to another. The formatting is defined by the Bullets and Numbering style gallery. To correctly populate the style gallery, the source document must have an example of each type of bulleted and numbered item. Leave the correct example of each item in the document until that style type is used. If a correctly formatted item does not exist in the document, copy the correct style from the template and paste it in the document to create the first item with that style.

**Note:** Be aware of paragraphs that have a hanging indent. The hanging indent can cause incorrect alignment of text on the first line of your generated output. For more information see “Defining the Appearance of Numbered Lists”.

## Bulleted Lists in Word

For bulleted lists, you may need multiple bullet levels, such as Bullet, Bullet2, and Bullet3. You may also need a format for a bullet within a procedure, such as a ProcedureBullet, and a bullet within a table, such as a CellBullet. Make sure you consider all supporting formats you may need, such as a ListIntro format for the paragraph that introduces the bulleted list, which should be set to stay with the list (Keep with Next).

Do not base two bulleted items with different bullets on the same base style. Otherwise, when you modify the bullet for one bulleted item style, the bullet on the other item is also affected.

## Numbered Lists in Word

For numbered lists, you may need multiple levels, such as a ProcedureStep that uses numbers and a ProcedureSubstep that uses lowercase letters. You may also need a numbered list item in tables, such as CellStep. Make sure you consider all supporting formats you may need, such as a ProcedureIntro format.

Microsoft Word can have issues with the built-in autonumbering when restarting the numbering in lists. To avoid these issues, you can use sequence field codes to completely control numbering in your source documents. You can define a paragraph style with a hanging indent for each numbered step item. Define a character style and apply it to the sequence field code to control how the numbered list appears both in print and in your generated output. Autotext can help you automate list creation. You can also create a macro to quickly number a set of paragraphs. To restart a field code, add the `\r1` option.

## Image Styles and Considerations in Word

If ePublisher cannot use an original image in the output, or if ePublisher determines it needs to modify the image based on how it is included in the source document, ePublisher rasterizes the image using the options you define for your graphic styles in Style Designer. For example, you can define the dots per inch (DPI) and format

for the final images. Rasterization of an image can cause the image to be less clear in the output.

To avoid reduced image quality in your output, and to avoid an extended transformation time during the Image stage and pipeline, review the following considerations:

- When ePublisher encounters an image in your Microsoft Word source documents, ePublisher checks for the following conditions:
  - \_ Is the frame a different size than the original image?
  - \_ Is there white space in the frame with the image?
  - \_ Is the image copied into the document, rather than imported by reference?
  - \_ Is the original image a file format other than `.jpg`, `.gif`, `.png`, or `.svg`?
  - \_ Are there additional elements in the frame, such as text boxes, multiple images, or callouts?

If ePublisher determines that any of these conditions apply, ePublisher rasterizes the entire frame and applies the options you defined in Style Designer.

- To display images at full size in online output and avoid resizing, which can cause the image to be rasterized, set the **By reference graphics use document dimensions** option for your graphic styles to **Disabled**.
- If you want ePublisher to rasterize all images according to your Style Designer options, set the **By reference graphics** option to **Disabled** for all graphic styles.
- When ePublisher finds an image included by-reference that is the original size and contains no callouts, ePublisher copies the image directly into the output folder in most cases, bypassing the graphic style options.
- To improve the image quality in your output, resize your images as needed using an image editing application before importing them, rather than adjusting the size in Microsoft Word. Otherwise, an image included by reference retains its original file size, and it is either scaled by the browser or rasterized according to the size in the source document, which can result in a distorted image.
- For the best compatibility with most computer monitors, save and import your images at 96 DPI using a format that ePublisher does not rasterize.



- Image callouts are useful in many publications. However, text boxes and line drawings cause images to be rasterized, which can make images less clear in your output. Add and edit callouts in your image editing application and then import the single, final image to avoid the rasterization process.
- You can add text boxes with GraphicStyle markers to your images without causing the image to be rasterized, since markers do not affect the appearance of the image.
- Store image files and source documents on the local computer when generating output.

## To achieve the best results when inserting images in Microsoft Word

1. Create a unique paragraph style for images. Use the paragraph alignment properties to control the position of your images.
2. Import your image file by reference onto the unique paragraph rather than copying it into the document. ePublisher supports only `.jpg`, `.gif`, `.png`, and `.svg` files. ePublisher rasterizes all other formats.
3. Do not resize the image in Microsoft Word.

## Table Styles in Word

Table styles, which are available in recent versions of Microsoft Word, allow you to define standard tables and quickly create tables with those standards in your source documents. If your version of Microsoft Word does not support table styles, use the TableStyle marker to specify the style to apply in ePublisher that defines the appearance of the table.

Table styles are often overlooked in Microsoft Word. The default template provides many default table styles, such as Table Grid and Table Normal. You can create custom table styles for your specific requirements. Define table header rows for each table that repeat when the table splits across pages, and do not allow rows to break across pages, which can create awkward breaks within tables in your printed content. You can use autotext to quickly create standard tables in your source documents.

When you define your table styles, be sure to consider the various types of tables you may need, such as with lines, without lines, checklists, and action/result tables. You can use a table without lines to layout content within an area on a page, such as a definition list with short terms. You can also create a table style for each indent position needed. For example, you can create a table style to use for tables within a bulleted list that is indented to align with the text of each bulleted list item.

ePublisher allows you to define how the header, footer, and main rows of a table appear in your generated output. To support these formatting properties, your tables must have each of these parts defined in your source documents. If a table does not have a header defined, ePublisher cannot apply the formatting defined for the header row. Microsoft Word does not support table footers, so the footer formatting settings in ePublisher do not apply to Microsoft Word source documents.

ePublisher applies the paragraph and character styles you define for content within each cell. You can also configure ePublisher to ignore character styles in a table. You may need additional paragraph styles to use in tables, such as CellBody and CellBullet, so you can define the proper margins and appearance for your generated output.

# Field Codes

Microsoft Word uses field codes to implement standard features, such as index entries and variables. You can use sequence field codes for numbering, such as numbering chapters, steps, figure captions, and table captions. ePublisher recognizes many of the standard field codes and uses them to implement these standard features in your generated output.

ePublisher projects also use custom field codes (markers) and styles to define online features. The toolbar provided by ePublisher in Microsoft Word allows you to quickly insert the custom markers. You need to give the list of markers and styles to the writers so they know how to implement each online feature. The writers use the markers and styles you create to define online features.

The Stationery defines the custom markers and styles. Markers with reserved names have their functions defined by default. You can use these default names, or you can create your own markers. To reduce complexity, use the default marker names, which are also used throughout the documentation. You can also use the style names defined in the documentation to reduce complexity. The following table lists the default custom marker types used to implement online features.

<b>Marker Type</b>	<b>Description</b>
AbbreviationTitle	Specifies abbreviation alternate text for browsers to display for abbreviations such as SS# when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. Used in combination with the Abbreviation character format.
AcronymTitle	Specifies acronym alternate text for browsers to display for acronyms such as HTML when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. Used in combination with the Acronym character format.
Citation	Specifies the source of a quote using a fully qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. Used in combination with the Citation character format.
Context Plugin	Specifies context plug-ins for Eclipse help systems. Other Eclipse plug-ins can use the context plug-in IDs to call the Eclipse help system. For more information, see "Using Markers to Specify Context Plug-ins in Eclipse Help".
DropDownEnd	Marks the end of an expand/collapse section. Used in conjunction with an Expand/Collapse paragraph format.
Filename	Specifies the name of an output file for a page or an image.
GraphicScale	Specifies a percentage to use to resize an image, such as 50 or 75 percent, in generated output.
GraphicStyle	Specifies the name of a graphic style defined in a project to apply to an image. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality.

Marker Type	Description
Hypertext	Specifies a link using the <code>newlink</code> and <code>gotolink</code> commands in Adobe FrameMaker. This marker type is a default Adobe FrameMaker marker type ePublisher automatically maps.
ImageAltText	Specifies alternate text for an image. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content.
ImageAreaAltText	Specifies alternate text for clickable regions in an image map. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content.
ImageLongDescByRef	Specifies the path to the file that contains the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content.
ImageLongDescNotReq	Specifies that a long description is not required for an image, which bypasses this accessibility check for the image when you create accessible content.
ImageLongDescText	Specifies the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content.
Keywords	Specifies the keywords to include in the <code>meta</code> tag for the topic. The <code>meta</code> tag improves searchability on the Web.
PageStyle	Specifies the name of a page style defined in the project to apply to a topic. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality.

Marker Type	Description
PassThrough	Specifies that ePublisher place the contents of the marker directly into the generated output without processing the content in any way. For example, you could use a PassThrough marker if you wanted to embed HTML code within your generated output.
Popup	Specifies the start of the content to include in a popup window. The content is displayed in a popup window when you hover over the link. When you click the link in some output formats, the topic where the popup text is stored, such as the glossary, is displayed.
PopupEnd	Marks the end of the content to include in a popup window.
PopupOnly	Specifies the start of the content to include in only a popup window. Browsers display the content in a popup window when you hover over or click the link.
RubiComposite	No longer supported.
SeeAlsoKeyword	Specifies an internal identifier for a topic. SeeAlsoLink markers in other topics can list this identifier to create a link to this topic. Used in conjunction with a See Also paragraph format or character format.
SeeAlsoLink	Identifies an internal identifier from another topic to include in the list of See Also links in this topic. Used in conjunction with a See Also paragraph format or character format.
SeeAlsoLinkDisplayType	Specifies whether to display the target topics on a popup menu or in a window. By default, the links are displayed in the Topics Found window. To display a popup menu, set the value to <code>menu</code> . This marker type is supported only in HTML Help.
SeeAlsoLinkWindowType	Specifies the name of the window defined in the <code>.hhp</code> file, such as TriPane or Main, that the topic opens

Marker Type	Description
	in when the user clicks the link. This marker type is supported only in HTML Help.
TableStyle	Specifies the name of a table style defined in the project to apply to a table in versions of Microsoft Word that did not support table styles. This marker type is an internal marker type that is not displayed in Stationery Designer. This marker type is supported only for Microsoft Word documents. You cannot create a marker type with a different name and assign it this functionality.
TableSummary	Specifies an alternate text summary for a table, which is used when you create accessible content. This text is added to the <code>summary</code> attribute of the <code>table</code> tag in the output. Screen readers read this description when you create accessible content.
TableSummaryNotReq	Specifies that a summary is not required for a table, which bypasses this accessibility check for that table.
TOCIconHTMLHelp	Identifies the image to use as the table of contents icon for a topic in the HTML Help output format.
TOCIconJavaHelp	Identifies the image to use as the table of contents icon for a topic in the Sun JavaHelp output format.
TOCIconOracleHelp	Identifies the image to use as the table of contents icon for a topic in the Oracle Help output format.
TOCIconWWHelp	Identifies the image to use as the table of contents icon for a topic in the WebWorks Help output format.
TopicAlias	Specifies an internal identifier for a topic that can be used to create a context-sensitive link to that topic.
TopicDescription	Specifies a topic description for a context-sensitive help topic in Eclipse help systems. For more information, see "Using Markers to Specify Topic Descriptions for Context-Sensitive Help Topics in Eclipse Help".

Marker Type	Description
WhatIsThisID	Identifies a What Is This help internal identifier for creating context-sensitive What Is This field-level help for Microsoft HTML Help.
WindowType	Specifies the name of the window defined in the help project that the topic should be displayed in. In Microsoft HTML Help, the window names are defined in the <code>.hhp</code> file. This marker type is supported in Microsoft HTML Help and Oracle Help.

## AutoText, AutoCorrect, and User-Defined Hotkeys

Autotext improves consistency and efficiency for your writing team. Writers can quickly create items, such as tables and lists, with the correct styles applied by default. You can create autotext for many common items:

- 2-, 3-, 4-, and 5-column tables
- Checklists
- Notes, tips, cautions, and warnings
- Cross reference introductory phrases, such as `For more information, see`
- New chapters and standard topics, such as tasks or command reference topics

AutoCorrect allows you to automatically fix common typing mistakes, including two capital letters in a row and misspelled words. You can also use autocorrect to define some shorthand character sequences that automatically insert longer common phrases. For example, you can define `acl` to be replaced with `access control list`. When you type `acl` and a space, Microsoft Word changes it to `access control list`.

Microsoft Word also allows you to define hot keys for common tasks, such as applying a style, inserting autotext, or running a macro. With hotkeys, autotext, and autocorrect, you can create content more efficiently.

## Toolbars and Menus in Word

Create custom toolbars with the options you use most often. You can copy toolbar buttons from the Transit menu and other toolbars to create one combined toolbar



with drop-down menus and selections based on your specific requirements. Helpful commands to include on your toolbars are Default Paragraph Font, Keep with Next, and Show/Hide field codes.

## **Variables and Conditions in Word**

On the Properties tab, you can define custom variables to use throughout your source documents. To insert a variable in your content, create field code that references the custom property you created. You can also use field codes to display the contents of a specific style type, such as the previous Heading 1.

For conditions in your generated output, you can use the field codes (markers) supported by ePublisher. You can also use paragraph and character styles to identify conditional content. With this style approach, you need to create extra styles for each condition you need. Then, you can create multiple templates that show or hide specific styles. By attaching the appropriate template, you can include or exclude the appropriate content in your printed output. You can also include or exclude content from your generated output based on styles.

## **Page Layouts and Sections in Word**

You should define all the sections you need in your template. Each section is separated by a section break and has its own page setup. The table of contents and index often have multiple section breaks to customize the page layout in those sections and correctly generate the lists based on field codes. You need to be careful when working around section breaks. If you delete a section break, the page layout for the section, including the headers and footers, may be changed.

In the headers and footers, use field codes to display the contents of a specific style from the associated section, such as the Title style from the title page to include the book title in the footer. This type of field code is automatically maintained and updated. If you use a variable field code in the headers or footers, you need to manually update those field codes. Variable field codes in headers and footers are not updated automatically with the rest of the document content.

## **Table of Contents and Index in Word**

Since the appearance of online table of contents and indexes often differ from printed versions, you need to be able to deliver customized table of contents and indexes in your online content. Therefore, ePublisher does not need the table of contents and index formatting defined in your source documents. ePublisher allows you to define the table of contents levels and appearance, as well as the appearance of the index in your generated output. ePublisher uses the index field codes throughout your source documents to build the online index. This support allows you to deliver the online content you require.

## **Automation with Macros in Word**

Macros allow you to automate, customize, and extend the default Microsoft Word capabilities, including some common tasks and maintenance operations:

- Deleting hidden table of contents bookmarks that build up over time
- Production book tasks, such as updating fields and paginating multiple files
- Attaching different templates for conditional text implementations

This automation can streamline the content authoring process and save you valuable time and effort. Microsoft Word provides a VBA (Visual Basic for Applications) environment for macros to give you the automation capabilities you need. To enable macros, set the security level to medium. You can record steps to perform a specific task, and then copy and edit the generated code to do exactly what you need.

## Implementing Online Features in Word

Implement online features in your output by preparing your Microsoft Word source documents with custom marker types, paragraph styles, and character styles defined by the Stationery designer for your Stationery. These markers and styles define the presentation and behavior of your online content. For example, markers can define the name of the file generated for a topic. Formats can define how content displays online.

### Custom Marker Types in Word

ePublisher projects use the custom marker types to implement online features when generating output. Custom markers are created using custom marker types available in the WebWorks Transit menu for Microsoft Word. ePublisher inserts markers based on marker types as field codes in your Microsoft Word source documents.

Before you begin using custom marker types to implement online features, talk to the Stationery designer and verify which online features your Stationery supports. Your Stationery only recognizes the custom marker types defined by the Stationery designer in your Stationery. If you try to implement online features using custom marker types not supported in your Stationery, ePublisher does not recognize these items when generating output.

When the Stationery designer creates the Stationery, the Stationery designer can use the default name for a custom marker type or the Stationery designer can use a different name for the customer marker type. The following table lists the default names of custom marker types used to implement online features. Always verify with the Stationery designer the names of the custom marker types you should use when implementing online features before you use these items in your source documents. If you need to create a custom marker type to implement an

online feature, verify with the Stationery designer that your Stationery supports the custom marker type before you create the custom marker type and insert and use the custom marker in a source document. For more information about creating custom marker types, see "Creating Custom Marker Types Using the WebWorks Transit Menu in Word".

<b>Marker Type</b>	<b>Description</b>
AbbreviationTitle marker type	Specifies abbreviation alternate text for browsers to display for abbreviations such as SS# when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. Used in combination with the Abbreviation character style. For more information, see "Assigning Alternate Text to Abbreviations in Word"
AcronymTitle marker type	Specifies acronym alternate text for browsers to display for acronyms such as HTML when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. Used in combination with the Acronym character style. For more information, see "Assigning Alternate Text to Acronyms in Word".
Citation marker type	Specifies the source of a quote using a fully qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. Used in combination with the Citation character style. For more information, see "Providing Citations for Quotes in Word".
Context Plugin marker type	Specifies context plug-ins for Eclipse help systems. Other Eclipse plug-ins can use the context plug-in IDs to call the Eclipse help system. For more information, see "Specifying Context Plug-ins in Word".
DropDownEnd marker type	Marks the end of an expand/collapse section. Used in conjunction with an Expand/Collapse paragraph style. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word".
Filename marker type	Specifies the name of an output file for a page or an image. For more information, see "Specifying Output File Names in Word".
GraphicScale marker type	Specifies a percentage to use to resize an image, such as 50 or 75 percent, in generated output. For more information, see "Assigning Image Scales in Word".

Marker Type	Description
GraphicStyle marker type	Specifies the name of a image style defined in a project to apply to an image. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality. For more information, see "Assigning Image Styles in Word".
ImageAltText marker type	Specifies alternate text for an image. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content. For more information, see "Assigning Alternate Text to Images in Word".
ImageAreaAltText marker type	Specifies alternate text for clickable regions in an image map. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output. Screen readers use this text when you create accessible content. For more information, see "Assigning Alternate Text to Image Maps in Word".
ImageLongDescByRef marker type	Specifies the path to the file that contains the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content. For more information, see "Assigning Long Descriptions to Images in Word".
ImageLongDescNotReq marker type	Specifies that a long description is not required for an image, which bypasses this accessibility check for the image when you create accessible content. For more information, see "Excluding Images from Accessibility Report Checks in Word".
ImageLongDescText marker type	Specifies the long description for an image. This text is added to the <code>longdesc</code> attribute of the <code>img</code> tag in the output. Screen readers read this description when you create accessible content. For more information, see "Specifying Long Descriptions for Images in Word".
Keywords marker type	Specifies the keywords to include in the <code>meta</code> tag for the topic. The <code>meta</code> tag improves searchability on the

Marker Type	Description
	Web. For more information, see "Creating Meta Tag Keywords in Word".
PageStyle marker type	Specifies the name of a page style defined in the project to apply to a topic. This marker type is an internal marker type that is not displayed in Stationery Designer. You cannot create a marker type with a different name and assign it this functionality. For more information, see "Assigning Custom Page Styles in Word".
PassThrough	Specifies that ePublisher place the contents of the marker directly into the generated output without processing the content in any way. For example, you could use a PassThrough marker if you wanted to embed HTML code within your generated output.
Popup marker type	Specifies the start of the content to include in a popup window. The content is displayed in a popup window when you hover over the link. When you click the link in some output formats, the topic where the popup text is stored, such as the glossary, is displayed. For more information, see "Using Markers to Create Popup Windows in Word".
PopupEnd marker type	Marks the end of the content to include in a popup window. For more information, see "Using Markers to Create Popup Windows in Word".
PopupOnly marker type	Specifies the start of the content to include in only a popup window. Browsers display the content in a popup window when you hover over or click the link. For more information, see "Using Markers to Create Popup Windows in Word".
RubiComposite marker type	No longer supported.
SeeAlsoKeyword marker type	Specifies an internal identifier for a topic. SeeAlsoLink markers in other topics can list this identifier to create a link to this topic. Used in conjunction with a See

Marker Type	Description
	Also paragraph style or character style. For more information, see "Creating See Also Links in Word".
SeeAlsoLink marker type	Identifies an internal identifier from another topic to include in the list of See Also links in this topic. Used in conjunction with a See Also paragraph style or character style. For more information, see "Creating See Also Links in Word".
SeeAlsoLinkDisplayType marker type	Specifies whether to display the target topics on a popup menu or in a window. By default, the links are displayed in the Topics Found window. To display a popup menu, set the value to <code>menu</code> . This marker type is supported only in HTML Help. For more information, see "Creating See Also Links in Word".
SeeAlsoLinkWindowType marker type	Specifies the name of the window defined in the <code>.hhp</code> file, such as TriPane or Main, that the topic opens in when the user clicks the link. This marker type is supported only in HTML Help. For more information, see "Creating See Also Links in Word".
TableStyle marker type	Specifies the name of a table style to apply to a table. This marker type is an internal marker type that is not displayed in the Style Designer as a marker. You cannot create a marker type with a different name and assign it this functionality. For more information, see "Applying Table Styles in Word"
TableSummary marker type	Specifies an alternate text summary for a table, which is used when you create accessible content. This text is added to the <code>summary</code> attribute of the <code>table</code> tag in the output. Screen readers read this description when you create accessible content. For more information, see "Assigning Alternate Text (Summaries) to Tables in Word".
TableSummaryNotReq marker type	Specifies that a summary is not required for a table, which bypasses this accessibility check for that table. For more information, see "Excluding Tables from Accessibility Report Checks in Word".

<b>Marker Type</b>	<b>Description</b>
TOCIconHTMLHelp marker type	Identifies the image to use as the table of contents icon for a topic in the HTML Help output format. For more information, see "Customizing Table of Contents Icons in Word".
TOCIconJavaHelp marker type	Identifies the image to use as the table of contents icon for a topic in the Sun JavaHelp output format. For more information, see "Customizing Table of Contents Icons in Word".
TOCIconOracleHelp marker type	Identifies the image to use as the table of contents icon for a topic in the Oracle Help output format. For more information, see "Customizing Table of Contents Icons in Word".
TOCIconWWHelp marker type	Identifies the image to use as the table of contents icon for a topic in the WebWorks Help output format. For more information, see "Customizing Table of Contents Icons in Word".
TopicAlias marker type	Specifies an internal identifier for a topic that can be used to create a context-sensitive link to that topic. For more information, see "Specifying Context-Sensitive Help Links in Word".
Topic Description marker type	Specifies a topic description for a context-sensitive help topic in Eclipse help systems. For more information, see "Creating Context-Sensitive Help in Word".
WhatIsThisID marker type	Identifies a What's This help internal identifier for creating context-sensitive What's This field-level help for Microsoft HTML Help. For more information, see "Creating What's This (Field-Level) Help in Word".
WindowType marker type	Specifies the name of the window defined in the Help project that the topic should be displayed in. In Microsoft HTML Help, the window names are defined in the <code>.hhp</code> file. This marker type is supported in Microsoft HTML Help and Oracle Help. For more



Marker Type	Description
	information, see "Opening Topics in Custom Windows in Word".

## Paragraph and Character Formats in Word

ePublisher projects use the paragraph styles and character styles defined by the Stationery designer to implement online features when generating output. Before you begin using paragraph styles and character styles to implement online features, talk to the Stationery designer and verify which online features your Stationery supports. Your Stationery only recognizes the paragraph styles and character styles defined by the Stationery designer in your Stationery. If you try to implement online features using paragraph styles and character styles not supported in your Stationery, ePublisher does not recognize these items when generating output.

When the Stationery designer creates the Stationery, the Stationery designer specifies the names of paragraph styles and character styles used to implement an online feature. Consult with the Stationery designer to obtain the names of the paragraph styles and character styles defined by the Stationery designer to support each online feature you want to implement.

The following table lists the default names of paragraph styles and character styles used to implement online features. Always verify with the Stationery designer the names of the styles formats and character styles you should use when implementing online features before you use these items in your source documents.

<b>style</b>	<b>Description</b>
AbbreviationTitle character style	Specifies abbreviation alternate text for browsers to display for abbreviations such as SS# when a user hovers over the abbreviation in output. Screen readers also can read the abbreviation alternate text. Used in combination with the AbbreviationTitle marker type. For more information, see "Assigning Alternate Text to Abbreviations in Word".
AcronymTitle character style	Specifies acronym alternate text for browsers to display for acronyms such as HTML when a user hovers over the acronym in output. Screen readers can also read the acronym alternate text. Used in combination with the AcronymTitle marker type. For more information, see "Assigning Alternate Text to Acronyms in Word".
Citation character style	Specifies the source of a quote using a fully qualified Uniform Resource Identifier (URI) when a user hovers over the quote in output. Screen readers can also read the URI for the quote. Used in combination with the Citation marker type. For more information, see "Providing Citations for Quotes in Word".
Expand/Collapse paragraph style	Specifies the content you want to include in an expand/collapse section. Used in conjunction with a DropDownEnd marker type. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word".
Popup paragraph style	Specifies the popup content to display in both a popup window and in a standard help topic. Applied to the first paragraph of popup content. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word".
Popup Append paragraph style	Specifies the popup content to display in a popup window and in a standard help topic. Applied to additional popup paragraphs when you have more than one paragraph of popup content. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word".

<b>style</b>	<b>Description</b>
Popup Only paragraph style	Specifies the popup content to display in only a popup window. Applied to the first paragraph of popup content. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word".
Popup Only Append paragraph style	Specifies the popup content to display in only a popup window. Applied to additional popup paragraphs when you have more than one paragraph of popup content. For more information, see "Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word".
Related Topic paragraph style	Specifies related topics links. For more information, see "Creating Related Topics in Word".
See Also character style	Specifies the text you want to include in a See Also button. For more information, see "Creating See Also Links in Word".
See Also paragraph style	Specifies the text you want to include in a See Also inline text link. For more information, see "Creating See Also Links in Word".

## Obtaining and Applying the Latest Microsoft Word Template

An efficient, effective, and consistent ePublisher online content generation process relies upon the use of templates. Templates define paragraph, character, and table styles and standards. Templates may also contain standard variables and cross-reference definitions that you can use when creating and working with source documents used to generate online content. Templates help control the look and feel of source documents and generated output across multiple writers, multiple projects, and multiple types of generated output.

The ePublisher content generation process assumes that you use marker types and paragraph, character, and table styles defined in a Microsoft Word template prepared by a Stationery designer as you create content and format your source documents. Using Microsoft Word templates and the marker types and paragraph, character, and table styles and other layout styles and characteristics defined in templates ensures that you format content in your source documents consistently

and also ensures ePublisher can use your source documents effectively to generate output.

If your source documents do not use templates or do not use the same marker types, styles and standards defined in your Stationery by the Stationery designer, your generated output may not conform to the styles and standards defined by the Stationery designer for output. You may also not be able to implement some online features if you do not use the correct templates or the correct marker types and styles defined in the templates.

As a part of preparing your Microsoft Word source documents for output generation, ensure your source documents use the correct Microsoft Word templates from the Stationery designer and you have applied all paragraph, character, and table styles specified in the template correctly.

## Working with the WebWorks Transit Menu for Word

This section explains how to work with the WebWorks Transit menu for Microsoft Word. For more information about using the WebWorks Transit menu to prepare Microsoft Word documents for output generation, see "Implementing Online Features in Word".

### WebWorks Transit Menu for Word

The WebWorks Transit menu for Microsoft Word allows you to insert the following items into your Microsoft Word source documents:

- Markers, including filename markers and TopicAlias markers. For more information about markers in Microsoft Word, see "Custom Marker Types in Word".
- Conditions. For more information about conditions in Microsoft Word, see "Using Conditions in Word".

**Note:** Writers authoring content in non-Word formats do not use the WebWorks Transit menu for Microsoft Word.

## Installing the WebWorks Transit Menu for Word

If you have Microsoft Word installed on your computer, running the ePublisher Express installer will automatically install the WebWorks Transit plug-in.

If you did not have installed Microsoft Word before installing ePublisher Express, you can do a repair installation after installing Word.

Make sure every time you run the ePublisher Express installer, to close and save your Word documents.

## Running Transit Menu in Secure Environments

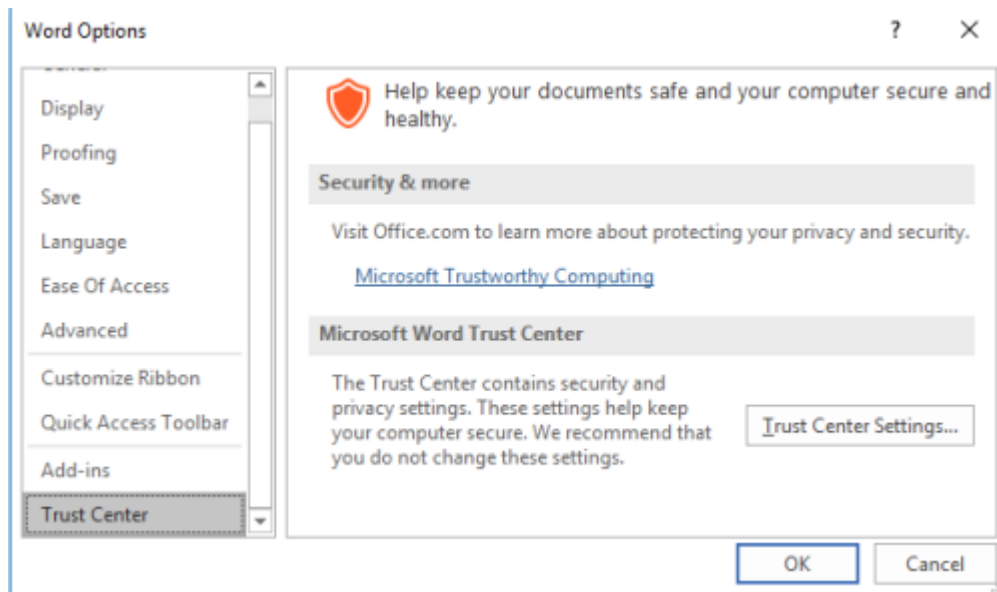
Thankfully, Microsoft eventually got serious about security. Office 2003 added a macro security level feature. By default, only macros signed with a trusted certificate could run. And guess what!!! Now Quadralay Corporation has a trusted certificate for macro documents.

Modern versions of Microsoft Office are even more restrictive. Office 2013 is set to disable all macros by default, providing a notification that the macro wasn't allowed to run. That's why after you install ePublisher Express you still need to say Microsoft Word that you do trust in us.

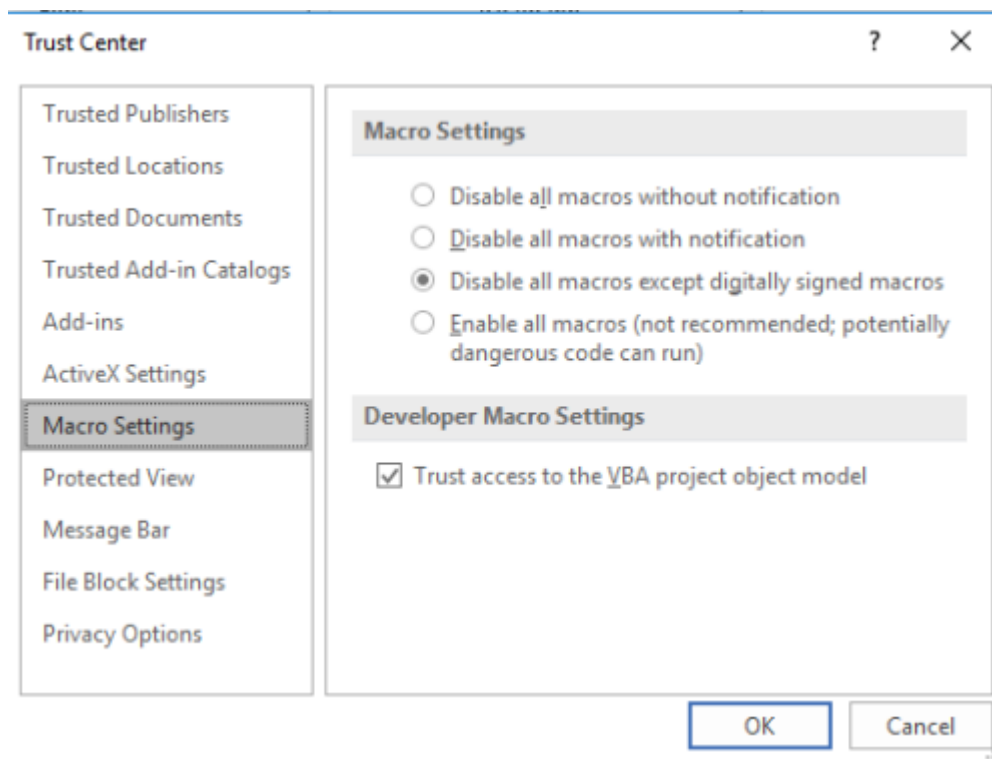
If you need to know a little bit more on how macros work and what are them, you can go to <https://www.howtogeek.com/171993/macros-explained-why-microsoft-office-files-can-be-dangerous/>.

## To run the WebWorks Transit menu for Microsoft Word in a Secure Environment

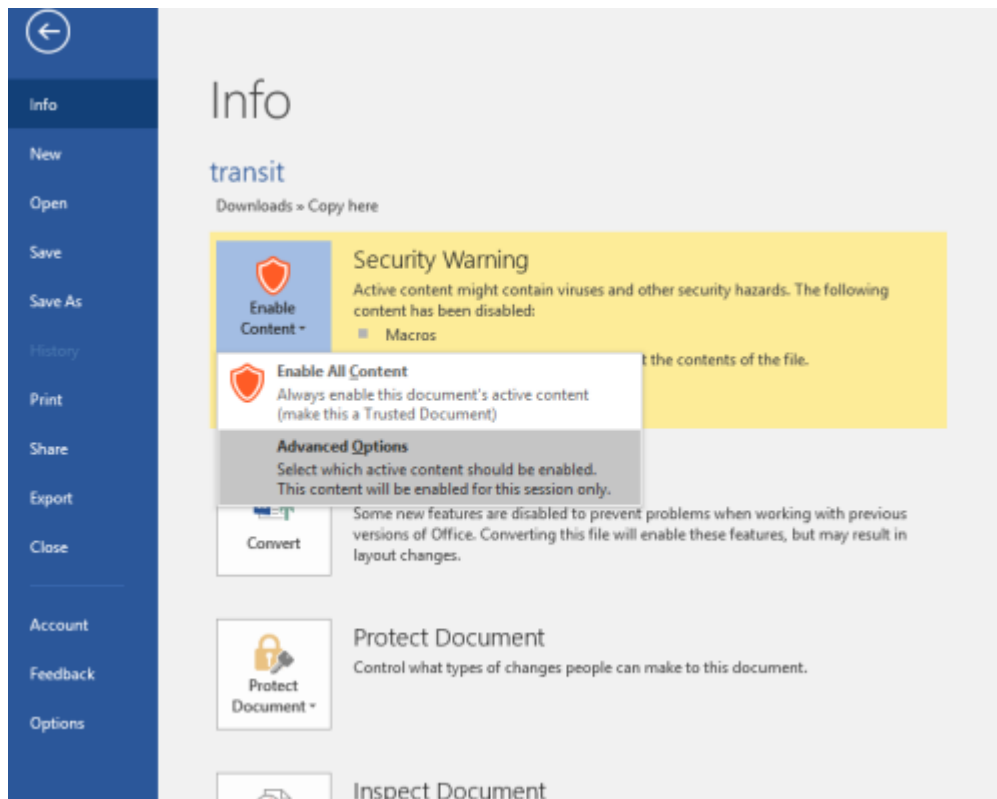
1. Open a Microsoft Word document, or even a Blank document.
2. On the **Word** menu, click **File** and then **Options**.
3. In the window that pops out in the left side menu click on **Trust Center**, and on the right side select **Trust Center Settings**. You should see a window similar to the following figure.



4. After selecting **Trust Center Settings** you'll see a new window, click on the left side on **Macro Settings** and on the right panel check **Disable all macros except digitally signed macros** and **Trust access to the VBA project object model**. You should see a window similar to the following figure.

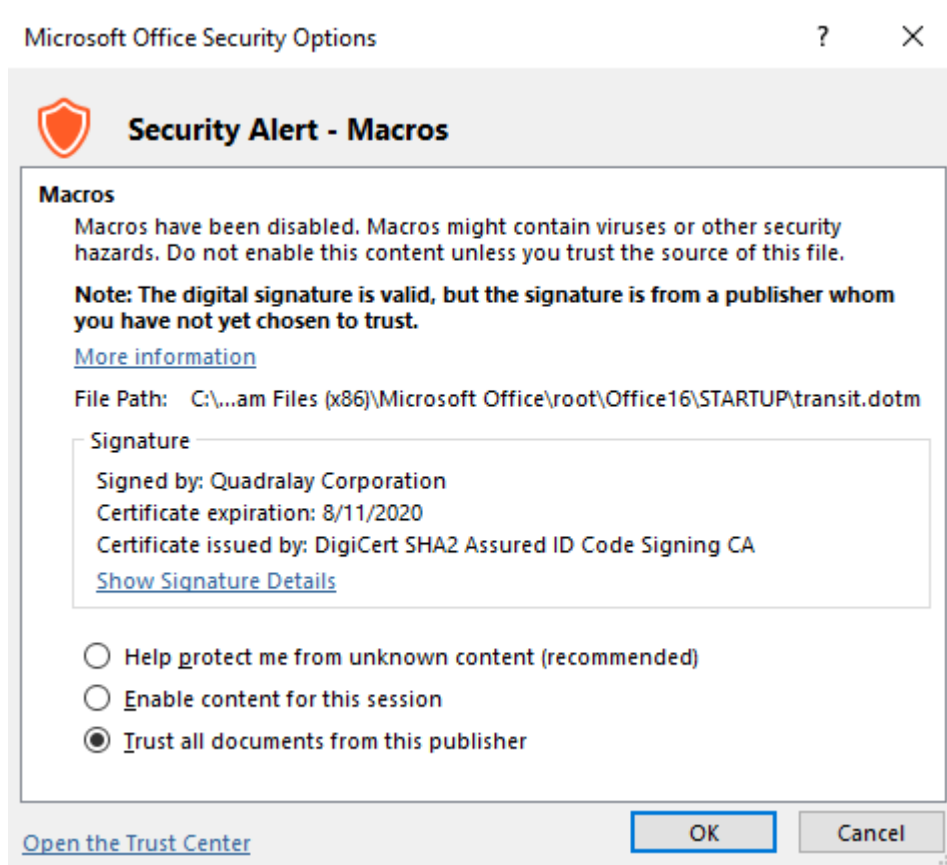


5. Then you can click OK to close all your opened windows and go again to the **Word** menu, click **File** and then **Info**. If you get the Security Warning then click on the **Enable Content** button and then **Advanced Options**. On the right side you'll see a window similar to the next one before clicking **Advanced Options**.

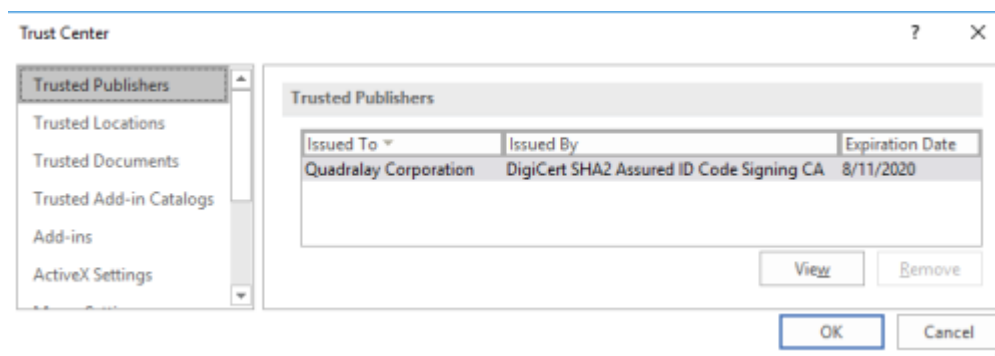


6. On the **Microsoft Office Security Options** check **Trust all documents from this publisher**. As you can see in the following figure: **The digital signature is valid, but the signature is from a publisher whom you have not yet chosen to trust.**





7. Finally you can double check going through Step 2 and 3 above, and then click on the left side **Trusted Publishers** and you'll see *Quadralay Corporation* there, as the next figure shows.



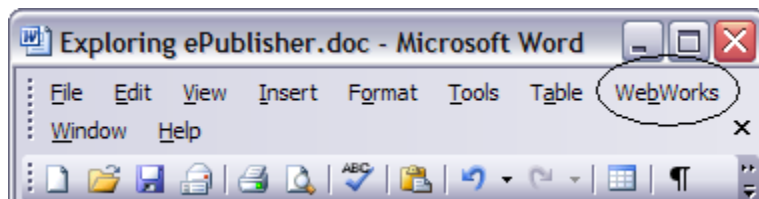
## Initializing the WebWorks Transit Menu for Microsoft Word

After you install the WebWorks Transit menu for Microsoft Word, you must initialize the WebWorks Transit menu before you can use it. For more information about

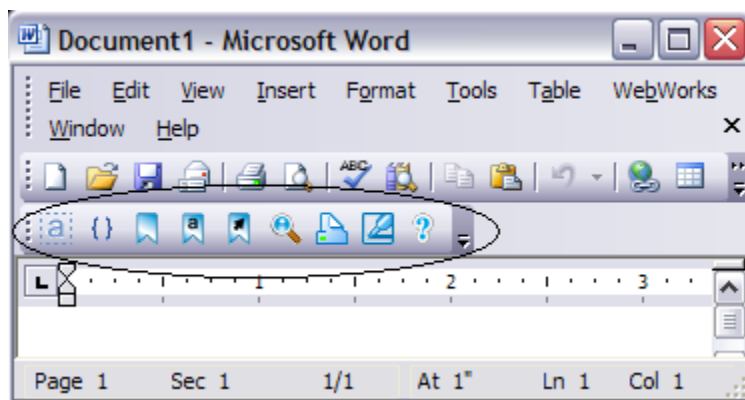
installing the WebWorks Transit Menu for Microsoft Word, see “Installing the WebWorks Transit Menu for Word”.

## To initialize the WebWorks Transit menu for Microsoft Word

1. Open Microsoft Word.
2. Verify that Microsoft Word displays the **WebWorks** menu on the Microsoft Word menu bar. Your Microsoft Word menu should be similar to the following figure.



3. *If you do not have the WebWorks Transit menu for Microsoft Word installed*, install it. For more information, see "Installing the WebWorks Transit Menu for Word".
4. *If this is the first time you are using the WebWorks Transit menu for Microsoft Word*, on the **WebWorks** menu, click **Initialize Menu**. WebWorks initializes the WebWorks Transit menu for Microsoft Word and displays the WebWorks Transit menu in the Microsoft Word window. Your Microsoft Word Window should be similar to the following figure.



## Displaying and Hiding the WebWorks Transit Menu in Word

You can choose to display or hide the WebWorks Transit menu in Microsoft Word.

For example, you may want to display the WebWorks Transit menu when you are working in Microsoft Word documents that you will use to generate output. However, you may want to hide the WebWorks Transit menu when you are working in Microsoft Word source documentation that you will not use to generate output.

## To display or hide the WebWorks Transit menu for Microsoft Word

1. Open Microsoft Word.
2. On the **WebWorks** menu, click **Preferences**.
3. *If you want to display the WebWorks Transit menu for Microsoft Word in Microsoft Word*, select the Enable Toolbar check box.
4. *If you do not want to display the WebWorks Transit menu for Microsoft Word in Microsoft Word*, clear the Enable Toolbar check box.
5. Click **OK**.

## Creating Custom Marker Types Using the WebWorks Transit Menu in Word

Typically your Stationery designer will provide the list of markers, which are a type of private or custom Microsoft Word field codes, that you can use in Microsoft Word source documents to create online features. In most cases, you should not need to create a custom marker type for use in Microsoft Word source documents. However, if you need to create a custom marker type to implement an online feature, verify with the Stationery designer that your Stationery supports the custom marker type before you create the custom marker and insert and use the custom marker in a source document.

Occasionally your Stationery may also support a custom marker type that is not defined in the WebWorks Transit menu for Microsoft Word. In this situation, first confirm with the Stationery designer that your Stationery supports the custom marker type. After confirming your project supports the custom marker type, you can create the custom marker type using the WebWorks Transit menu for Word.

## To create a custom marker type using the WebWorks Transit menu for Word

1. In your Microsoft Word source document, on the **WebWorks** menu, click **Markers**.
2. Click the plus (+) icon.
3. In the **Type** field, type *Custom*`MarkerTypeName` to create a custom marker type, where *Custom*`MarkerTypeName` is the name of the custom marker type you want to create.

**Note:** The custom marker type name you type must match the name of the custom marker type supported in your ePublisher Stationery. If you specify a name for the custom marker type that is different than the name of the custom marker type supported in your ePublisher Stationery, ePublisher will not be able to recognize and use the custom marker type when generating output.

4. Click **OK**.
5. Click **OK** again to close the window.

The WebWorks Transit menu for Word saves the custom marker you specified in the marker list and inserts the marker in your Microsoft Word source document.

## Creating a Passthrough Marker in Word

A passthrough marker is a marker that allows you to insert content that you do not want ePublisher to process when you generate output. For example, if you have embedded multimedia files in your source documents, such as Audio Video Interleave files (`.avi`) or Adobe Software Flash files (`.swf`), you can insert a passthrough marker with a value that is set to the HTML code that you do not want ePublisher to process.

The following example shows `.avi` code to which you could insert using a passthrough marker.

```
<embed src="sample.avi" width="400"
height="300" pluginspage=";>
</embed>
```

## To create a passthrough marker in a Microsoft Word source document

1. In your Microsoft Word source document, locate the paragraph in which you want to insert the passthrough marker.
2. Insert your cursor in the location on the page where you want to insert the **Passthrough** marker.
3. On the **WebWorks** menu, click **Markers**.
4. In the **Marker** field, select **Passthrough** from the list of markers.
5. In the **Value** field, type the html code that you would like to not be processed by ePublisher such as the Flash embed code indicated in the previous topic.r
6. Click **OK**. ePublisher inserts the Passthrough marker into your source document.
7. Save your Microsoft Word source document.
8. Generate output for your project. For more information, see "Generating Output".
9. In Output Explorer, verify ePublisher created the appropriate result for your embedded html code. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Working with Tables in Word

This section explains how to prepare tables in source documents for output generation. Obtain your latest templates and apply the latest table formats from the template to tables in your source documents. If your tables do not have header rows, create a header row for each table.

## Applying Table Styles in Word

Table styles define the appearance of your tables, and ePublisher uses table styles to define the appearance of tables in generated output. When you work with tables in your Microsoft Word source documents, ensure you apply the correct table styles to your tables. The Stationery designer defines the table styles you can use in your Microsoft Word source documents in the Microsoft Word templates you associate with your Microsoft Word source documents. If you want to specify a different table styles for sets of tables in your generated output, first ensure the different table styles you want to apply are available in your Microsoft Word source

document. Then apply the different table styles to tables in your Microsoft Word source documents as appropriate.

For example, you may have a small set of tables that contain information about a specific component in a product. If you decide you want to modify the appearance of these tables in your generated output by specifying that the tables associated with this component display with a yellow background in your generated output, apply a table style available in your Microsoft Word source document that the Stationery designer created to meet this requirement. When you generate output, the Stationery designed by the Stationery designer specifies that any tables created with a table style configured to display tables with a yellow background display in your output with a yellow background.

***If you are working in Microsoft Word 2000 or earlier,*** Microsoft did not provide support for named table styles in these versions of Word. You can use TableStyle markers to manually assign table style names to tables. To use the TableStyle marker, insert the TableStyle marker using the WebWorks Transit menu for Word into a cell in the table header row, then save your document, generate output, and verify in Output Explorer that ePublisher applied the table style you specified correctly.

***If you are working in Microsoft Word XP (2002) or later,*** you may use TableStyle markers to assign table styles if you prefer that approach to using Microsoft's available table style assignment controls.

**Note:** TableStyle markers take precedence over table style names derived from Word assigned table styles.

Because of the ambiguity between the different table styles present in the Microsoft Word ribbon for the versions of 2007 and beyond, a recommended approach would be to use the TableStyle marker in lieu of assigning table styles,

### To set a **TableStyle** marker in a Microsoft Word source document (instructions for Word 2007 or higher)

1. In your Microsoft Word source document, locate the table for which you want to assign a tablestyle marker
2. Insert cursor any where inside the table cells.
3. Click the **Add-Ins** tab to access the **WebWorks Transit** menu
4. Select the **Markers** option from the menu
5. In the **Configure Markers** dialog box, select the **TableStyle** marker from the list
6. In the text box for **Value:**, enter in desired name of the TableStyle
7. Click **OK**

For more information about generating output and using Output Explorer to view output files, see “Generating and Regenerating Output” and “Viewing Output in Output Explorer”.

## Creating Table Header Rows in Word

Header rows are rows that contain information that help identify the content of a particular column. If the table spans several pages of a print layout, the header row will usually repeat itself at the beginning of each new page.

When you create a table in Microsoft Word, by default Microsoft Word does not create a header row. However, if you create header rows in your Microsoft Word source documents, you can quickly and easily specify the appearance that you want for table header rows in your generated output.

**Note:** You cannot create table footer rows in Microsoft Word source documents. Microsoft Word does not support the creation of table footer rows.

The following procedure provides an example of how to create table header rows in Microsoft Word source documents using Microsoft Word 2003. Steps for creating table header rows in Microsoft Word may be different in other versions of Microsoft Word.



### To create a table header row in a Microsoft Word source document

1. In your Microsoft Word source document, locate the table for which you want to create a table header row.
2. Select the row or rows of an existing table you want to use to create the header row.
3. On the **Table** menu, click **Table Properties**.
4. On the **Row** tab, in the **Options** area, verify that the **Repeat as header row at the top of each page** check box is selected.
5. ***If the check box is not selected***, select the check box to create a header row for the table.
6. Click **OK**.

## Working with Images in Word

Many writers include images when producing documents using Microsoft Word. Most writers typically insert images into Microsoft Word source documents in one of the following ways:

- Inserting images in Microsoft Word source documents, also known as embedding images
- Inserting links to image files in the Microsoft Word source documents

If you insert an image into a Microsoft Word source document, Microsoft Word inserts, or embeds, the image in the Microsoft Word source document, and the image becomes a part of the document. Embedded images move with the text of the paragraph in the document. Embedded images in Microsoft Word are also sometimes called **inline shapes**.

If you insert a link to an image in Microsoft Word source documents, Microsoft Word inserts a link to the image and displays the image in the Microsoft Word source document. The link becomes a part of the document, but the actual image file is not inserted into the document, although the actual image file is displayed in the document. If you update the image file referenced by the link, Microsoft Word displays the updated image referenced by the link automatically. Linked images in Microsoft Word are also sometimes called shapes.

There are benefits and drawbacks to inserting images directly into Microsoft Word source documents and inserting links to images used in Microsoft Word source documents.

For example, if you insert images in Microsoft Word source documents, you do not have worry about breaking links between the Microsoft Word source documents and the image files. If you link to images in Microsoft Word source documents, you must ensure that you keep the same file structure for the image files in order to not break links between the Microsoft Word source document and the image file.

However, linking images in Microsoft Word source files, rather than inserting or embedding images, provides some of the following benefits:

- You can update image files without reinserting the image file into your Microsoft Word source documents.
- If you have one image used in multiple places, you can update the image in one place, rather than reinserting the image into multiple places.
- You can manage your documentation files and image files separately, which makes organizing images easier.
- Source documents with linked images are smaller in size than source documents with inserted, or embedded, images.

When you work with Microsoft Word source documents that you will use to generate output, ensure you follow the guidelines specified by the Stationery designer for the following items:

- Method used to insert images
- Correct DPI to use for inserted images
- Correct image file format to use for inserted images

## **Inserting Images in Word**

Before you insert images into Microsoft Word source documents you plan to use to generate output, review image considerations. For more information, see “Working with Images in Word”.

The following procedure provides an example of how to insert an image in Microsoft Word source documents using Microsoft Word 2003. Steps for inserting an image in Microsoft Word may be different in other versions of Microsoft Word.

After you insert your images, validate your images. For more information, see “Validating Images in Word”.

## To insert an image in a Microsoft Word source document

1. In your Microsoft Word source document, on the **Insert** menu, click **Picture > From File**.
2. Browse to the location of the image file you want to insert in your Microsoft Word source document, and then select the image file.
3. ***If you want to embed the image in your Microsoft Word source document***, click the **Insert** button. Microsoft Word inserts the image in your source document and displays the image.
4. ***If you want to insert a link to the image file in your Microsoft Word source document***, click the drop-down button on the **Insert** button and then click **Link to File**. Microsoft Word inserts a link to the image in your source document and displays the image in your source document.

After you insert an image, you can assign alternate text or a long description to the image. For more information, see “Assigning Alternate Text to Images and Image Maps in Word” and “Assigning Long Descriptions to Images in Word”.

## Validating Images in Word

If you inserted images in your Microsoft Word source documents, you can validate the images you inserted using the WebWorks Transit menu for Microsoft Word before you generate output. For more information about inserting images into Microsoft Word source documents, see “Inserting Images in Word”.

In the WebWorks Transit menu for Microsoft Word, embedded images are referred to as **inline shapes**, and images referenced by links in Microsoft Word source documents are referred to as **shapes**. When you validate embedded images or images referenced by links in Microsoft Word source documents, if ePublisher detects an issue with the image in your source document, ePublisher displays an error and highlights the image with the issue in your Microsoft Word source document.

## To validate images in a Microsoft Word source document:

1. Open the Microsoft Word source document that contains the images you want to validate.
2. ***If you want to validate embedded images***, complete the following steps:
  - a. On the **WebWorks** menu, click **Tools > Validate Inline Shapes**.
  - b. ePublisher displays a message that tells you how many embedded images are in the source document. Click **OK** to continue with the validation.
  - c. ***If ePublisher did not detect any issues with the embedded images***, ePublisher displays a status message that tells you all embedded images were validated successfully.
  - d. ***If ePublisher detects an issue with an embedded image***, ePublisher displays an error message. Complete one of the following steps:
    - ***If you want to fix the image with the issue***, click **No** to stop the validation scan. Go to the highlighted inline shape with the issue, fix the inline shape by re-adding the inline shape to your Microsoft Word source document, and then run the inline shape validation scan again.
    - ***If you want to continue the scan without fixing the image with the issue***, click **Yes** to continue with the validation scan.
3. ***If you want to validate images referenced by links***, complete the following steps:
  - a. On the **WebWorks** menu, click **Tools > Validate Shapes**.
  - b. ePublisher displays a message that tells you how many images referenced by links are in the source document. Click **OK** to continue with the validation.
  - c. If ePublisher did not detect any issues with the images referenced by links, ePublisher displays a status message that tells you all images were validated successfully.
  - d. If ePublisher detects an issue with an image referenced by a link, ePublisher displays an error message. Complete one of the following steps:
    - ***If you want to fix the image with the issue***, click **No** to stop the validation scan. Go to the highlighted shape with the issue,

fix the shape by re-adding the link to the shape in your Microsoft Word source document, and then run the inline shape validation scan again.

- ***If you want to continue the scan without fixing the image with the issue***, click **Yes** to continue with the validation scan.

## Creating Image Links in Word

You can create image links that allow users who click the image to link to content in another location. For example, if you include your company logo in a source document, you can define a link for the logo so that when users click the logo, they link to your company home page.

The following procedure provides an example of how to create an image link in Microsoft Word source documents using Microsoft Word 2003. Steps for creating an image link in Microsoft Word may be different in other versions of Microsoft Word.

## To create an image link in a Microsoft Word source document

1. In your Microsoft Word source document, insert the image for which you want to create an image link. For more information, see “Inserting Images in Word”.
2. Select the image for which you want to create an image link.
3. On the **Insert** menu, click **Hyperlink**.

**Note:** If Microsoft Word does not display **Hyperlink** on the **Insert** menu, you cannot use this procedure to create a hyperlinked image. However, you can create a hyperlinked text box to create a hyperlinked image. For more information, see “Creating Clickable Regions for Image Maps in Word”.

4. In the Insert Hyperlink window, select the object you want to link to and specify the appropriate options. For example, you can link to an existing file or web page, a location in a document, or an email address.
5. Click **OK**.
6. Save your Microsoft Word source document.
7. Generate output for your project. For more information, see “Generating Output”.
8. In Output Explorer, verify ePublisher created the image link using the link information you specified on the page by clicking on the image. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

## Creating Clickable Regions for Image Maps in Word

An image map can be a single image separated with clickable regions or a composite image made up of multiple images grouped together, yet still separated with clickable regions. For example, you could create an image of the countries of Europe and then define an image map for the image that allows users to link to a topic about each country when they click on an area of the image. User can click France to see information about France, Italy to see information about Italy, and so on.

When you define an image map, you can also define alternate text for each clickable region. For example, you might define alternate text for the Italy region as “Click here for more information about Italy.” For more information about assigning

alternate text to image maps, see “Assigning Alternate Text to Images and Image Maps in Word”.

## **Creating Image Maps for Single Images in Word**

You create an image map for a single image by inserting the image into a drawing canvas and then creating text boxes with hyperlinks that link to a location with additional content. You can also create image maps for composite images. For more information about creating image maps for composite images, see “Creating Image Maps for Composite Images in Word”.

The following procedure provides an example of how to create an image map for a single image in Microsoft Word source documents using Microsoft Word 2003. Steps for creating an image map for a single image in Microsoft Word may be different in other versions of Microsoft Word.

**To create an image map for a single image in a Microsoft Word source document:**

1. Insert your cursor on the line where you want to insert the single image you want to use for your image map.
2. On the **Insert** menu, click **Text Box**. Microsoft Word inserts a drawing canvas. You will insert your image and the text boxes that contain hyperlinks for each clickable area you want to specify for the image into this drawing canvas.
3. Click in the drawing canvas to insert a text box in the drawing canvas.
4. On the **Insert** menu, click **Picture > From File**.
5. Browse to the location of the image you want to use for your image map, select the image, and then click **Insert**, **Link to File**, or **Insert and Link** based on the image insertion method you use for your projects. Microsoft Word inserts the image into the drawing canvas.
6. Add a text box that covers each region in the image that you want to be able to click by completing the following steps:
  - a. Select the drawing canvas.
  - b. On the **Insert** menu, click **Text Box**.
  - c. Click on the drawing canvas, and then drag and drop the text box over an area of the image you want to make clickable.
  - d. Right-click the text box, and then click **Format Text** box.
  - e. On the **Colors and Lines** tab, in the **Fill** area, in the **Color** field, select **No Fill** from the list.
  - f. In the **Line** area, in the **Color** field, select **No Line** from the list.
  - g. Click **OK**.
7. Specify a hyperlink for each text box by completing the following steps:
  - a. Right-click the text box, and then click **Hyperlink** from the right-click menu.
  - b. Specify the location to which you want to link, and then click **OK**. For example, you can link to a Web site or you can link to a location in the source document.



8. Press and hold the **SHIFT** key, and then click the image and the text box you created for each hyperlinked area you want to use to create the clickable image map for the area.
9. When you have the image and all of the hyperlinked text boxes you created for the image map selected, continue to press and hold the **SHIFT** key, then right-click the selection and then click **Grouping > Group** on the context menu.
10. Save your Microsoft Word source document.
11. Generate output for your project. For more information, see "Generating Output".
12. In Output Explorer, verify ePublisher created the image map using the link information you specified by clicking on the page that contains the image map and then clicking on each area of the image where you created a link. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Creating Image Maps for Composite Images in Word

You can create composite images by inserting the composite images into a drawing canvas and then grouping the composite images with hyperlinked text boxes.

The following procedure provides an example of how to create image maps for composite images in Microsoft Word source documents using Microsoft Word 2003. Steps for creating image maps for composite images in Microsoft Word may be different in other versions of Microsoft Word.

## To create an image map for a composite image in a Microsoft Word source document

1. Insert your cursor on the line where you want to insert the composite image you want to use for your image map.
2. On the **Insert** menu, click **Picture > New Drawing**. Microsoft Word inserts a drawing canvas. You will insert the images that make up your composite image and the text boxes that contain hyperlinks for each clickable area you want to specify for the image into this drawing canvas.
3. Select the drawing canvas, and then on the **Insert** menu, click **Picture > From File**.
4. Browse to the location of each image that makes up your composite image, select the image, and then click **Insert, Link to File**, or **Insert and Link** based on the image insertion method you use for your projects. Microsoft Word inserts the image into the drawing canvas.
5. Position each image that makes up your composite image in the drawing canvas by dragging and dropping the image into its correct position.
6. Add a text box that covers each region in the image that you want to be able to click by completing the following steps:
  - a. Select the drawing canvas.
  - b. On the **Insert** menu, click **Text Box**.
  - c. Click on the drawing canvas, and then drag and drop the text box over an area of the image you want to make clickable.
  - d. Right-click the text box, and then click **Format Text** box.
  - e. On the **Colors and Lines** tab, in the **Fill** area, in the **Color** field, select **No Fill** from the list.
  - f. In the **Line** area, in the **Color** field, select **No Line** from the list.
  - g. Click **OK**.
7. Specify a hyperlink for each text box by completing the following steps:
  - a. Right-click the text box, and then click **Hyperlink** from the right-click menu.
  - b. Specify the location that you want to link to, and then click **OK**. For example, you can link to a Web site or you can link to a location in the source document.

8. Press and hold the **SHIFT** key, and then click each image that makes up your composite image and the text box you created for each hyperlinked area you want to use to create the clickable image map for the area.
9. When you have the image and all of the hyperlinked text boxes you created for the image map selected, continue to press and hold the **SHIFT** key, then right-click the selection and then click **Grouping > Group** on the context menu.

**Note:** After grouping the image and the text boxes, do not use the Hyperlink command on the right-click menu to assign a hyperlink to the entire group. If you do, the hyperlink you assign for the group will override the hyperlinks you assigned to the individual text boxes in the group.

10. Save your Microsoft Word source document.
11. Generate output for your project. For more information, see "Generating Output".
12. In Output Explorer, verify ePublisher created the image map using the link information you specified by clicking on the page that contains the image map and then clicking on each area of the image where you created a link. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Assigning Image Scales in Word

When ePublisher converts images inserted into your source documents, it can scale images to make them display larger or smaller in your generated output. By default, ePublisher uses the scaling factor applied to images as specified by the image style you apply to each image. For example, if you apply an image style to images and the Stationery designer defined the image style to scale images to 80% of their original size, all images that have this image style applied to them will be scaled to 80% in the generated output.

Typically, using the standard scaling factor specified in the image style is sufficient. Occasionally, however you may want to override the scaling factor for an individual image. For example, while most **.gif** images scale to 80%, you may have one large image that you want scaled to 60% in your generated output. You can manually override the standard scaling factor specified in your Stationery for a specific image by using the GraphicScale marker.

To assign a scale to a specific image, your Stationery and template must have the GraphicScale marker type configured. Your output format must also support scaling by image.

The following procedure provides an example of how to specify image scaling for an image Microsoft Word source documents using Microsoft Word 2003. Steps for

specifying image scaling for an image in Microsoft Word may be different in other versions of Microsoft Word.

## To specify an image scale for an image in a Microsoft Word source document

1. In your Microsoft Word source document, locate the image for which you want to specify image scaling.
2. Right-click the image, and then click **Format Picture** or **Format Object**.
3. Change the layout setting of the image to **Top and Bottom** by completing the following steps:

**Note:** By default when you insert images into Microsoft Word, Microsoft Word inserts the image using the **Inline with text layout** setting. In order to specify the image scale for image output files, you must group the image and the text box that contains the GraphicScale marker. However, you cannot group images using the **In line with text** layout setting in Microsoft Word. To work around this known Microsoft Word issue, if you have an image that uses an **In line with text** layout setting, use the **Top and Bottom** layout setting for the image while you insert the GraphicScale marker, and then reapply the **In line with text** layout setting after you group the image and the GraphicScale marker.

- a. On the **Layout** tab, click **Advanced**.
  - b. On the **Text Wrapping** tab, click **Top and Bottom**.
  - c. Click **OK**, and then click **OK** again to close the window.
4. Select your image.
  5. On the **Insert** menu, click **Text Box**, and then click to the right of your image. Microsoft Word inserts a text box.
  6. Insert your cursor into the text box, and then complete the following steps:
    - a. On the **WebWorks** menu, click **Markers**.
    - b. In the **Markers** field, select **GraphicScale** from the list of markers.
    - c. In the **Value** field, type a scaling value for the image.

For example, if you want the image in your Microsoft Word source document reduced by 50% when you generate output, type .

Click **OK**. ePublisher inserts the GraphicScale marker into the text box.

- d. Select the text box.
- e. Right-click the selected text box, and then click **Format Text Box**.

- f.** On the **Colors and Lines** tab, in the **Fill** area, in the **Color** field, select **No Fill**.
  - g.** In the **Line** area, in the **Color** field, select **No Line**.
  - h.** Click **OK**.
- 7.** Drag and drop the text box onto the image.
- 8.** Select the text box and the image.
- 9.** Right-click the selected text box and image, and then click **Grouping > Group**.

**Note:** When you select **Group**, the location of the image in your Microsoft Word source document may change in relation to the text in your source document. For example, the image may move up or down in your Microsoft Word source document. This is known Microsoft Word behavior. You may need to scroll up or down in your source document to the new location of the image to find the image.

- 10. If your image previously used the *In line with text layout setting for the image*, reassign this style to your image by completing the following steps:**

- a.** Right-click **only** the image, and then click **Format Object**.

**Note:** You must ensure you right-click **only** the image, and not on the text box or the grouped text box and image. If you right-click on the text box or the grouped text box and image, Microsoft Word does not display the **Format Object** menu option on the context menu.

- b.** On the **Layout** tab, click **In line with text**.
  - c.** Click **OK**, and then click **OK** again to close the window.
- 11.** Save your Microsoft Word source document.
- 12.** Generate output for your project. For more information, see "Generating Output".
- 13.** In Output Explorer, verify ePublisher created the image using the image scale you specified in the GraphicScale marker by clicking on the page that contains the image for which you specified image scaling. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

# Assigning Image Styles in Word

Typically you do not need to specify an image style for images when you generate output. By default, each image generated by ePublisher is associated with the default image style defined in the Stationery used by your Stationery. However, if you want to change the image style of one image or a small set of images, you can specify the image style you want to use for an image in your source document using the GraphicStyle marker type.

For example, if you want to specify a yellow border around a set of screen shot images that illustrate a particular piece of product functionality, you can specify that each of the screen shots images in the set have a yellow border around them through the use of the GraphicStyle marker type.

To assign a style to a specific image, your Stationery and template must have the GraphicStyle marker type configured. Your output format must also support specifying image styles.

The following procedure provides an example of how to specify image styles for images in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying image styles for images in Microsoft Word may be different in other versions of Microsoft Word.

## To specify an image style for an image in a Microsoft Word source document

1. In your Microsoft Word source document, locate the image for which you want to specify an image style.
2. Select the image, and then select the **WebWorks** menu to insert a **GraphicStyle** marker next to the image. To insert the marker follow these steps.
  - a. On the **WebWorks** menu, click **Markers**.
  - b. In the **Markers** field, select **GraphicStyle** from the list of markers.
  - c. In the **Value** field, type the name of the image style the Stationery designer configured for the Stationery used by your ePublisher project.

For example, if the Stationery designer configured an image style called GreenBorder in your Stationery, type `GreenBorder`.

Click **OK**. ePublisher inserts the GraphicStyle marker into the text box.

3. Save your Microsoft Word source document.
4. Generate output for your project. For more information, see "Generating Output".
5. In Output Explorer, verify ePublisher created the image using the image style you specified by clicking on the page that contains the image for which you specified an image style and verifying ePublisher applied the image style you specified in the generated output. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Creating Index Entries in Word

An index lists the terms and topics discussed in a document and the page or pages on which they appear. An online index provides the user with a point-and-click resource for quickly navigating online content.

ePublisher uses the same native index entry features used in source documents to create a printed index to create an online index. If you include index entries in your source documents, ePublisher detects the index entries and uses the index entries to create an online index in your generated output.

Microsoft Word inserts index entries as an XE (Index Entry) field in a field code. To create index entries in a Microsoft Word source document, insert index entries into your Microsoft Word source document. ePublisher then uses the index entries to create an online index when you generate output.



Before you insert index entries, verify with the Stationery designer that your Stationery is configured to support online index generation. By default, ePublisher enables online index generate for output, but this functionality can be disabled in your Stationery by the Stationery designer. Also confirm that your output format supports online index creation.

The following procedure provides an example of how to insert index entries in Microsoft Word source documents using Microsoft Word 2003. Steps for inserting index entries in Microsoft Word may be different in other versions of Microsoft Word.

## To insert an index entry in a Microsoft Word source document

1. In your Microsoft Word source document, select the word you want to include in your index.
2. Press **ALT+SHIFT+X**. Microsoft Word displays the selected text in the **Main entry** field on the Mark Index Entry window.
3. Specify the appropriate options for the index entry, and then click **Mark**. For more information about the options for the index entry, see the Microsoft Word Help.

Microsoft Word inserts each index entry as an XE (Index Entry) field in a field code. Field codes use hidden text format. If you don't see the XE field after you insert your index entry, click the **Paragraph** symbol on the **Standard** toolbar.

4. After you insert your index entries, update all of your inserted index entries by completing the following steps:
  - a. On the **Edit** menu, click **Select All**.
  - b. Press **F9**. Microsoft Word updates all of the field codes in the Microsoft Word source document, including the XE (Index Entry) field codes.
5. Hide the XE (Index Entries) in your source document by clicking the **Paragraph** symbol on the **Standard** toolbar to hide the index field codes and hidden text.
6. Save your Microsoft Word source document.
7. Generate output for your project. For more information, see "Generating Output".
8. In Output Explorer, verify ePublisher created the index correctly by clicking on the page or tab that displays the index and then clicking on the index entries. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer"

## Using Variables in Word

A variable serves as a placeholder for information that may change frequently. Using variables in source documents allows you to quickly and easily control the content in your generated output. When you change the value of a variable in an ePublisher project, it changes the value in only your generated output. The variable value does not change in your source document.

Once you insert variables into your source documents, whenever the value of a item defined by a variable needs to change, you can make the change in a single location, rather than searching and replacing for all instances of the item. For example, you can use variables in the following ways:

- If you have publication dates or release dates in your source documents that you need to update periodically, you can set up the date as a variable.
- If you work with products that have names or versions that frequently change, you can set up variables for product names and versions.
- If you need to produce documentation sets for a product with multiple brands, you can use variables to help you produce documentation for each different brand using the same set of source files.

## Creating Variables in Word

Microsoft Word implements variables as `DocProperty` field codes. When you work with Microsoft Word source documents, typically you use variables defined in a Microsoft Word template by a Stationery designer. The variables in these templates will also include the built-in document properties such as `Author` or `Subject`. You import these variables into your Microsoft Word source documents when you apply the template to your source documents. After you import the variables, you insert the variables as appropriate.

Typically you should not need to create variables in your Microsoft Word source files if you use a Microsoft Word template created by a Stationery designer. However, in some cases you may need to create a variable in a Microsoft Word source document if you do not have a Microsoft Word template that includes a variable you need for your project.

The following procedure provides an example of how to create variables in Microsoft Word source documents using Microsoft Word 2007 and Word 2010.

**Note:** Newer versions of Word may or may not use this exact procedure, however this may provide enough information to get working with variables in Word.

## To create a variable in Word 2007 or Word 2010

1. Go to **File**, and click **Info**
2. Click the Properties tab in the right-hand side of the window and click the **Advanced Properties** option from the dropdown
3. Click the **Custom** tab
4. Type in the Name of your variable in **Name**, for example `BookName`
5. For the **Value**, type in the information you want the variable to represent, for example *User Guide*
6. Click **Add** to add the variable to the list

## Inserting Variables into Word

You can insert a variable into a source document after you apply a Microsoft Word template that contains the variables to your source document. If you want to use a variable that is not defined in a Microsoft Word template, you must create the variable in your source document before you can insert it. For more information about creating a variable, see “Creating Variables in Word”.

The following procedure provides an example of how to insert variables in Microsoft Word source documents using Microsoft Word 2007 or 2010.

**Note:** Newer versions of Word may or may not use this exact procedure, however this may provide enough information to get working with variables in Word.

## To insert a variable (DocProperty field) into a Microsoft Word document

1. Open the Microsoft Word source document in which you want to insert a variable.
2. Place your cursor in the location where you want to insert the variable.
3. Go to the **Insert** ribbon and click on the **Quick Parts** dropdown and then select **Field...**
4. In the **Field names** selection box, select **DocProperty**
5. In the adjacent selection box labeled **Property**, select the appropriate variable name to insert in your document

## Changing Variable Values in Word

You can change the value assigned to a variable in a Microsoft Word source document.

**Note:** After you change a variable value in your source document, update the variable value for the variable from the previous value to the new value by selecting all the content in your Microsoft Word source document and then pressing the **F9** key.

The following procedure provides an example of how to change a variables in Microsoft Word source documents using Microsoft Word 2007 or 2010.

**Note:** Newer versions of Word may or may not use this exact procedure, however this may provide enough information to get working with variables in Word.

## To change a variable value in a Microsoft Word source document

1. Open the Microsoft Word source document that contains the variable with a value you want to change.
2. On the **File** menu, click **Info**.
3. Click the Properties tab in the right-hand side of the window and click the **Advanced Properties** option from the dropdown
4. Click the **Custom** tab and from the **Name** list box, select the variable you wish to change
5. In the **Value** field, type a new value for the variable. The value you type is the value that Microsoft Word displays in your Microsoft Word document.
6. Click **OK**
7. On the **Edit** menu, click **Select All**.
8. Press the **F9** key. Microsoft Word updates the variable value in each place in your source document where you inserted the variable.
9. Click **Done** again to close the window.

## Deleting Variables in Word

Delete a variable in a Microsoft Word source document when you no longer want to use the variable. Before you delete a variable, ensure you search for the variable and delete or replace all references to the variable. If your source document still contains a reference to a variable after you delete it, Microsoft Word displays errors in places where a reference to a deleted variable still exists.

The following procedure provides an example of how to delete variables in Microsoft Word source documents using Microsoft Word 2007 or 2010.

## To delete a variable in a Microsoft Word source document

1. Open the Microsoft Word source document that contains the variable you want to delete.
2. Press **Alt+F9** to display all field codes in the source document.
3. Search for and replace all references to the variable you want to delete.
4. On the **File** menu, click **Info**
5. Click **Properties** and select Advanced Properties
6. On the **Custom** tab, in the **Properties** field, select the name of the variable you want to delete in the **Name** column.
7. Click **Delete**.
8. Click **OK**.
9. On the **Edit** menu, click **Select All**.
10. Press the **F9** key. Microsoft Word updates the fields in your Microsoft Word source document. If there are any fields that reference the deleted variable in your source document, Microsoft Word displays an error message in the field.
11. Search for the word Error in your Microsoft Word source document to verify no references to the deleted variable remain in your source document.
12. Save your Microsoft Word source document.

## Using Conditions in Word

Conditions allow you to show or hide information in your source documents and in your online output. You apply conditions to the content in your source documents, and then you set the visibility for those conditions either in your source documents or in your ePublisher project.

For example, your source documents might contain some content that should be displayed in only the printed version and other content that should be displayed in only the online version. You can use the same set of source documents for both printed and online versions through the use of conditions. You can create one condition called PrintOnly specifically for printed content, and then you can create another condition called OnlineOnly specifically for online content. After you create the PrintOnly and OnlineOnly conditions, you can apply them to the appropriate content in your source documents.

Use the WebWorks Transit menu plug-in for Microsoft Word to work with conditions in your Microsoft Word source documents. ePublisher installs the WebWorks Transit menu plug-in for Microsoft Word by default when you install ePublisher Express. You also have the option to install the WebWorks Transit menu plug-in for Microsoft Word when you install ePublisher Designer or ePublisher AutoMap. For more information about installing the WebWorks Transit menu plug-in for Microsoft Word, see “Installing ePublisher Components”.

After you apply conditions in your source documents, ePublisher can use the conditions defined in your source document to control the visibility of content when it generates output. You can also change the visibility specified for any condition in your ePublisher project. Changing the visibility specified for any condition in your ePublisher project does not change the visibility specified for the condition in your source documents.

## **Creating Conditions in Word**

The WebWorks Transit menu for Microsoft Word allows you to quickly and easily create conditions you can then use to control content in your source documents. Obtain a list of supported conditions from the Stationery designer, and then create each supported condition in each of your Microsoft Word source documents using the WebWorks Transit menu for Microsoft Word.

The following procedure provides an example of how to create conditions in Microsoft Word source documents using Microsoft Word 2003. Steps for creating conditions in Microsoft Word may be different in other versions of Microsoft Word.



## To create a condition in a Microsoft Word source document

1. Open Microsoft Word.
2. Ensure that the WebWorks Transit Menu for Microsoft Word is installed on your computer and initialized. For more information, see “Working with the WebWorks Transit Menu for Word”.
3. In your Microsoft Word source document, on the **WebWorks** menu, click **Conditions**.
4. Click the **Add** icon.
5. In the **Type** field, type a name for the condition.

For example, if you want to create a condition for content that you want to display in only online content, type `OnlineOnly`. If you want to create a condition for content that you want to display in only printed content, type `PrintOnly`.

6. *If you want the content the condition is applied to hidden in Microsoft Word*, select the **Hidden** check box.
7. *If you want to highlight the content the condition is applied to in Microsoft Word*, in the **Highlight** field, select a color from the drop-down list. Highlighting the content the condition is applied to allows you to more easily see the conditionalized content in your Microsoft Word source documents.
8. Click **OK**.
9. Click **OK** again.

## Applying Conditions in Word

After you have created conditions in your Microsoft Word source documents, you can apply conditions to content. For more information about creating conditions in Microsoft Word source documents, see “Creating Conditions in Word”.

The following procedure provides an example of how to apply conditions in Microsoft Word source documents using Microsoft Word 2003. Steps for applying conditions in Microsoft Word may be different in other versions of Microsoft Word.

## To apply a condition to content in a Microsoft Word source document

1. In your Microsoft Word source document, select the content to which you want to apply the condition.
2. On the **WebWorks** menu, click **Conditions**.
3. Select a condition.
4. Click **Apply Condition**.
5. Click **OK**.

## Validating Conditions in Word

An unbalanced condition is a condition that does not have either an opening or closing tag. You may accidentally create an unbalanced condition if you delete an opening or closing tag.

The following is an example of a balanced condition:

```
{PRIVATE WWMTS PrintOnly} Timing Devices {PRIVATE WWMTE PrintOnly}
```

The following is an example of an unbalanced condition:

```
{PRIVATE WWMTS PrintOnly} Timing Devices
```

If you have any unbalanced conditions in your Microsoft Word source documents, ePublisher cannot apply the condition when it generates output.

If you use conditions in your Microsoft Word source documents, validate your conditions and verify that your conditions are balanced before you generate output. When you validate conditions, if you have unbalanced conditions in your Microsoft Word source document ePublisher displays the following error.



If ePublisher displays the error, you can go either go to the location of the error, fix the unbalanced condition in your source document, and then continue the validation, or you can cancel the validation.

## To validate conditions in a Microsoft Word source document

1. In your Microsoft Word source document, on the **WebWorks** menu, click **Tools > Validate Conditions**. ePublisher scans the Microsoft Word source document for unbalanced conditions.
2. *If the validation scan detects an unbalanced condition*, click **Navigate to error** to go to the unbalanced condition and correct the error.
3. *If you want the validation scan to continue without correcting the unbalanced condition*, click **Continue scan**.
4. *If you want to cancel the validation scan*, click **Cancel scan**.

## Removing Conditions in Word

If you no longer want to apply a condition to content in a Microsoft Word source document, you can remove the applied condition from the content.

The following procedure provides an example of how to remove conditions from content in Microsoft Word source documents using Microsoft Word 2003. Steps for removing conditions from content in Microsoft Word may be different in other versions of Microsoft Word.

### To remove a condition from content in a Microsoft Word source document

1. In your Microsoft Word source document, select the content with the condition you want to remove.
2. On the **WebWorks** menu, click **Conditions**.
3. Click the **Delete** icon.
4. *If you want to remove the condition from the content but keep the content in your Microsoft Word source document*, click **OK**.
5. *If you want to remove both the condition from the content and delete the content from your Microsoft Word source document*, select the **Delete applied content** check box, and then click **OK**.
6. Click **OK** again.

## Modifying Conditions in Word

You can edit the name of the condition, specify whether you want the content to which you applied the condition hidden or displayed in Microsoft Word, and change the color assigned to a condition.

The following procedure provides an example of how to modify conditions in Microsoft Word source documents using Microsoft Word 2003. Steps for modifying conditions in Microsoft Word may be different in other versions of Microsoft Word.

## To modify a condition in a Microsoft Word source document

1. In your Microsoft Word source document, on the **WebWorks** menu, click **Conditions**.
2. Select the condition you want to modify.
3. Click the **Edit** icon.
4. *If you want to change the name of the condition*, in the **Type** field, type a new name for the condition.
5. *If you want the content the condition is applied to hidden in Microsoft Word*, select the **Hidden** check box.
6. *If you want the content the condition is applied to displayed in Microsoft Word*, clear the **Hidden** check box.
7. *If you want to change the color used to highlight the content to which the condition is applied*, in the **Highlight** field, select a color from the drop-down list. Specifying a color for the condition allows you to more easily see the content the condition is applied to in your Microsoft Word source document.
8. Click **OK**.
9. Click **OK** again.

## Highlighting All Conditions in Word

You can use WebWorks Transit menu functionality to highlight conditions you applied in your Microsoft Word source document. Highlighting all of the conditions you applied in your Microsoft Word source document allows you to see where all of the conditional content is in your Microsoft Word source document.

The following procedure provides an example of how to highlight all conditions in Microsoft Word source documents using Microsoft Word 2003. Steps for highlighting all conditions in Microsoft Word may be different in other versions of Microsoft Word.

### To highlight all conditions in a Microsoft Word source document

1. In your Microsoft Word source document, on the **WebWorks** menu, click **Preferences**.
2. Select the **Show highlighting** check box.
3. Click **OK**.

## Displaying Conditionalized Content with Conflicting Settings in Word

You can apply more than one condition to content in your Microsoft Word source documents. If you apply more than one condition to content in your Microsoft Word source document and the conditions that you applied to the content have different show and hide settings, you can specify how you want conditionalized content with conflicting show and hide settings displayed in your Microsoft Word source documents.

For example, you may have content with three conditions applied to it. Two of the conditions applied may be set to show, or display, in the Microsoft Word source document, while one of the conditions may be set to hide, or not display, in the Microsoft Word source document.

If you have multiple conditions applied to content in your source documents with conflicting show and hide settings, you can choose if you want to display the content with conflicting conditions in a Microsoft Word source document or hide the content.

The following procedure provides an example of how to display conditionalized content with conflicting show and hide settings in Microsoft Word source documents using Microsoft Word 2003. Steps for displaying conditionalized content with conflicting show and hide settings in Microsoft Word may be different in other versions of Microsoft Word.

## To display conditionalized content with conflicting show and hide settings in a Microsoft Word source document

1. In your Microsoft Word source document, on the **WebWorks** menu, click **Preferences**.
2. *If you want to show content that has conditions applied with conflicting show and hide settings*, select the **Give priority to show conditions** check box. This check box is selected by default.
3. *If you want to hide content that has conditions applied with conflicting show and hide settings*, clear the **Give priority to show conditions** check box.
4. Click **OK**.

## Using Passthrough Conditions in Word

A passthrough condition is a condition you apply to content that you do not want ePublisher to process when you generate output. For example, if you have embedded multimedia files in your source documents, such as Audio Video Interleave files (`.avi`) or Adobe Software Flash files (`.swf`), you can apply a passthrough condition to the code so that ePublisher does not process the code.

The following example shows `.avi` code to which you can apply a passthrough condition.

```
<embed src="sample.avi" width="400"
height="300" pluginspage="";>
</embed>
```

The following example shows `.swf` code to which you can apply a passthrough condition.

```
<embed src="sample.swf" width="400"
height="300" pluginspage="
http://www.macromedia.com/shockwave/download/index.cgi?
P1_Prod_Version=ShockwaveFlash";>
</embed>
```

If you have code in your Microsoft Word source documents that you do not want ePublisher to process, create a passthrough condition and then apply the passthrough condition to the code. For more information, see "Creating Conditions in Word" and "Applying Conditions in Word".

You can also use Passthrough markers and the Passthrough paragraph styles and character styles options to insert content directly into your output without being transformed and coded for your output.



# Deleting Conditions in Word

Delete a condition in a Microsoft Word source document when you no longer want to apply the condition to content in the source document.

The following procedure provides an example of how to delete conditions in Microsoft Word source documents using Microsoft Word 2003. Steps for deleting conditions in Microsoft Word may be different in other versions of Microsoft Word.

## To delete a condition in a Microsoft Word source document

1. In your Microsoft Word source document, on the **WebWorks** menu, click **Conditions**.
2. Select the condition you want to delete.
3. Click the **Delete condition** icon.
4. *If you want to delete the condition from the list of available conditions and remove the condition from any content to which it was applied in the source document*, click **OK**. ePublisher removes the condition from the list of conditions and removes the condition from any content in the source document to which you applied the condition.
5. *If you want to delete the condition from the list of available conditions and also delete any content to which the condition was applied*, select the **Delete applied content** check box, and then click **OK**. ePublisher removes the condition from the list of conditions and deletes any content to which the condition was applied in the source document.

## Specifying Output File Names in Word

By default, ePublisher automatically assigns file names to your generated output files for topics (pages) and for embedded images (graphics).

**Note:** If you insert your images using the **Link to File** or **Insert and Link** option in the Insert Picture window in Microsoft Word, ePublisher preserves the original file names. For more information, see “Working with Images in Word”.

You can customize this naming convention using one of the following methods:

- Inserting Filename markers into your source documents
- Specifying the topic (page) and image (graphic) naming patterns for ePublisher to use in the target settings for your output

This section explains how you can specify page and image output file names in your Word source documents using Filename markers. For more information about using target settings to specify output file names using page and image naming patterns, see “Specifying Page, Image, and Table File Naming Patterns”.

## Specifying Page Output File Names in Word

Specify specific names for page output files when you generate output using Filename markers. Insert Filename markers into your source document for each page you want to specify a file name for when you generate content.

**Note:** You can also use page naming patterns to specify names for page output files and embedded image output files. For more information, see “Specifying Page, Image, and Table File Naming Patterns”.

To specify a file name for a page output file, your Stationery and template must have the Filename marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to specify page output file names in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying page output file names in Microsoft Word may be different in other versions of Microsoft Word.

## To specify page output file names in a Microsoft Word source document

1. In your Microsoft Word source document, locate the page for the topic to which you want to assign a specific file name. For more information about creating pages using page breaks, see “Specifying Page Breaks Settings”.
2. Insert your cursor at the beginning of the first heading on the page.
3. On the **WebWorks** menu, click **Insert Filename Marker**.
4. In the **Filename** field, complete the following steps:
  - a. Type the file name you want to specify for the output page file. Do not include the output file extension when you type the file name text.
  - b. Click **OK**. ePublisher inserts the Filename marker into your Microsoft Word source document.
5. Save your Microsoft Word source document.
6. Generate output for your project. For more information, see “Generating Output”.
7. In Output Explorer, verify ePublisher created an output file using the file name you specified. For more information, see “Viewing Output in Output Explorer”.

## Specifying Image Output File Names in Word

Specify specific names for image output files when you generate output if you embed, or insert images directly into the source document instead of inserting and linking or linking images.

Many writers do not need use Filename markers to specify image output file names because many writers prefer to insert images in Microsoft Word source documents as references, or links, using the **Link to File** or **Insert and Link** option in the Insert Picture window in Microsoft Word. When writers use one of these methods to insert images, ePublisher automatically uses the name of the image file referenced by the link as the name of the image output file when generating image output files from Microsoft Word source documents.

However, some writers prefer to insert images directly into Microsoft Word source documents using the **Insert** option in Microsoft Word. If you use the **Insert** option, Microsoft Word inserts the image directly into the Microsoft Word source document. When you use the **Insert** option to insert images directly into Microsoft Word source documents, by default, ePublisher assigns image output file names for the inserted images using an image naming pattern. For more information about image naming patterns, see “Specifying Page, Image, and Table File Naming Patterns”.

However, if you use the **Insert** option to insert images directly into Microsoft Word source documents, you can also use Filename markers to specify image output file names. Insert Filename markers into your source document for each image you want to specify a file name for when you generate content. To specify a file name for an image output file, your Stationery must have the Filename marker type configured. Your output format must also support this feature.

When you specify image output file names, you create a text box, insert a Filename marker in the text box, and then group the image and the text box that contains the Filename marker. The following procedure provides an example of how to specify image output file names in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying image output file names in Microsoft Word may be different in other versions of Microsoft Word.

## To specify image output file names for inserted images in a Microsoft Word source document

1. In your Microsoft Word source document, locate the inserted image for which you want to assign an output image file name.

**Note:** Only perform this procedure if you have inserted the image directory into the file using the **Insert** option when you inserted the image into your Microsoft Word source document. Do not perform this procedure if you have inserted the image using the **Link to File** or **Insert and Link** options when you inserted the image into your Microsoft Word source document.

2. Right-click the image, and then select **Format Picture** or **Format Object**.
3. Change the layout setting of the image to **Top and Bottom** by completing the following steps:

**Note:** By default when you insert images into Microsoft Word, Microsoft Word inserts the image using the **Inline with text** layout setting. In order to specify an image output file name for an inserted image, you must group the image and the text box that contains the Filename marker. However, you cannot group images using the **In line with text** layout setting in Microsoft Word. To work around this known Microsoft Word issue, if you have an image that uses an **In line with text** layout setting, use the **Top and Bottom** layout setting for the image while you insert the Filename marker, and then reapply the **Inline with text** layout setting after you group the image and the Filename marker.

- a. On the **Layout** tab, click **Advanced**.
  - b. On the **Text Wrapping** tab, click **Top and Bottom**.
  - c. Click **OK**, and then click **OK** again to close the window.
4. Select the image.
  5. On the **Insert** menu, click **Text Box**, and then click to the right of your image. Microsoft Word inserts a text box.
  6. Insert your cursor into the text box, and then complete the following steps:
    - a. On the **WebWorks** menu, click **Insert Filename Marker**.
    - b. In the **Filename** field, type the file name you want to specify for the output image file, and then click **OK**. ePublisher inserts a Filename marker into the text box.
    - c. Select the text box.

- d. Right-click the selected text box, and then click **Format Text Box**.
  - e. On the **Colors and Lines** tab, the **Fill** area, in the **Color** field, select **No Fill**.
  - f. In the **Line** area, in the **Color** field, select **No Line**.
  - g. Click **OK**.
- 7. Drag and drop the text box onto the image.
- 8. Select the text box and the image.
- 9. Right-click the selected text box and image, and then click **Grouping > Group**.

**Note:** When you select **Group**, the location of the image in your Microsoft Word source document may change in relation to the text in your source document. For example, the image may move up or down in your Microsoft Word source document. This is known Microsoft Word behavior. You may need to scroll up or down in your source document to the new location of the image to find the image.

- 10. If your image previously used the **In line with text** layout setting for the image, reassign this style to the image by completing the following steps:

- a. Right-click only the image, and then click **Format Object**.

**Note:** You must ensure you right-click only the image, and not on the text box or the grouped text box and image. If you right-click on the text box or the grouped text box and image, Microsoft Word does not display the **Format Object** menu option on the context menu.

- b. On the **Layout** tab, click **In line with text**.
    - c. Click **OK**, and then click **OK** again to close the window.
- 11. Save your Microsoft Word source document.
- 12. Generate output for your project. For more information, see "Generating Output".
- 13. In Output Explorer, verify ePublisher created an image output file using the file name you specified in the Filename marker. For more information, see "Viewing Output in Output Explorer".

# Creating Context-Sensitive Help in Word

This section explains how you can use ePublisher to create links to context-sensitive help content in Microsoft Word source documents

## Context-Sensitive Help in Word

Context-sensitive help links provide content based on the context of what the user is doing. In many cases, this help content is based on the window that is open and active. For example, the Help button on a window in a software product can open a specific Help topic that provides important information about the window:

- What the window allows you to do
- Brief concepts needed to understand the window
- Guidance for how to use the window
- Descriptions about each field on the window, valid values, and related fields
- Links to related topics, such as concepts and tasks related to the window

The Help topic can also be embedded in the window itself, such as an HTML pane that displays the content of the Help topic. Providing this content when and where the user needs it, without requiring the user to search through the help, keeps the user productive and focused. This type of help also makes the product more intuitive by providing answers when and where needed.

There are several methods for creating context-sensitive Help. In addition, output formats use different mechanisms to support context-sensitive Help. You can reference a topic in the following ways:

### File name

Use a Filename marker to assign a file name to a topic. Each topic can have no more than one Filename marker by default. However, you can create a custom mapping mechanism using file names. Then, you can open the specific topic with that file name. However, if your file naming changes, you need to change the link to the topic. This file naming approach delivers context-sensitive help capabilities in output formats that do not provide a mapping mechanism.

### Internal identifier (topic alias)

Use a TopicAlias marker to define an internal identifier for each topic. The benefit of using an internal identifier is that it allows file names to change



without impacting the links from the product. The writer inserts this marker in a topic and specifies a unique value for that topic. Then, the mapping mechanism of your output format determines how that internal identifier is supported. Some output formats, such as HTML Help, use a mapping file that defines these topic aliases.

To simplify the coding of your source documents, the Stationery designer can also configure your Stationery to define both the file name and the topic alias for each topic file.

Before you begin to insert Filename markers or TopicAlias markers into your source documents, consult with your Stationery designer. Confirm that your Stationery supports context-sensitive help links, and discuss with your Stationery designer the type of marker you should use to define context-sensitive help link in your source documents.

For more information about configuring Filename and TopicAlias markers for context-sensitive help links, see the following topics:

- “Defining Context-Sensitive Help Links”
- “Defining Filename Markers for Context-Sensitive Help Links”

“Defining Filename Markers for Context-Sensitive Help Links” ***If you generate Eclipse Help output***, you also can choose the topic description you want to display for each context-sensitive link. When you use a TopicAlias marker to create context-sensitive links, Eclipse creates a `contexts.xml` file that lists all of the context IDs for the Eclipse Help system you created using TopicAlias markers. In the `contexts.xml` file, Eclipse also provides a description of the context-sensitive link. By default, the description Eclipse provides for the context-sensitive link is the text of the first paragraph of the topic. However, if you want to specify a different description for the context-sensitive link, you can do this by using the TopicDescription marker. For more information about using the TopicDescription marker, see “Specifying Context-Sensitive Help Links in Word”.

## Planning for Context-Sensitive Help in Word

Creating context-sensitive help requires you to collaborate with application developers. Because topic IDs and map numbers must be embedded in both the software application and in your source documents, you and the application developers must agree in advance on the values to use.

Before you create context-sensitive help topics, first confirm with your application developers that the application supports context-sensitive help. Then work with your application developers to decide how to choose the topic ID for each context-sensitive help topic:

### You choose the topic IDs

You can choose a set of topic IDs and embed them in your source documents using TopicAlias markers. When you generate output, ePublisher can generate a mapping file using those topic IDs and assign a unique number to each topic ID. You can provide the generated mapping file to your application developers, who can embed the topic IDs in the application code. You can then manually maintain this mapping file, or you can allow ePublisher to generate a new file each time you generate the help. Remember to give the updated help system and mapping file to your application developers each time.

### **Your developers choose the topic IDs**

Your application developers can choose a set of topic IDs and embed them in the application code. Then, you can get a copy of the mapping file from your application developers, specify this mapping file in your project settings, and embed the topic IDs in your source documents using TopicAlias markers. In this case, ePublisher does not generate the mapping file.

Before you begin to implement context-sensitive help, meet with your application developers to select one of these methods for assigning the topic IDs to use for context-sensitive help links. Once you choose a set of topic IDs, embed them in your source documents using TopicAlias markers and do not change them.

**Note:** Because of the way Microsoft Word uses markers with the Transit menu, in order for ePublisher to best pickup a marker such as a TopicAlias, please place this marker after the heading and not before.

## **Specifying Context-Sensitive Help Links in Word**

You can use TopicAlias markers that contain topic IDs, or Filename markers that specify file names, to create context-sensitive help. If your output format supports the use of mapping files and topic IDs, typically you use TopicAlias markers to create context-sensitive help. If your output format does not support the use of mapping files and topic IDs, typically you use Filename markers to create context-sensitive help.

**If you are generating Eclipse Help**, you can also choose to specify a topic description for each context-sensitive help link you created using a TopicAlias marker by using a TopicDescription marker in conjunction with the TopicAlias marker. For more information about how TopicAlias markers and TopicDescription markers can work together when generating Eclipse Help, see “Context-Sensitive Help in Word”.

To specify a context-sensitive help link, your Stationery and template must have a TopicAlias or Filename marker type configured. If you are generating Eclipse Help and you want to be able to specify topic descriptions for your context-sensitive help links, your Stationery and template must also have a TopicDescription marker type

configured. Consult with the Stationery designer to determine which marker type you should use to create context-sensitive help links and topic descriptions in your source documents. Your output format must also support this feature.

The following procedure provides an example of how to create context-sensitive help links and topic descriptions in Microsoft Word source documents using Microsoft Word 2003. Steps for creating context-sensitive help links in Microsoft Word may be different in other versions of Microsoft Word.

## To create a context-sensitive help link in a Microsoft Word source document

1. Open the Microsoft Word source document that contains the context-sensitive topic you want to link to when users click a help button or help icon from within an application.
2. Insert your cursor at the end of the heading paragraph (or body paragraph if no heading) to which you want to link.
3. On the **WebWorks** menu, click **Markers**.
4. Select the marker type the Stationery designer configured your Stationery to support from the drop-down list. For example, select **TopicAlias** or **Filename**.
5. In the **Value** field, type the topic ID you want to specify for the topic.
6. Click **OK**.
7. ***If you are generating Eclipse Help and you want to specify topic descriptions for each context-sensitive help link you are creating,*** complete the following steps:
  - a. Insert your cursor in the topic after the TopicAlias marker you inserted for the Eclipse context-sensitive help topic.
  - b. On the **WebWorks** menu, click **Markers**.
  - c. Select the **TopicDescription** marker type from the list.
  - d. ***If the TopicDescription marker type is not on the list,*** check with the Stationery designer to obtain the name of the marker type the Stationery designer created to support this functionality, and then use the marker type specified by the Stationery designer. For more information, refer to "Implementing Online Features in Word".
  - e. In the **Value** field, type the topic description you want to use.
  - f. Click **OK**.
8. Save your Microsoft Word source document.
9. Generate output for your project. For more information, see "Generating Output".
10. In Output Explorer, complete the following steps:

- a. Verify that ePublisher inserted the topic ID into the map file when it generated output.
- b. ***If you generated Eclipse Help and specified topic descriptions for your context-sensitive help topics***, verify that the `contents.xml` file for your Eclipse Help system contains the topic descriptions you specified for context-sensitive help topics.
- c. Test the generated output using the application and verify that the application links to the appropriate context-sensitive help topic. This testing ensures the context-sensitive help link you created displays correctly within the application.

## Creating Popup Windows in Word

A popup window is a window that is smaller than standard windows and typically does not contain some of the standard window features such as tool bars or status bars. Popup windows display when users hover over or click on a link. The popup window closes automatically as soon as the users click somewhere else.

A typical use of popup windows is to display glossary terms. For example, in printed documentation, terms and definitions are typically grouped in a separate glossary document. However, in online content, you can display glossary definitions in popup windows. With glossary popup windows, users can choose whether or not they want to view the definition of a term.

You create popup windows by creating a link between the word or phrase in a topic and the content you want to display in the popup window. After you create the link, you then insert Popup markers or apply Popup paragraph styles to define the content you want to display in the popup window.

If the Stationery designer configured the Stationery to support popup windows using markers, you use the following Popup markers to create popup windows:

### **Popup**

Specifies the start of the content to include in a popup window. The content displays in a popup window when users hover over or click on the link. In some output formats users can also view the content in a standard help topic window in addition to viewing the content in a popup window. For example, if you insert a Popup marker in front of a glossary definition, the glossary definition displays in both a popup window and in a glossary topic that contains the definition.

### **PopupEnd**

Specifies the end of the content to display in the popup window.

## **PopupOnly**

Specifies that the popup content displays only through a popup window. For example, if you insert a PopupOnly marker in front of a glossary definition, the glossary definition displays only in a popup window.

If the Stationery designer configured the Stationery to support popup windows using paragraph styles, you use the following paragraph styles to create popup windows:

### **Popup and Popup Append paragraph styles**

Specifies that content displays both in popup windows and in standard help topics. You apply the Popup paragraph style to the first paragraph of content you want displayed in the popup window. If you have more than one paragraph of content you want to display, you apply the Popup Append style to the additional paragraphs.

For example, if you apply a glossary term and glossary definitions style for a glossary using the Popup and Popup Append styles, the terms and definitions in your output display in both a popup window and in a glossary topic that contains the definitions.

### **Popup Only and Popup Only Append paragraph styles**

Specifies that content displays only in popup windows. You apply the Popup Only paragraph style to the first paragraph of content you want displayed in the popup window. If you have more than one paragraph of content you want to display, you apply the Popup Only Append style to the additional paragraphs.

For example, if you apply a glossary term and glossary definition style for a glossary using the Popup Only and Popup Only Append paragraph styles, the terms and definitions in your output display in only popup windows. The content is not displayed in an additional glossary topic that contains the definitions.

## **Creating Popup Window Links in Word**

Your first step in creating a popup window is to create a link between a word or phrase in a topic and the popup content you want to display when users hover over or click the link. Use native Microsoft Word functionality to create a link between the word or phrase in a topic and the content you want to display in a popup window. You can create a link in Microsoft Word with a bookmark and a cross-reference or hyperlink.

Before you create popup window links, verify that your output format supports this feature.

The following procedure provides an example of how to create a popup window link in Microsoft Word source documents using Microsoft Word 2003. Steps for creating popup window links in Microsoft Word may be different in other versions of Microsoft Word.

## To create a link between a word or phrase and popup content in a Microsoft Word source document

1. In your Microsoft Word source document, locate the text you want to create a link to and display in the popup window.
2. ***If you want to create a link that includes the link target text***, create the link using a bookmark and a cross-reference by completing the following steps:

- a. Select the text to which you want to link.
- b. On the **Insert** menu, click **Bookmark**.
- c. In the **Bookmark name** field, type a name for the bookmark in CamelCase. The bookmark name cannot include spaces.

For example, if you are creating a bookmark for the definition of WebWorks Help in your source document, type `WebWorksHelpDefinition`.

- d. Click **Add**. Microsoft Word inserts a hidden bookmark.
- e. In your Microsoft Word source document, locate the word or phrase for which you want to create a link.
- f. Using your cursor, select the text you for which you want to create a link.

For example, if you want to specify WebWorks Help as a link, select **WebWorks Help**.

- g. On the **Insert** menu, click **Reference** > **Cross-reference**.
- h. In the **Reference type** field, select **Bookmark**.
- i. In the **Insert reference to** field, select **Bookmark text**.
- j. Select the **Insert as hyperlink** check box.
- k. In the **For which bookmark** field, click the name of the bookmark for the text you want to display in the popup.

For example, if you created a bookmark named WebWorksHelpDefinition for text that provides a definition for WebWorks Help, click **WebWorksHelpDefinition**.

- l. Click **Insert**, and then click **Close**.



**3. If you want to create a link that does not include the link target text,** create the link using a bookmark and a hyperlink by completing the following steps:

- a.** Insert your cursor in front of the text to which you want to link.
- b.** On the **Insert** menu, click **Bookmark**.
- c.** In the **Bookmark name** field, type a name for the bookmark in CamelCase. The bookmark name cannot include spaces.

For example, if you are creating a bookmark for the definition of WebWorks Help in your source document, type WebWorksHelpDefinition.

- d.** Click **Add**. Microsoft Word inserts a hidden bookmark.
- e.** In your Microsoft Word source document, locate the word or phrase for which you want to create a link.
- f.** Using your cursor, select the text you for which you want to create a link.

For example, if you want to specify WebWorks Help as a link, select **WebWorks Help**.

- g.** On the **Insert** menu, click **Hyperlink**.
- h.** In the **Link to** area, click **Place in This Document**.
- i.** In the **Select a place in this document** field, under **Bookmarks**, click the name of the bookmark for the text you want to display in the popup.

For example, if you created a bookmark named WebWorksHelpDefinition for text that provides a definition for WebWorks Help, click **WebWorksHelpDefinition**.

- j.** Click **OK**.

**4.** Verify that the link goes to the appropriate location in the source document by pressing and holding down the **CTRL** key and then clicking the link.

**5.** Save your Microsoft Word source document.

After you create a link between a word or phrase in a topic and the popup content you want to display in the popup window, define the content you want to display in the popup window using one of the following methods:

- Create popup windows using Popup markers. For more information, see “Using Markers to Create Popup Windows in Word”.
- Create popup windows using Popup paragraph styles. For more information, see “Using Paragraph Styles to Create Popup Windows in Word”.

## Using Markers to Create Popup Windows in Word

You can insert Popup markers into your Microsoft Word source documents to create popup windows. To use Popup markers to create popup windows, your Stationery must have the following items configured:

- Popup marker type
- PopupEnd marker type
- PopupOnly marker type

Your output format must also support this feature.

The following procedure provides an example of how to insert Popup markers in Microsoft Word source documents using Microsoft Word 2003. Steps for inserting Popup markers in Microsoft Word may be different in other versions of Microsoft Word.

**Note:** Popup content is created from whole paragraphs. You cannot include a subset of a paragraph in a popup.

## To use popup markers to create popup windows in a Microsoft Word source document

1. In your Microsoft Word source document, create a link between a word or phrase in the topic and the content you want to display in the popup window. For more information, see "Creating Popup Window Links in Word".
2. Insert your cursor in front of the text you want to display in the popup window.
3. On the **WebWorks** menu, click **Markers**.
4. ***If you want the popup content to display in both a popup window and in a standard help topic,*** complete the following steps:
  - a. Select **Popup** from the list of markers in the **Marker** field.
  - b. Leave the **Value** field blank.
  - c. Click **OK** to insert the marker.
5. ***If you want the popup content to display only in a popup window,*** complete the following steps:
  - a. Select **PopupOnly** from the list of markers in the **Marker** field.
  - b. Leave the **Value** field blank.
  - c. Click **OK** to insert the marker.
6. Specify where you want the popup content to end by completing the following steps:
  - a. Insert your cursor at the end of the content you want to display in the popup window.
  - b. On the **WebWorks** menu, click **Markers**.
  - c. Select **PopupEnd** from the list of markers in the **Marker** field.
  - d. Leave the **Value** field blank.
  - e. Click **OK** to insert the marker.
7. Save your Microsoft Word source document.
8. Generate output for your project. For more information, see "Generating Output".

9. In Output Explorer, go to the page where you created the popup window and verify that ePublisher created the popup window that the popup window displays the content you specified. For more information, see “Viewing Output in Output Explorer”.

## Using Paragraph Styles to Create Popup Windows in Word

You can use Popup paragraph styles in your Microsoft Word source documents to create popup windows. To use Popup paragraph styles to create popup windows, your Stationery and Microsoft Word template must have the following items configured:

- Popup and Popup Append paragraph style behaviors if you want your content to display both in popup windows and in standard help topics.
- Popup Only and Popup Only Append paragraph style behaviors if you want your content to display only in popup windows.

Your output format must also support this feature.

The following procedure provides an example of how to use Popup paragraph styles to create popup windows in Microsoft Word source documents using Microsoft Word 2003. Steps for using Popup paragraph styles to create popup windows in Microsoft Word may be different in other versions of Microsoft Word.

## To create popup windows using Popup paragraph styles in a Microsoft Word source document

1. In your Microsoft Word source document, create a link between a word or phrase in the topic and the content you want to display in the popup window and ensure that the link resolves in the document. For more information, see "Creating Popup Window Links in Word".
2. Save your Microsoft Word source document.
3. In the ePublisher **Style Designer**, configure the destination paragraph styles with the appropriate popup behavior via the **Options** panel.
4. Generate output for your project. For more information, see "Generating Output".
5. In Output Explorer, go to the page where you created the popup window and verify that ePublisher created the popup window and that the popup window displays the content you specified. For more information, see "Viewing Output in Output Explorer".

## Creating Expand/Collapse Sections (Drop-Down Hotspots) in Word

You can create sections of content that expand and collapse when you click a link or hot spot. This structure allows you to create items, such as tasks with numbered procedures, bulleted lists, or definitions, that are easy to scan. Users can then expand individual items to display additional information.

Hot spots for expand/collapse sections initially display in one of the following states:

- The content is initially collapsed and will expand beneath the hotspot when the user clicks the hotspot. Clicking the hotspot a second time causes the expanded content to return to its original collapsed state.
- The content is initially expanded and will collapse or disappear from beneath the hotspot when the user clicks the hotspot.

You create expand/collapse sections in Microsoft Word source documents by using the following items:

- An Expand/Collapse paragraph style
- A DropDownEnd marker

You use an Expand/Collapse paragraph style to start expand/collapse sections and a DropDownEnd marker to specify where the content in the expand/collapse section

ends. The Stationery defines whether the sections should initially be expanded (shown) or collapsed (hidden) and the image used to show the state of the section.

To create expand/collapse sections, your Stationery and template must have the following items configured:

- An Expand/Collapse paragraph style
- A DropDownEnd marker

Your output format must also support this feature.

The following procedure provides an example of how to create expand/collapse sections in Microsoft Word source documents using Microsoft Word 2003. Steps for creating expand/collapse sections in Microsoft Word may be different in other versions of Microsoft Word.

## To create an expand/collapse section in a Microsoft Word source document

1. In your Microsoft Word source document, identify a topic that contains text for which you want to create an expand/collapse section.
2. Apply an Expand/Collapse paragraph style to the text you want users to click to expand or collapse content.

For example, in the following sample procedure, you would apply the Expand/Collapse paragraph style to the *To open a project* text.

*To open a project*

- a. On the **File** menu, click **Open**.
  - b. Browse to the location of the project on your local computer.
  - c. Select the project, and then click **Open**.
3. Insert your cursor at the end of the content you want to display in the expand/collapse section.

For example, in the following sample procedure, you would insert your cursor after the period in the last sentence of the procedure, *Select the project, and then click Open*.

*To open a project*

- a. On the **File** menu, click **Open**.
  - b. Browse to the location of the project on your local computer.
  - c. Select the project, and then click **Open**.
4. On the **WebWorks** menu, click **Markers**.
  5. In the **Markers** field, select the **DropDownEnd** marker.
  6. Leave the **Value** field blank.
  7. Click **OK**. ePublisher inserts a DropDownEnd marker at your insertion point. This marker identifies where the contents of your expand/collapse section will end.
  8. Save your Microsoft Word source document.
  9. Generate output for your project. For more information, see "Generating Output".

10. In Output Explorer, go to the page where you created the expand/collapse section and verify that ePublisher created the expand/collapse section and that the expand/collapse section displays the content you specified. For more information, see “Viewing Output in Output Explorer”.

## Creating Related Topics in Word

Related topics provide a list of other topics that may be of interest to the user viewing the current topic. For example, you could have a section called Creating Web Pages in your help. You may also have many other topics, such as HTML Tags and Cascading Style Sheets, that related to creating Web pages. Identifying these related topics for users can help them find the information they need and identify additional topics to consider. However, providing these types of links as cross-references within the content itself may not be the most efficient way to present the information. By utilizing related topics links, you combine the capabilities of cross-references with the efficiency of a related topics button.

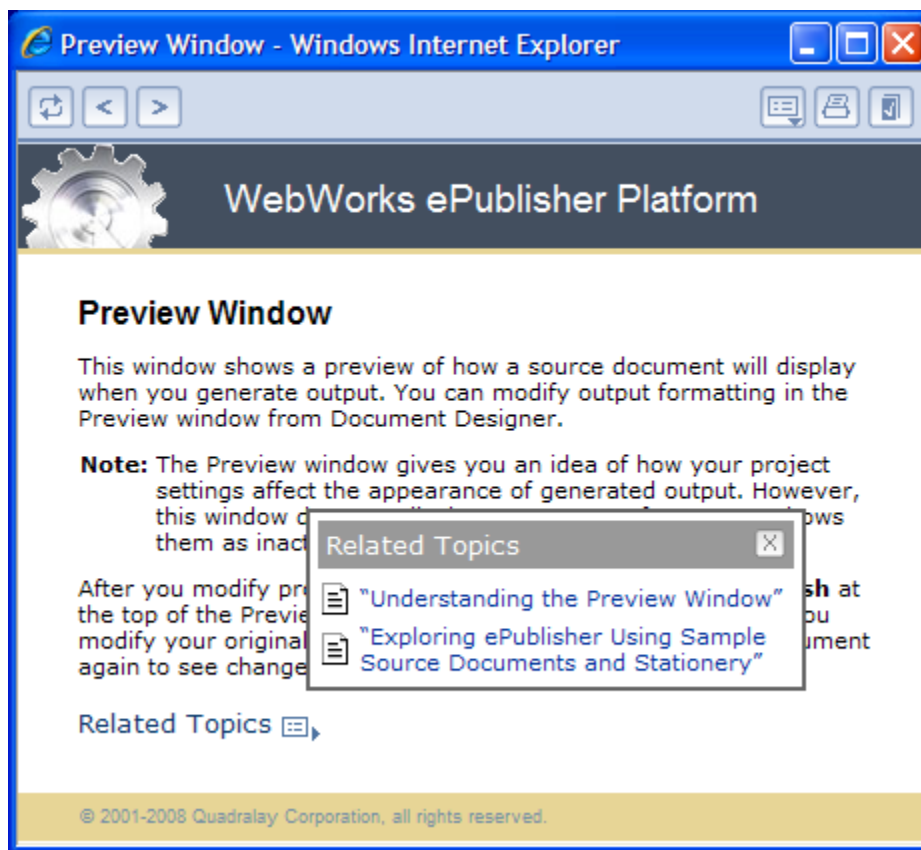
Related topics and See Also links provide similar capabilities, but there are several important differences:

- Related topics can link to headings in a Help system that do not start a new page.
- Related topics links are static and defined in the source documents as links. You must have all the source documents to create the link and generate the output.
- If a related topics list contains a broken link in the source document, that link is broken in the generated output. In a See Also link list, the broken link is not included in the output.

The Stationery designer can configure related topics to display in the following ways:

- Included as a list in the topic itself.
- Displayed in a popup window when the user clicks a button, as show in the following figure.





**Note:** If a related topic link is broken in the source document, in most cases that link is broken in the generated output. WebWorks Help and WebWorks Reverb provide an additional feature by removing broken links from related topics lists that are displayed in a popup window when a user clicks the Related Topics button.

To create related topics links, your Stationery and template must have a Related Topics paragraph style configured. Your output format must also support this feature.

The following procedure provides an example of how to create related topics links in Microsoft Word source documents using Microsoft Word 2003. Steps for creating related topics links in Microsoft Word may be different in other versions of Microsoft Word.

## To create a related topics list in a Microsoft Word source document

1. Identify the topic in which you would like to insert a related topics list.
2. Identify the different topics you want to link to from this topic.

**Note:** Generally, you should only create one related topics list for each section of your source document that corresponds to a help topic. For example, if the Stationery designer specified in your Stationery that there will be a page break at each Heading 1 section, then you should only create one related topics list for each Heading 1 section within your source document.

3. Create a cross-reference to each topic you want to include in the related topics list by completing the following steps:
  - a. Insert your cursor in the location in your Microsoft Word source document where you want to insert the link to the related topic.
  - b. On the **Insert** menu, click **Reference > Cross-reference**.
  - c. In the **Reference type** field, select **Heading**.
  - d. In the **Insert reference to** field, click **Heading text**.
  - e. Select the insert as hyperlink checkbooks.
  - f. In the **For which heading** field, select the heading to which you want to cross reference.
  - g. Click **Insert**.
  - h. Click **Close**.
4. Apply the Related Topic paragraph style to the cross-references in your related topics list.
5. ***If you want to display the list of related topics in only your generated output***, apply an OnlineOnly condition to the list of related topics. For more information about applying conditions, refer to "Applying Conditions in Word".
6. Save your Microsoft Word source document.
7. Generate output for your project. For more information, see "Generating Output".
8. In Output Explorer, go to the page where you created the related topics list and verify that ePublisher created the related topics and that the related

topics list displays the topics you specified. For more information, see "Viewing Output in Output Explorer".

## **Creating Links to PDF in Word**

You have the ability to link to different information in an external document such as a PDF with a hyperlink to the content.

## To create an external hyperlink to a PDF document

1. In the Word menu, go to **Insert > Hyperlink**.
2. Select **Existing File or Web Page** for the **Link to** label located on the left.
3. In the **Text to display** text box, enter the text you would like for the hyperlink
4. Navigate to the file location on the system to link against for the PDF.
5. Save the Word document.

## Creating See Also Links in Word

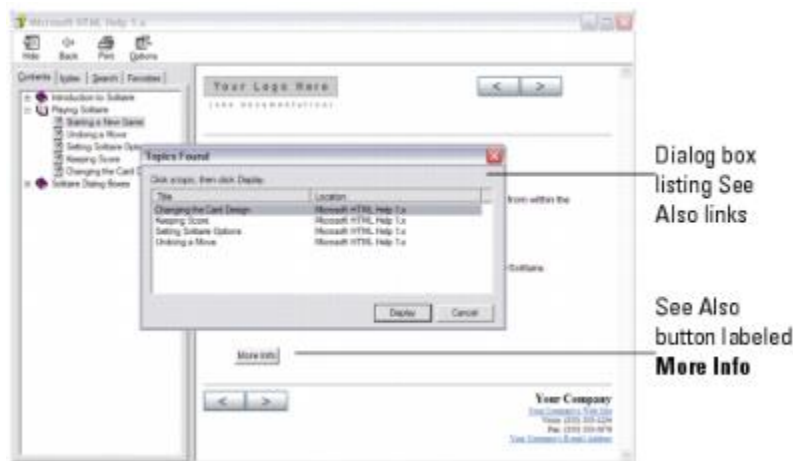
See Also links, also known as ALinks, or associative links, are links that may be of interest to the user viewing the current topic. These links use internal identifiers to specify the links and the link list is built dynamically based on the topics available when the user clicks to display the links. See Also links are important to use with larger help sets and merged help sets.

Related topics and See Also links provide similar capabilities, but there are several important differences:

- See Also links must link to styles that start a new topic, such as a heading.
- See Also links are dynamic and the lists of links are built at display time instead of during help generation.
- Since see Also link lists are dynamically built, they do not include links to topics that are not available when the user displays the links. If a related topics list contains a broken link in the source document, that link is broken in the generated output for most output formats.

See Also links are useful if you plan to merge help systems. For example, if you have a multiple help systems that you merge into one main help system at run time and if your topics in the merged help systems contain See Also keywords that are also used in the main help system, links to those topics are included in the See Also lists in the main project.

You can create See Also links as buttons or as inline text links in Microsoft HTML Help and WebWorks Help. The following example shows how the two different types of See Also links display in a Microsoft HTML Help system.



Create See Also links by applying the See Also paragraph style or character style to text in your Microsoft Word source documents and inserting markers into your Microsoft Word source documents.

Create See Also links by applying the See Also paragraph style or character style to text in your Microsoft Word source documents and inserting markers into your Microsoft Word source documents. To create See Also links, your Stationery and template must have the following items configured:

- See Also paragraph style if you want to create See Also links with buttons
- See Also paragraph style if you want to create see Also links as inline text links
- SeeAlsoKeyword marker type
- SeeAlsoLink marker type
- SeeAlsoLinkDisplay marker type if you generate Microsoft HTML Help and you want to display the target topics in a popup menu

SeeAlsoLinkWindowType marker type if you generate Microsoft HTML Help and you want to display the target topics in a custom window

The following procedure provides an example of how to create See Also links in Microsoft Word source documents using Microsoft Word 2003. Steps for creating See Also links in Microsoft Word may be different in other versions of Microsoft Word.

## To create a See Also link in a Microsoft Word source document

1. Identify each topic to which you want to link from a See Also link, and then complete the following steps for each topic:

- a. Insert your cursor into the topic to which you want to link.
- b. On the **WebWorks** menu, click **Markers**.
- c. In the **Marker** field, select the **SeeAlsoKeyword** marker.
- d. In the **Value** field, type a text string that is a unique identifier as the See Also keyword for the topic. See Also keywords are case sensitive and cannot contain punctuation or spaces.

For example, if you have a unique topic called About WebWorks Help, type `AboutWebWorks` help in the **Value** field.

- e. Click **OK**. ePublisher inserts a SeeAlsoKeyword marker at your insertion point. This marker identifies the topic for See Also links.
2. Identify the topic where you want to insert a list of See Also links.
  3. Enter the text you want to display for the See Also button or for the See Also inline text link on a separate line in the source document where you want the See Also button or inline text link to display.

For example, if you want to create a button with the text See Also on the button, type `See Also`. If you want to create inline text with the text Additional Information for the link, type `Additional Information`.

4. **If you want to create a See Also button for your See Also links**, apply the See Also paragraph style to the text you want to display in the See Also button.
5. **If you want to create a See Also inline text link for your See Also links**, apply the See Also character style to the text you want to display in the See Also inline text link.
6. Apply an OnlineOnly condition to the See Also text. Applying an OnlineOnly condition to the See Also button or See Also inline text displays the See Also link in your generated output, but does not display the See Also button or link in your printed content.
7. Insert your cursor inside the text you specified for the See Also button or See Also inline text link.
8. For each topic to which you want to link from a See Also link, complete the following steps:

- a. On the **WebWorks** menu, click **Markers**.
- b. In the **Marker** field, select the **SeeAlsoLink** marker.
- c. In the **Value** field, type the text string that is a unique identifier for the topic to which you want to link. This text string is the text string you typed when you created the **SeeAlsoKeyword** marker for the topic.

For example, if you created a **SeeAlsoKeyword** marker with the text string AboutWebWorksHelp, type `AboutWebWorksHelp` in the **Value** field for the **SeeAlsoLink** marker.

- d. Click **OK**. ePublisher inserts a **SeeAlsoLink** marker at your insertion point. This marker identifies the topics users can link to when they click the See Also button.

**9. If you generate Microsoft HTML Help output and you want to display the target topics in a popup menu,** complete the following steps:

- a. Insert your cursor inside the text you specified for the See Also button or inline text link.

- a. On the **WebWorks** menu, click **Markers**.

- b. In the **Marker** field, select the **SeeAlsoLinkDisplayType** marker.

**Note:** This marker type is supported only in Microsoft HTML Help.

- c. In the **Value** field, type `menu`. By default, Microsoft HTML Help displays See Also links in the Topics Found window. To display See Also links in a popup menu, specify menu for the marker value.

- d. Click **OK**. ePublisher inserts a SeeAlsoDisplayType marker at your insertion point.

**10. If you generate Microsoft HTML Help output and you want to display the target topics in a custom window,** complete the following steps:

- a. Insert your cursor inside the text you specified for the See Also button or See Also inline text link.

- b. On the **WebWorks** menu, click **Markers**.

- c. In the **Marker** field, select the **SeeAlsoLinkWindowType** marker.

**Note:** This marker type is supported only in Microsoft HTML Help.

- d. In the **Value** field, type the name of a custom window defined for Microsoft HTML Help by the Stationery designer.

For example, if the Stationery designer defined a custom window called ContextHelp to use to when displaying context-sensitive help topics, type `ContextHelp` in the **Value** field for the SeeAlsoLinkWindowType marker.

- e. Click **OK**. ePublisher inserts a SeeAlsoDisplayType marker at your insertion point.

**11.** Save your Microsoft Word source document.

**12.** Generate output for your project. For more information, see "Generating Output".

**13.** In Output Explorer, go to the page where you created the See Also links and verify that ePublisher created the See Also button or See Also inline text and that the See Also button or inline text displays the links you specified. For more information, see "Viewing Output in Output Explorer".

## Creating Meta Tag Keywords in Word

Meta tags are lines of code placed between the `<head>` and `</head>` tags in HTML pages. Meta tags give web search engines information about the content of the web page and how search engines should treat the web page. Users viewing web pages do not see the meta tags, but meta tags can be used to influence the way web pages on a web site appear in web search engine results. Users also see the text you specify for meta tags right following the title of your page when your page comes up in search results.

In help systems, search ranking works like ranking in an Internet search engine. If you generate help system output, you can use meta tag keywords to specify terms for pages for help topics where you want to improve searchability. For example, assume that in your help system you have a topic called See Also links. However, you know that See Also links are also sometimes referred to as ALinks, and you think that some users of your help system may search for information about See Also links by typing `ALinks` into the **Search** field for your help system. In this example, you can insert ALinks as a meta tag keyword for each page that discusses See Also links, so users who search your system for information about ALinks can find the information they are looking for in your See Also link topics.

To assign meta tag keywords, your Stationery and template must have the Keywords marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to create meta tag keywords in Microsoft Word source documents using Microsoft Word 2003. Steps for creating



meta tag keywords in Microsoft Word may be different in other versions of Microsoft Word.

## To create meta tag keywords for a page in a Microsoft Word source document

1. In your Microsoft Word source document, find the first paragraph in the page for the page for which you want to create a meta tag keyword.
2. On the **WebWorks** menu, click **Markers**.
3. In the **Marker** field, select **Keywords** from the list of markers.
4. In the **Value** field, type the comma-delimited list of keywords that you want web search engines to use when crawling Web sites and to display immediately following the title of your page when your page comes up in search results.

For example, type `keyword1, keyword2, keyword3`, where *keyword* is the keyword you want web search engines to use when crawling your Web site.

5. Click **OK**.
6. Save your Microsoft Word source document.
7. Generate output for your project. For more information, see "Generating Output".
8. In Output Explorer, verify that ePublisher inserted your meta tag keywords correctly by completing the following steps:
  - a. On the **View** menu, click **Output Explorer**.
  - b. In the *TargetName\ProjectName* folder, open the page to which you assigned meta tag keywords in Notepad, where *TargetName* is the name of your target and *ProjectName* is the name of your project.
  - c. Verify that the text you specified for your meta tag displays in the `meta name` attribute between in the `<head>` and `</head>` tags section of your web page. For example, if you typed `keyword1, keyword2, keyword3`, for your meta tag keywords, your meta tags in for the page should be similar to the following entry:

```
<meta name="keywords" content="keyword1, keyword2, keyword3" />
```

## Assigning Custom Page Styles in Word

By default, each page generated by ePublisher is associated with the default page style defined in the Stationery used by your ePublisher project. This means that typically you do not need to specify a page style for pages when you generate

output. However, if you want to change the page style of one page or a smaller set of pages, you can specify the page style you want to use for a page in your Microsoft Word source document using the PageStyle marker.

For example, you may want to use one page style in your help system for all concept and procedure topic pages, and another page style for all context-sensitive window description topic pages in your help system. In this example, you can use the default page style for all of your concept and procedure topic pages, and then you can use a second custom page style defined in your Stationery for all context-sensitive window description topic pages in your help system.

Before you begin, obtain the names of the custom page styles you can use with your Stationery from the Stationery designer. Then insert a PageStyle marker with the page style name into the topic you want to display using a custom page style. After you assign a custom page style to a topic using the PageStyle marker, the generated output displays the topic using the specified page style.

To assign custom page styles, your Stationery and template must have the following items configured:

- Custom page styles defined for your Stationery by the Stationery designer
- PageStyle marker type

Your output format must also support this feature.

The following procedure provides an example of specifying page styles for pages in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying page styles for pages in Microsoft Word may be different in other versions of Microsoft Word.

## To specify a custom page style for a page in a Microsoft Word source document

1. In your Microsoft Word source document, locate the page for the topic to which you want to assign a page style.
2. Insert your cursor in the location on the page where you want to insert the **PageStyle** marker.
3. On the **WebWorks** menu, click **Markers**.
4. In the **Marker** field, select **PageStyle** from the list of markers.
5. In the **Value** field, type the name of the page style you want to associate with the page.

For example, if you Stationery designer configured a page style for your Stationery called YellowBackground, type `YellowBackground`.

6. Click **OK**. ePublisher inserts the PageStyle marker into your source document.
7. Save your Microsoft Word source document.
8. Generate output for your project. For more information, see “Generating Output”.
9. In Output Explorer, verify ePublisher created the page using the page style you specified by clicking on the page and verifying ePublisher applied the page style you specified in the generated output. For more information about viewing output files in Output Explorer, see “Viewing Output in Output Explorer”.

## Creating What’s This (Field-Level) Help in Word

If you generate Microsoft HTML Help output, you can implement What’s This help for product dialog boxes and windows. What’s This Help is also known as field-level help. Only Microsoft HTML Help supports field-level help. In addition, not all products are designed to support field-level help for product dialog boxes and windows. Before you begin implementing field-level help, consult your product team to determine if field-level help part of the product design. If field-level help is part of the product design, you will also need to obtain the appropriate ID from your product team for each field-level help topic you need.

Users can view the field-level help you create using one of following methods:

- Users click on the question mark icon in the upper right corner of the dialog box or window.

When users click on the question mark icon, their cursor changes to a question mark. Users can then move the question mark cursor over the fields on the dialog box or window, and Windows displays the field-level help you created in a popup window when they hover over a specific field.

- Users right-click a field on a dialog box or window and then select the **What's This?** option from the single option menu Windows displays.

After users select this option, Windows displays the field-level help you specify in in a popup window.

Users close the popup window that provides the field-level help by pressing the `ESC` key on the keyboard. When users press the `ESC` key, their cursor returns to the regular cursor shape for the user.

To create What's This help, your Stationery and template must have the following items configured:

- What Is This help paragraph style

WhatIsThisHelpIDThe following procedure provides an example of how to create What's This help in Microsoft Word source documents using Microsoft Word 2003. Steps for creating What's This help in Microsoft Word may be different in other versions of Microsoft Word.

## To create What's This help in a Microsoft Word source documents

1. Identify a topic that contains field-level help.
2. Apply the What Is This paragraph style to the text that contains the field-level help.
3. Insert your cursor into the field-level help text.
4. On the **WebWorks** menu, click **Markers**.
5. In the **Marker** field, select **WhatIsThisID** from the list of markers.
6. In the **Value** field, type the appropriate ID for the field-level description. Obtain appropriate IDs for each field-level description from your product team.
7. Click **OK**.
8. Save your Microsoft Word source document.
9. Generate output for your project. For more information, see "Generating Output".
10. In Output Explorer, verify ePublisher created the What's This help you specified by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. Open the `ProjectName` folder, where *ProjectName* is the name of your project.
  - c. Open the `whatisthis.txt` file and verify that the field-level help you created is associated with the correct ID you received from your product development team.
  - d. Open the `whatisthis.h` file and verify that each new string you added is listed in the file.

## Opening Topics in Custom Windows in Word

You can open topics in custom windows in Microsoft HTML Help and Oracle Help. By default, Microsoft HTML Help displays content in the standard Microsoft HTML Help tri-pane window. The Stationery designer can modify the size, position, and other characteristics of the tri-pane window in your Microsoft HTML Help project. The Stationery designer can also define custom windows for you to use in a Microsoft

HTML Help project. If the Stationery designer defines custom windows in a Microsoft HTML Help project, you can specify which topics you want to display in the custom window using the WindowType marker.

By default, Oracle Help displays content in the standard Oracle Help viewer. The Stationery designer can modify the size, position, and other characteristics of Oracle Help windows. The Stationery designer can also define custom windows for you to use in an Oracle Help project. If the Stationery designer defines custom windows in an Oracle Help project, you can specify which topics you want to display in the custom window using the WindowType marker.

For example, if you want your context-sensitive help topics to display in a different type of window than other content, after you create a context-sensitive help topic you can use the WindowType marker to specify that you want the context-sensitive help topics to display in a custom window. After you assign a custom window to a topic using the WindowType marker, the help system displays the topic in your generated output in the custom window whenever users access the topic from the table of contents, index, a standard hyperlink, a related topics list, or a See Also link.

To open topics in custom windows, your Stationery and template must have the following items configured:

- Custom window styles defined for your Stationery by the Stationery designer
- PageStyle marker type

The following procedure provides an example of how to specify topics open in custom Microsoft HTML Help or Oracle Help windows in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying topics open in custom Microsoft HTML Help or Oracle Help windows in Microsoft Word may be different in other versions of Microsoft Word.

## To specify topics open in a custom window in a Microsoft Word source document

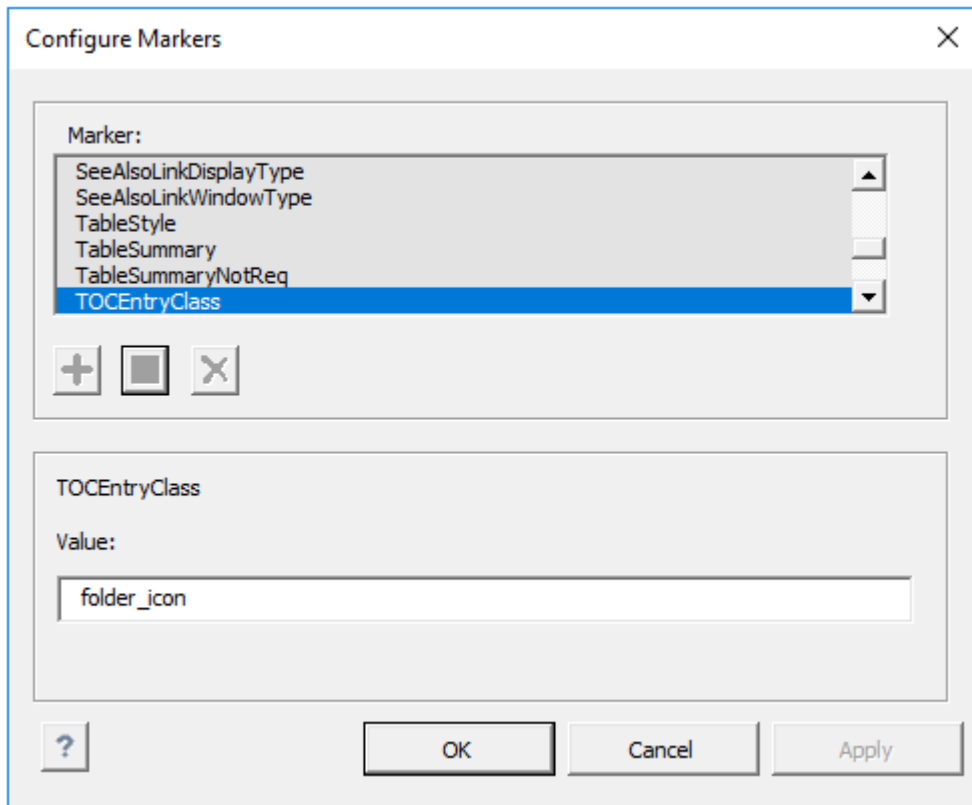
1. Obtain the names of custom windows configured in the Stationery you use for your ePublisher project from the Stationery designer.
2. In your Microsoft Word source document, locate the topic that you want to open in a custom window.
3. Insert your cursor into the topic.
4. On the **WebWorks** menu, click **Markers**.
5. In the **Marker** list, select **WindowType** from the list.
6. In the **Value** field, type the name of the custom window configured by the Stationery designer that you want to specify for the topic.
7. Click **OK**.
8. Save your Microsoft Word source document.
9. Generate output for your project. For more information, see "Generating Output".
10. In Output Explorer, verify the topic displays in the custom window you specified for the topic. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Customizing TOC Entry in Word

Use these steps to customize a TOC entry in your **Reverb 2.0** output. Your Word file must have a nested heading structure for TOC Icons to appear.

1. In your Word document, click in or highlight the header that will have the customized TOC entry.
2. Click Markers from the Transit menu in Word.
3. Select TOCEntryClass.





4. Give the TOCEntryClass a value. This value will become the class. In the example, `folder_icon` is used.
5. Save your Word Document.
6. Scan the document in ePublisher Designer.
7. Open the **Style Designer**.
8. Open **Marker Styles**.
9. Locate the **Marker Type Option** from the **Options** tab and set its value to `TOC Entry Class`.
10. In this example, the assigned class for the Menu TOC entry will be the value of the marker: `folder_icon`.
11. Add the following to a target override of `_icons.scss`. Notice how the CSS class is the name of the value given in the Marker Text Window. In this example we change the icon color and the icon of the TOC entry. You are able to make other customizations such as adding a border, or changing the background color.

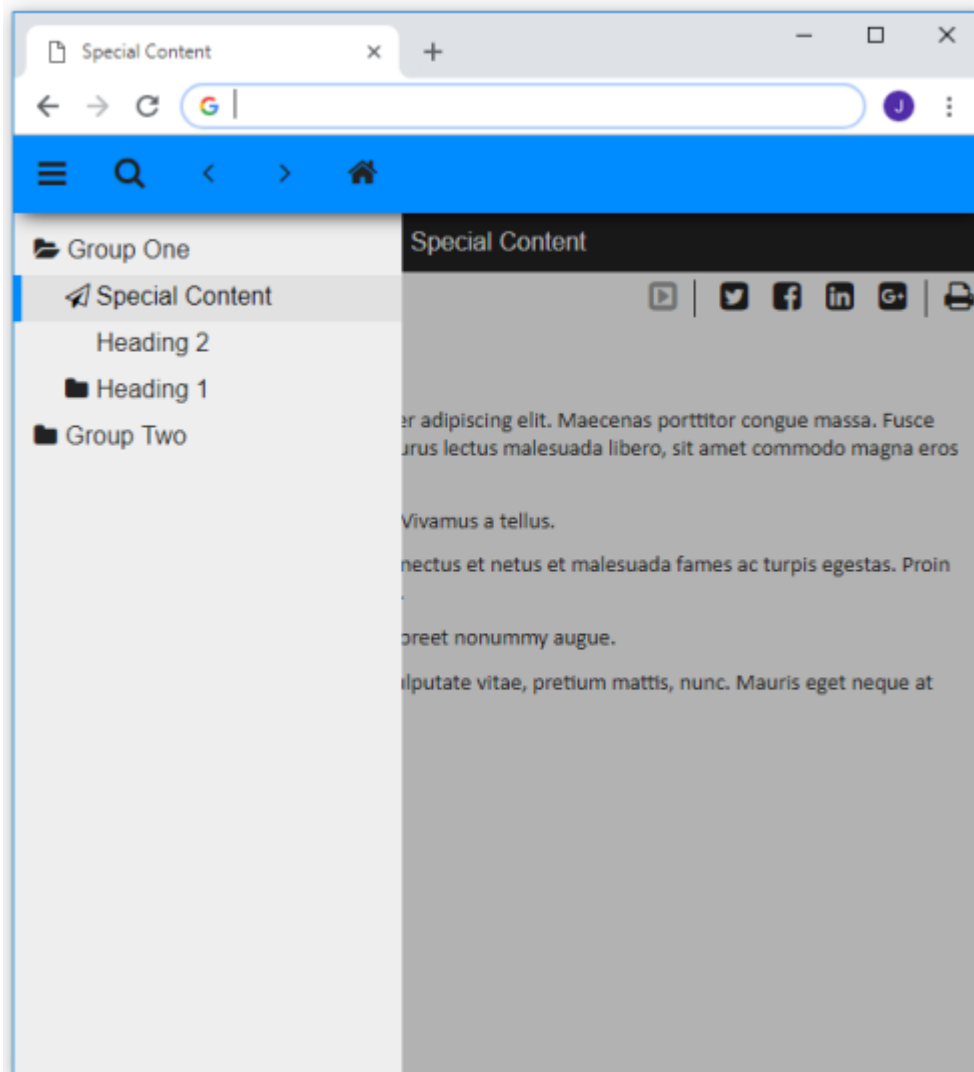
```

.folder_icon {
  > div > span > i {
    color: black;

    &:before {
      content: $folder_icon;
    }
  }
}

```

**12.** Save your project and generate the output.



# Customizing Table of Contents Icons in Word

By default, the **Contents** tab in a Microsoft HTML Help, Oracle Help, and WebWorks Help uses book and page icons to identify entries. By default, the **Contents** tab in Sun JavaHelp uses folder and page icons to identify entries. You can also customize the table of contents icons.

For example, if you want to make new topics stand out by using a unique icon specific to new books, pages, or folders, you can insert a marker into a topic and specify the icon you want to display for the book, page, or folder in your help system table of contents.

To customize a table of contents icon, your Stationery and template must have the following items configured:

- TOCIconHTMLHelp for Microsoft HTML Help
- TOCIconOracleHelp for Oracle Help
- TOCIconJavaHelp for Sun JavaHelp
- TOCIconWWHelp for WebWorks Help

The following procedure provides an example of how to customize table of contents icons for topics in Microsoft Word source documents using Microsoft Word 2003. Steps for customizing table of contents icons for topics in Microsoft Word may be different in other versions of Microsoft Word.

## To specify a custom table of contents icon in a Microsoft Word source document

1. ***If you want to specify a custom table of contents icon for Microsoft HTML Help***, identify the number of the image you want to use for the table of contents image for the topic in the `.hhp` file for your Microsoft HTML Help project by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. Open the `ProjectName` folder, where *ProjectName* is the name of your project.
  - c. Open the `ProjectName.hhp` file where *ProjectName* is the name of your project.
  - d. On the **Contents** tab, select a table of contents entry, and then click the **Pencil** icon.
  - e. On the **Advanced** tab, in the **Image index** field, use the up and down arrows to identify the table of contents image you want to use for the topic.
  - f. Note the number of the image you want to use for the table of contents image for the topic.

For example, if you want to use a question mark icon with a red star for the table of contents icon for new topics, note that the number for this icon is 10.
  - g. Close HTML Help Workshop.
2. ***If you want to specify a custom table of contents icon for Oracle Help or Sun JavaHelp***, create the graphic file for the custom table of contents icon in `.gif` format. The default graphics used as Sun JavaHelp or Oracle Help table of contents icons are 17 x 17 pixels. The custom graphics you create for Sun JavaHelp or Oracle Help table of contents icons should also be 17 x 17 pixels. You can assign any name to the graphic files.
3. ***If you want to specify a custom table of content icon for WebWorks help***, create graphics files containing the collapsed and expanded versions of the icons you want to use, then save the graphic files in `.gif` format. The default graphics used as WebWorks Help table of contents icons are 17 x 17 pixels. The custom graphics you create for WebWorks Help table of contents icons should also be 17 x 17 pixels. You can assign any name to the graphic files.
4. Copy the graphic files you want to use as icons in the table of contents into the following folder:

**Note:** If the folder does not exist, first create the folder using the specified folder structure and then copy the graphic files you want to use as icons into the folder. You do not need to perform this step when specifying custom table of contents icons for Microsoft HTML Help.

- **If you are generating Oracle Help**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Oracle Help\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating Sun JavaHelp 1.1.3**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Sun Java Help 1.1.3\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating Sun JavaHelp 2.0**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Sun Java Help 2.0\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating WebWorks Help**, in your *ProjectName* `\Files` folder, where *ProjectName* is the name of your project, create a `wwhelp\images` subfolder and copy the graphic files you want to use into this folder. Your project file structure should be similar to the following structure:

`ProjectName\Files\wwhelp\images`, where *ProjectName* is the name of your project.

5. In your Microsoft Word source document, locate the topic where you want to use the custom table of contents icon.
6. Insert your cursor into the heading for the topic.
7. On the **WebWorks** menu, click **Markers**.
8. In the **Marker** list, select the appropriate TOCIcon marker type from the list.
9. In the **Value** field, type the following text:
  - **If you are generating Microsoft HTML Help**, type the number of the icon that you want to use for the table of contents image.

For example, if you want to use a question mark icon with a red star for the table of contents icon for new topics, type `10`.

- **If you are generating Oracle Help or Sun JavaHelp**, type the following text:

```
images/TOCIcon.gif
```

where *TOCIcon*.gif is the name of the table of contents icon you want to display for the topic.

- **If you are generating WebWorks Help**, type the following text:

```
c="collapsed.gif" e="expanded.gif"
```

where *collapsed.gif* is the name of the icon you want to use when the table of contents entry is collapsed, and *expanded.gif* is the name of the icon you want to use when the table of contents entry is expanded. If the table of contents entry is for a page instead of a book, the entry will never be expanded, so you can omit the *e="expanded.gif"* portion of the entry for pages.

For example, you might create a special icon to highlight books that are new for a particular release of your WebWorks Help system. If you named these icons *newbookopen.gif* and *newbookclosed.gif*, you would type the following text into the **Value** field:

```
c="newbookclosed.gif" e="newbookopen.gif"
```

10. Click **OK**.
11. Save your Microsoft Word source document.
12. Generate output for your project. For more information, see "Generating Output".
13. In Output Explorer, verify ePublisher created the table of contents using the table of contents icon you specified for the topic. For more information about viewing output files in Output Explorer, see "Viewing Output in Output Explorer".

## Specifying Context Plug-ins in Word

You can specify Eclipse Help context plug-ins by using Context Plugin markers in your source documents. ePublisher places the context plug-ins you specify in your source documents in the *plugin.xml* file generated for each source document group you have in Document Manager. You can then have developers use the context plug-ins defined in *plugin.xml* files to call your Eclipse Help system as appropriate from Eclipse plug-ins.

For example, assume you have the following three top-level groups in Document Manager for your Eclipse Help system target:

- Component A group - contains the source documents for ComponentA Feature1 and ComponentA Feature2
- Component B group - contains the source documents for ComponentB Feature1 and ComponentB Feature 2
- Component C group - contains the source documents for ComponentC Feature1 and ComponentC Feature 2

You insert the following Context Plugin markers into the source documents for each group:

- ComponentAFeature1 and ComponentAFeature2 Context Plugin markers in source documents contained in the ComponentA group
- ComponentBFeature1 and ComponentBFeature2 Context Plugin markers in source documents contained in the ComponentB group
- ComponentCFeature1 and ComponentCFeature2 Context Plugin markers in source documents contained in the ComponentC group

When you generate your Eclipse Help system, ePublisher creates the following folder structure in the *ProjectName*\Output\*TargetName* folder, where *ProjectName* is the name of your ePublisher project, and *TargetName* is the name of your target:

- **ComponentA** folder, which contains a `plugin.xml` file with the following entries:

```
plugin="ComponentAFeature1ContextPlugin"
```

```
plugin="ComponentAFeature2ContextPlugin"
```

- **ComponentB** folder, which contains a `plugin.xml` file with the following entries:

```
plugin="ComponentBFeature1ContextPlugin"
```

```
plugin="ComponentBFeature2ContextPlugin"
```

- **ComponentC** folder, which contains a `plugin.xml` file with the following entries:

```
plugin="ComponentCFeature1ContextPlugin"
```

```
plugin="ComponentCFeature2ContextPlugin"
```

You can then provide the context plug-in IDs in your `plugin.xml` files to the appropriate Eclipse developers to use. The Eclipse developers use the context plug-ins defined in `plugin.xml` files to call your Eclipse Help system as appropriate from Eclipse plug-ins.



## To specify a context plug-in in a Microsoft Word source document

1. Identify a topic in a source document where you want to insert the context plug-in.
2. On the **WebWorks** menu, click **Markers**.
3. In the **Marker Type** field, select **Context Plugin** from the list of markers.
4. In the **Value** field, type the appropriate ID for the context plug-in.

**Note:** If you are responsible for defining the ID, ensure you supply the context plug-in ID to your developers to use as appropriate for their Eclipse plug-ins. If your developers define the ID, use the context plug-in ID you obtained from your developers.

5. Click **OK**.
6. Save your Microsoft Word source document.
7. Generate output for your project. For more information, see "Generating Output".
8. In Output Explorer, verify ePublisher generated a `plugin.xml` file that contains the context plug-in IDs you specified by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. Open the *ProjectName* folder, where *ProjectName* is the name of your project.
  - c. Open the group folder for a group that contains the source documents where you specified your context plug-in ID.
  - d. Open the `plugin.xml` file in Notepad and verify that the context plug-in IDs you specified in your source documents are listed in the `plugin.xml` file. Your context plug-in IDs should be listed in the Contexts area of the file. Following is an example of how the context plug-in IDs you specified in your source documents should be displayed in the `plugin.xml` file:

```
<!-- Contexts -->
```

```
<!-- -->
```

```
<extension point="org.eclipse.help.contexts">
```

```
<contexts file="contexts.xml"
plugin="ComponentAFeature1ContextPlugin" />

</extension>

<extension point="org.eclipse.help.contexts">

<contexts file="contexts.xml"
plugin="ComponentAFeature2ContextPlugin" />

</extension>
```

## Creating Accessible Online Content in Word

Accessible content is content that can be easily accessed by users with certain disabilities. This section explains how you can prepare your Microsoft Word source documents to ensure your content is accessible to users using assistive technologies.

### Accessible Content in Word

Images and tables are helpful ways to convey information to end users. However, users with disabilities often cannot access the important information provided by images and table layouts in online content. You should document images and other non-text items such as table layouts so that users using assistive technologies to access online content can access the information these items provide.

Content that must easily be accessed by people with disabilities must conform to certain guidelines published by both the W3C and the United States government in order to produce accessible online output, also known as Section 508 compliant output. These guidelines are intended to help writers produce accessible content.

You can use ePublisher to help you produce online content that conforms to the W3C Web Content Accessibility Guidelines 1.0 (WCAG), Section 508 of the U.S. Rehabilitation Act of 1998, and the Americans with Disabilities Act (ADA). If you are required to generate accessible content, typically you provide the following items in your online content:

- Alternate text and descriptions for all images and image maps. For more information, see "Assigning Alternate Text to Images and Image Maps in Word".
- Long descriptions for all images. For more information, see "Assigning Long Descriptions to Images in Word".

- Summaries for all tables. For more information, see “Assigning Alternate Text (Summaries) to Tables in Word”.

You may also choose to provide the following items in your online content:

- Alternate text for abbreviations. For more information, see “Assigning Alternate Text to Abbreviations in Word”.
- Alternate text for acronyms. For more information, see “Assigning Alternate Text to Abbreviations in Word”.
- Citations for quotes. For more information, see “Providing Citations for Quotes in Word”.

You must prepare source documents and configure your ePublisher project in order to create accessible content. You prepare your source documents by inserting markers into your source documents and by applying character formats and paragraph formats. You configure accessibility settings in the ePublisher project. ePublisher uses the information in your source documents and your ePublisher project to generate accessible online output.

For more information about producing accessible content and to check your content further for compliance, see the following Web sites:

- For the complete W3C note on the WCAG, visit <http://www.w3c.org/TR/WCAG10-CORE-TECHS>.
- For information about the related Web Accessibility Initiative, visit <http://www.w3.org/WAI>.
- For information about Section 508 of the U.S. Rehabilitation Act of 1998, visit <http://www.w3.org/WAI/Policy/#508>.

## Accessible Content Navigation in Word

Users can navigate through the accessible content using keys on the keyboard. The following output formats support navigation keys:

- Dynamic HTML
- Microsoft HTML Help
- Oracle Help
- WebWorks Help

**Note:** For the Dynamic HTML, navigation key behavior may vary based on the browser the user uses. For example, in Netscape and Mozilla, users must hold down the **Alt** key while pressing the navigation keys. In Internet

Explorer, users must first hold down the `Alt` key while pressing the navigation key, and then press `Enter`.

The following table lists the how each output format supports navigation keys.

Navigation Key	Function	Format
1	Display the TOC	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• WebWorks Help 5.0</li> </ul>
2	Display the Index	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• WebWorks Help 5.0</li> </ul>
3	Display the Search tab	WebWorks Help 5.0
4	Go to the previous page	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• Microsoft HTML Help</li> <li>• Oracle Help</li> <li>• WebWorks Help 5.0</li> </ul> <p><b><i>If you are using Microsoft HTML Help,</i></b> <code>Alt+4</code> works only if the topic pane has the focus. If the topic pane does not have the focus, you must press <code>Alt+0</code> and then <code>Alt+4</code>.</p> <p><b><i>If you are using Oracle Help,</i></b> you must press Enter after pressing <code>Alt+4</code>.</p>
5	Go to the next page	<ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• Microsoft HTML Help 1.x</li> <li>• Oracle Help</li> <li>• WebWorks Help 5.0</li> </ul>

Navigation Key	Function	Format
		<p><b><i>If you are using Microsoft HTML Help,</i></b> the <b>Alt+5</b> key works only if the topic pane has the focus. If the topic pane does not have the focus, you must press <b>Alt+ 0</b> and then <b>Alt +5</b>.</p> <p><b><i>If you are using Oracle Help,</i></b> you must press <b>Enter</b> after pressing <b>Alt +5</b>.</p>
6	Shift the focus to the related topics list displayed at the bottom of the current page	<p>WebWorks Help 5.0</p> <p>After you press the <b>6</b> key, you can press <b>Tab</b> to cycle through the entries in the related topics list.</p>
7	Display a blank feedback e-mail (equivalent to clicking the e-mail button in the toolbar frame)	WebWorks Help 5.0
8	Print the current page (equivalent to clicking the Print button in the toolbar frame)	WebWorks Help 5.0
9	Bookmark the current page (equivalent to clicking the Bookmark button in the toolbar frame)	WebWorks Help 5.0
10	Shift the focus to the topic frame (equivalent to clicking within the topic frame)	WebWorks Help 5.0

# Validating Accessible Content in Word

After you configure your source documents and configure the appropriate settings, ePublisher uses Accessibility conformance reports to perform the following checks to verify that the generated output conforms to accessibility standards:

- Alternate text for all images
- Alternate text for all clickable regions in all image maps
- Long descriptions for all images
- Summaries for all tables

**Note:** ePublisher does not verify that you have provided alternate text for abbreviations or acronyms or verify that you have included citations for quotes. For more information about understanding and using the Accessibility conformance reports ePublisher provides, see “Configuring Reports”, and “Generating Reports”.

## Assigning Alternate Text to Images and Image Maps in Word

This section provides information about how to create accessible images and image maps in your generated output by assigning alternate text to images.

### Image and Image Map Alternate Text in Word

One of the largest accessibility challenges with online content today is the lack of alternative text for images and image maps. Sight-impaired users often use screen readers or refreshable Braille devices to read online content. However, when these assistive technologies come across images or image maps without alternative text, also known as alternate text, they are unable to provide users with information about the image or image map and its meaning.

The Web Content Accessibility Guidelines require that alternate text be provided for all images and image maps in online content. The alternate text is an image label that describes the image or each area of the image map. Online content should display alternate text for images and image maps when users perform the following actions:

- The user hovers the mouse pointer over an image or section of an image map.
- The user browser has been configured to disable display of images and image maps.
- The user browser is a text-only browser such as Lynx.

- The user uses assistive technology such as a screen reader.

The alternate text you assign to an image or sections of an image map should be as accurate and as succinct as possible and provide users with a brief description of the image and how the image relates to the page they are viewing. Make sure that your alternate text conveys all of the important information related to the image or image map section, but do not burden users with excessively long alternative text. Screen readers or refreshable Braille devices always read the alternative text, so if your page has several images or complex image maps with long descriptions, it can take a long time for the assistive devices to read image-heavy pages with long descriptions. If you need to provide a description of the image or image map section that is more than a few words or a few short sentences, you should provide a brief alternate text description of the image or image map section and then assign a longer description the image using either the `longdesc` attribute or a description. Once you specify a long description using the `longdesc` attribute, you can also optionally display a D link next to the image. For more information about assigning long descriptions to images, see "Assigning Long Descriptions to Images in Word".

## Assigning Alternate Text to Images in Word

Use the **Web** tab on the Format Picture window to assign alternate text to images in Microsoft Word source documents.

The following procedure provides an example of how to assign alternate text to images in Microsoft Word source documents using Microsoft Word 2003. Steps for assigning alternate text to images in Microsoft Word may be different in other versions of Microsoft Word.



## To assign alternate text to an image in a Microsoft Word source document

1. In your Microsoft Word source document, locate the image for which you want to specify image scaling.
2. Right-click the image, and then click **Format Picture** or **Format Object**.
3. On the **Web** tab, in the **Alternative text** field, type the alternate text you want to specify for the image.
4. Click **OK**.
5. Save your Microsoft Word source document.
6. Generate output for your project. For more information, see "Generating Output".
7. Verify ePublisher assigned the alternate text you specified to the image when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the image to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the alternate text you specified is included in the `alt` tag for the image.

## Assigning Alternate Text to Image Maps in Word

Use the **Web** tab on the Format Text Box window to assign alternate text to areas of an image map in Microsoft Word source documents.

The following procedure provides an example of how to assign alternate text to an image map in Microsoft Word source documents using Microsoft Word 2003. Steps for assigning alternate text to an image map in Microsoft Word may be different in other versions of Microsoft Word.

## To assign alternate text to an image map in a Microsoft Word source document

1. In your Microsoft Word source document, locate the image map for which you want to specify alternate text.
2. For each clickable area of the image map, complete the following steps:
  - a. Right-click the text box that defines a clickable region for the image map.
  - b. On the **Web** tab, in the **Alternative text** field, type the alternate text you want to specify for the image.
  - c. Click **OK**.
3. Save your Microsoft Word source document.
4. Verify ePublisher assigned the alternate text you specified to each area of the image map when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the image map to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the alternate text you specified is included in the `alt` tag for each area of the image map.

## Assigning Long Descriptions to Images in Word

This section explains how to create accessible images in your generated output by assigning long descriptions to images.

### Image Long Descriptions

The Web Content Accessibility Guidelines and Section 508 guidelines require you to include long descriptions for each image in an HTML document. You can use the `longdesc` attribute and a long descriptions stored in an external `.txt` file to assign a long description to an image. When you use this approach, the long descriptions are referenced in the HTML `<img>` tag in the `longdesc` attribute as shown in the following example:

```

```

The `longdesc` attribute in the `<img>` tag provides a link to a separate page where a long description is available. The link is invisible to sighted users, but when a conformant screen reader application reads the `longdesc` attribute, it loads the file referenced in the `longdesc` attribute and reads it. In the previous example, the screen reader would load and read the `mission.txt` file.

ePublisher provides the following options for assigning long descriptions to images:

- You can use the ImageLongDescText marker to assign a long description to an image. With this method, you assign a long description to an image using a description you include in a marker you insert into your source document. For more information, see “Specifying Long Descriptions for Images in Word”.
- You can use the ImageLongDescByRef marker to assign a long description to an image by referencing a long description saved in an external text (`.txt`) file. With this method, you specify the path to the external text file in a marker. For more information, see “Using Text in External Files to Assign Long Descriptions to Images in Word”.

If you assign long descriptions to some, but not all of your images, you can use the ImageLongDescNotReq marker. Use this marker when you use accessibility reports to verify that all images have long description but you have certain images in your source document that do not require a long description. For more information, see “Excluding Images from Accessibility Report Checks in Word”.

Although using the `longdesc` attribute is recommended in the Web Content Accessibility Guidelines and in 508 guidelines, older screen readers and many current browsers do not support this attribute and few online content developers use this attribute. As a result, the `longdesc` attributed benefits a only a small number of users. Only users who use modern screen readers can access the `longdesc` attribute easily. Older screen readers did not support this attribute. In addition, even users who use the latest version of screen reader may be unfamiliar with the `longdesc` attribute and may not know how to access long descriptions using their screen reader because the `longdesc` attribute is used so infrequently in online content.

If you use the ImageLongDescText marker to assign long descriptions to images, as an interim solution ePublisher allows you to display a D link immediately after the image. The D link is an upper case letter D link that directs users to another page that contains the text you specified in the ImageLongDescText marker. Although a D link is not required for accessible Web pages, it can be used in addition to the `longdesc` attribute. The D link technique works in all browsers, but it is less elegant than using the `longdesc` attribute. Some users may be confused when they see a D link on the page, while other users will ignore the D link.

If you want to use D links in addition to the `longdesc` attribute when you generate output, your Stationery must have the D link option enabled. If you have permissions to modify target settings in ePublisher, you can enable the D

link option setting in an project. For more information about enabling the D link option in an project, see "Specifying Accessibility Settings". For more information about permissions required to modify target settings using ePublisher Express, see "Working with Target Settings".

## Specifying Long Descriptions for Images in Word

To assign a long description to an image, your Stationery and template must have the ImageLongDescText marker type configured. Your output format must also support this feature.

When you use the ImageLongDescText marker to assign long descriptions to images, ePublisher generates an external text file that contains the long description you specify. When a conformant screen reader application reads the generated page, it loads the `.txt` file referenced in the `longdesc` attribute on the page and reads the file.

The following procedure provides an example of how to specify long descriptions for images in Microsoft Word source documents using Microsoft Word 2003. Steps for specify long descriptions for images in Microsoft Word may be different in other versions of Microsoft Word.

## To assign a long description to an image using marker text in a Microsoft Word source document

1. In your Microsoft Word source document, locate the image to which you want to assign a long description.
2. Right-click the image, and then click **Format Picture** or **Format Object**.
3. Change the layout setting of the image to **Top and Bottom** by completing the following steps:

**Note:** By default when you insert images into Microsoft Word, Microsoft Word inserts the image using the **Inline with text layout** setting. In order to specify the image scale for image output files, you must group the image and the text box that contains the ImageLongDescText marker. However, you cannot group images using the **In line with text** layout setting in Microsoft Word. To work around this known Microsoft Word issue, if you have an image that uses an **In line with text** layout setting, use the **Top and Bottom** layout setting for the image while you insert the ImageLongDescText marker, and then reapply the **In line with text** layout setting after you group the image and the ImageLongDescText marker.

- a. On the **Layout** tab, click **Advanced**.
  - b. On the **Text Wrapping** tab, click **Top and Bottom**.
  - c. Click **OK**, and then click **OK** again to close the window.
4. Select your image.
  5. On the **Insert** menu, click **Text Box**, and then click to the right of your image. Microsoft Word inserts a text box.
  6. Insert your cursor into the text box, and then complete the following steps:
    - a. On the **WebWorks** menu, click **Markers**.
    - b. In the **Markers** field, select **ImageLongDescText** from the list of markers.
    - c. In the **Value** field, type the long description you want to specify for the image.
    - d. Click **OK**. ePublisher inserts the ImageLongDescText marker into the text box.
    - e. Select the text box.

- f.** Right-click the selected text box, and then click **Format Text Box**.
  - g.** On the **Colors and Lines** tab, in the **Fill** area, in the **Color** field, select **No Fill**.
  - h.** In the **Line** area, in the **Color** field, select **No Line**.
  - i.** Click **OK**.
- 7.** Drag and drop the text box onto the image.
- 8.** Select the text box and the image.
- 9.** Right-click the selected text box and image, and then click **Grouping > Group**.

**Note:** When you select **Group**, the location of the image in your Microsoft Word source document may change in relation to the text in your source document. For example, the image may move up or down in your Microsoft Word source document. This is known Microsoft Word behavior. You may need to scroll up or down in your source document to the new location of the image to find the image.

- 10. If your image previously used the *In line with text layout setting for the image*, reassign this style to your image by completing the following steps:**

- a.** Right-click **only** the image, and then click **Format Object**.

**Note:** You must ensure you right-click **only** the image, and not on the text box or the grouped text box and image. If you right-click on the text box or the grouped text box and image, Microsoft Word does not display the **Format Object** menu option on the context menu.

- b.** On the **Layout** tab, click **In line with text**.
  - c.** Click **OK**, and then click **OK** again to close the window.
- 11.** Save your Microsoft Word source document.
- 12.** Generate output for your project. For more information, see “Generating Output”.
- 13.** Verify ePublisher assigned the long description to the image by completing the following steps:
  - a.** On the **View** menu, click **Output Directory**.

- b. In the *TargetName*\images folder, verify that ePublisher created a .txt file that contains the long description you specified in the ImageLongDescText marker, where *TargetName* is the name of your target.

For example, if you specified a long description for *ImageName*.png, verify that ePublisher created an *ImageName*.txt file in the images folder, where *ImageName* is the name of the image to which you assigned a long description.

- c. In the *TargetName*\ProjectName folder, open the page that contains the image to which you assigned the long description in Notepad and verify that the longdesc attribute references the *ImageName*.txt file ePublisher created for the image, where *TargetName* is the name of your target, *ProjectName* is the name of your project, and *ImageName* is the name of the image to which you assigned a long description.
- d. ***If you used the ImageLongDescText marker and the Stationery designer configured your Stationery to support D links***, open the page in a browser, verify that the D link displays in the browser, and then click the D link and verify that a page opens that displays the long description that you specified in the ImageLongDescText marker.

## Using Text in External Files to Assign Long Descriptions to Images in Word

Use the ImageLongDescByRef marker to assign long descriptions to images using text in external files. To assign a long description to an image, your Stationery and template must have the ImageLongDescText marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to use text in external files to assign long descriptions to images in Microsoft Word source documents using Microsoft Word 2003. Steps for using text in external files to assign long descriptions to images in Microsoft Word may be different in other versions of Microsoft Word.

## To assign a long description to an image using marker text in a Microsoft Word source document

1. Create a `.txt` file that contains each image long description.
2. Place each image long description text file in a folder in the `ProjectName\Formats\TargetName\Files` folder for your project, where *ProjectName* is the name of your ePublisher project and *TargetName* is the name of your target.

For example, place the each image long description in the following location:

`ProjectName\Formats\TargetName\Files  
\longdescriptions\imagelongdescription.txt`

where *ProjectName* is the name of your ePublisher project, *TargetName* is the name of your target, *longdescriptions* is the name of the folder where you placed the image long description, and *imagelongdescription* is the name of the `.txt` file that contains the image long description.

3. In your Microsoft Word source document, locate the image to which you want to assign a long description.
4. Right-click the image, and then click **Format Picture** or **Format Object**.
5. Change the layout setting of the image to **Top and Bottom** by completing the following steps:

**Note:** By default when you insert images into Microsoft Word, Microsoft Word inserts the image using the **Inline with text layout** setting. In order to specify the image scale for image output files, you must group the image and the text box that contains the ImageLongDescByRef marker. However, you cannot group images using the **In line with text** layout setting in Microsoft Word. To work around this known Microsoft Word issue, if you have an image that uses an **In line with text** layout setting, use the **Top and Bottom** layout setting for the image while you insert the ImageLongDescText marker, and then reapply the **In line with text** layout setting after you group the image and the ImageLongDescText marker.

- a. On the **Layout** tab, click **Advanced**.
  - b. On the **Text Wrapping** tab, click **Top and Bottom**.
  - c. Click **OK**, and then click **OK** again to close the window.
6. Select your image.



7. On the **Insert** menu, click **Text Box**, and then click to the right of your image. Microsoft Word inserts a text box.
8. Insert your cursor into the text box, and then complete the following steps:
  - a. On the **WebWorks** menu, click **Markers**.
  - b. In the **Markers** field, select ImageLongDescByRef from the list of markers.
  - c. In the **Value** field, type the path to the `.txt` file that contains the long description you want to assign to the image.

For example, type:

```
./longdescriptions/imagelongdescription.txt
```

where *longdescriptions* is the name of the folder where you placed the image long description, and *imagelongdescription* is the name of the `.txt` file that contains the image long description.
  - d. Click **OK**. ePublisher inserts the ImageLongDescText marker into the text box.
  - e. Select the text box.
  - f. Right-click the selected text box, and then click **Format Text Box**.
  - g. On the **Colors and Lines** tab, in the **Fill** area, in the **Color** field, select **No Fill**.
  - h. In the **Line** area, in the **Color** field, select **No Line**.
  - i. Click **OK**.
9. Drag and drop the text box onto the image.
10. Select the text box and the image.
11. Right-click the selected text box and image, and then click **Grouping > Group**.

**Note:** When you select **Group**, the location of the image in your Microsoft Word source document may change in relation to the text in your source document. For example, the image may move up or down in your Microsoft Word source document. This is known Microsoft Word behavior. You may need to scroll up or down in your source document to the new location of the image to find the image.

**12. If your image previously used the *In line with text layout setting for the image*, reassign this style to your image by completing the following steps:**

- a.** Right-click **only** the image, and then click **Format Object**.

**Note:** You must ensure you right-click **only** the image, and not on the text box or the grouped text box and image. If you right-click on the text box or the grouped text box and image, Microsoft Word does not display the **Format Object** menu option on the context menu.

- b.** On the **Layout** tab, click **In line with text**.

- c.** Click **OK**, and then click **OK** again to close the window.

**13.** Save your Microsoft Word source document.

**14.** Generate output for your project. For more information, see “Generating Output”.

**15.** In Output Explorer, verify ePublisher assigned the long description to the image using the long description in the external file when it generated output by completing the following steps:

- a.** On the **View** menu, click **Output Directory**.

- b.** In the *TargetName\ProjectName* folder, open the page that contains the image to which you assigned the long description using an external file in Notepad and verify that the `longdesc` attribute references the external text file that contains the long description for the image, where *TargetName* is the name of your target, and *ProjectName* is the name of your project.

## Excluding Images from Accessibility Report Checks in Word

In some instances, alternate text is sufficient for an image, and assigning a long description to an image in addition to alternate text would be redundant. However, you may have configured Accessibility reports to check for images without long descriptions and notify you when an image does not have a long description.

In this scenario, while you want an Accessibility report to notify you when you have an image without a long description, you do not want to be notified when you deliberately did not assign a long description to an image because assigning a both a long description and alternative text would be redundant. To address this issue, you can use the ImageLongDescNotReq marker to exclude an image that deliberately does not have a long description from validation when you

generate Accessibility reports. For more information about Accessibility reports and configuring and generating Accessibility reports, see "Accessibility Reports", "Configuring Reports", and "Generating Reports".

To exclude images without long descriptions from Accessibility reports, your Stationery and template must have the ImageLongDescNotReq marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to exclude images without long descriptions from Accessibility report checks in Microsoft Word source documents using Microsoft Word 2003. Steps for excluding images without long descriptions from Accessibility report checks in Microsoft Word may be different in other versions of Microsoft Word.

## To exclude an image without a long description from Accessibility report checks in a Microsoft Word source document

1. In your Microsoft Word source document, locate the image without a long description that you want to exclude from an Accessibility report check.
2. Change the layout setting of the image to **Top and Bottom** by completing the following steps:

**Note:** By default when you insert images into Microsoft Word, Microsoft Word inserts the image using the **Inline with text layout** setting. In order to specify the image scale for image output files, you must group the image and the text box that contains the ImageLongDescNotReq marker. However, you cannot group images using the **In line with text** layout setting in Microsoft Word. To work around this known Microsoft Word issue, if you have an image that uses an **In line with text** layout setting, use the **Top and Bottom** layout setting for the image while you insert the ImageLongDescNotReq marker, and then reapply the **In line with text** layout setting after you group the image and the ImageLongDescNotReq marker.

- a. On the **Layout** tab, click **Advanced**.
  - b. On the **Text Wrapping** tab, click **Top and Bottom**.
  - c. Click **OK**, and then click **OK** again to close the window.
3. Select your image.
  4. On the **Insert** menu, click **Text Box**, and then click to the right of your image. Microsoft Word inserts a text box.
  5. Insert your cursor into the text box, and then complete the following steps:
    - a. On the **WebWorks** menu, click **Markers**.
    - b. In the **Markers** field, select ImageLongDescNotReq from the list of markers.
    - c. In the **Value** field, do not enter any text. You do not need to enter any text in this field when you insert a ImageLongDescNotReq marker.
    - d. Click **OK**. ePublisher inserts the ImageLongDescText marker into the text box.
    - e. Select the text box.
    - f. Right-click the selected text box, and then click **Format Text Box**.

- g. On the **Colors and Lines** tab, in the **Fill** area, in the **Color** field, select **No Fill**.
  - h. In the **Line** area, in the **Color** field, select **No Line**.
  - i. Click **OK**.
- 6. Drag and drop the text box onto the image.
  - 7. Select the text box and the image.
  - 8. Right-click the selected text box and image, and then click **Grouping > Group**.

**Note:** When you select **Group**, the location of the image in your Microsoft Word source document may change in relation to the text in your source document. For example, the image may move up or down in your Microsoft Word source document. This is known Microsoft Word behavior. You may need to scroll up or down in your source document to the new location of the image to find the image.

- 9. ***If your image previously used the In line with text layout setting for the image***, reassign this style to your image by completing the following steps:

- a. Right-click **only** the image, and then click **Format Object**.

**Note:** You must ensure you right-click **only** the image, and not on the text box or the grouped text box and image. If you right-click on the text box or the grouped text box and image, Microsoft Word does not display the **Format Object** menu option on the context menu.

- b. On the **Layout** tab, click **In line with text**.
  - c. Click **OK**, and then click **OK** again to close the window.
- 10. Save your Microsoft Word source document.
  - 11. Generate output for your project. For more information, see "Generating Output".
  - 12. Generate an Accessibility report and confirm that ePublisher did not generate an `Image is missing a long description` message for the image. For more information about generating Accessibility reports and Accessibility report messages, see "Generating Reports" and "Accessibility Report Messages".

# Assigning Alternate Text (Summaries) to Tables in Word

Tables, just like images, are a way to visually display information. Although tables typically contain text, the purpose of the table is often not evident from text alone. The organization and display of the table may contain information that is not evident to assistive technologies. However, through the use of table summaries, assistive technologies can convey useful information to users about tables. The Web Content Accessibility Guidelines recommend that you provide summary text for each table in an HTML document. Table alternate text, or table summaries, provide users with information about what type of information the table contains.

You can create accessible tables by typing the table summary into a TableSummary marker. When ePublisher generates content, ePublisher puts the table summary you specify into the table in the `summary` attribute.

To assign alternate text to tables, your Stationery and template must have the TableSummary marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to assign alternate text to tables in Microsoft Word source documents using Microsoft Word 2003. Steps for assigning alternate text to tables in Microsoft Word may be different in other versions of Microsoft Word.

## To assign table summaries in a Microsoft Word source document

1. In your Microsoft Word source document, locate the table to which you want to assign a table summary.
2. Insert your cursor in front of the table.
3. On the **WebWorks** menu, click **Markers**.
4. In the **Markers** field, select **TableSummary** from the list of markers.
5. In the **Value** field, type the alternate text for the table.
6. Click **OK**. ePublisher inserts the TableSummary marker into the table.
7. Insert the marker into the table caption by clicking **OK**.
8. Save your Microsoft Word source document.
9. Generate output for your project. For more information, see "Generating Output".
10. Verify ePublisher assigned the table summary you specified to the table when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the table to which you assigned a table summary in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the table summary you specified is included in the `summary` attribute for the table.

## Excluding Tables from Accessibility Report Checks in Word

Tables used specifically for layout may not need a table summary. For example, if you use a table for layout, you probably would not assign a table summary to the table. However, you may have configured Accessibility reports to check for tables without table summaries and notify you when a table does not have a table summary.

In this scenario, while you want an Accessibility report to notify you when you have a table without a table summary, you do not want to be notified when you deliberately did not assign a table summary to a table because a table summary is not required. To address this issue, you can use the TableSummaryNotReq marker to exclude a table that deliberately does not have a table summary from validation

when you generate Accessibility reports. For more information about Accessibility reports and configuring and generating Accessibility reports, see "Accessibility Reports", "Configuring Reports", and "Generating Reports".

To exclude tables from Accessibility report checks, your Stationery must have the TableSummaryNotReq marker type configured. Your output format must also support this feature.

The following procedure provides an example of how to exclude tables without table summaries from Accessibility report checks in Microsoft Word source documents using Microsoft Word 2003. Steps for excluding tables without table summaries from Accessibility report checks in Microsoft Word may be different in other versions of Microsoft Word.



## To exclude a table with a table summary from Accessibility report checks in a Microsoft Word source document

1. In your Microsoft Word source document, locate the table without a table summary that you want to exclude from an Accessibility report check.
2. Insert your cursor in front of the table.
3. On the **WebWorks** menu, click **Markers**.
4. In the **Markers** field, select **TableSummaryNotReq** from the list of markers.
5. In the **Value** field, do not enter any text. You do not need to enter any text in this field when you insert a TableSummaryNotReq marker.
6. Click **OK**. ePublisher inserts the TableSummaryNotReq marker into the table.
7. Save your Microsoft Word source document.
8. Generate output for your project. For more information, see "Generating Output".
9. Generate the Accessibility report and confirm that ePublisher did not generate an `Table is missing a table summary` message for the table. For more information about generating Accessibility reports and Accessibility report messages, see "Generating Reports" and "Accessibility Report Messages".

## Assigning Alternate Text to Abbreviations in Word

Abbreviations are often used in written communication. Using an Abbreviation character style and an AbbreviationTitle marker, you can specify alternate text for abbreviations. For example, if your source document includes an abbreviation such as SS#, you can specify Social Security Number as alternate text for the abbreviation. When you use an AbbreviationTitle marker and Abbreviation character style to specify alternate text for an abbreviation, ePublisher adds the abbreviation alternate text you specify to the `title` attribute of the `abbr` tag in the output.

Following is an example of the HTML code produced when you specify Social Security Number as alternate text for SS#.

```
<th>First name</th>
<th><abbr title="Social Security Number">SS#</abbr></th>
```

To assign alternate text to abbreviations, your Stationery and template must have the following items configured:

- Abbreviation character style

- AbbreviationTitle marker type

Your output format must also support this feature.

The following procedure provides an example of how to specify alternate text for abbreviations in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying alternate text for abbreviations in Microsoft Word may be different in other versions of Microsoft Word.

## To specify alternate text for an abbreviation in a Microsoft Word source document

1. In your Microsoft Word source document, locate the abbreviation for which you want to specify alternate text.
2. Apply the AbbreviationTitle character style to the abbreviation text.
3. Insert your cursor anywhere inside the abbreviation.
4. On the **WebWorks** menu, click **Markers**.
5. In the **Markers** field, select **AbbreviationTitle** from the list of markers.
6. In the **Value** field, type the abbreviation alternate text.
7. Click **OK**. ePublisher inserts the AbbreviationTitle marker into the abbreviation.
8. Save your Microsoft Word source document.
9. Generate output for your project. For more information, see "Generating Output".
10. Verify ePublisher assigned the abbreviation alternate text you specified when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the abbreviation to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the alternate text you specified for the abbreviation is included in the `abbr` tag in the `title` attribute.

## Assigning Alternate Text to Acronyms in Word

Acronyms are often used in written communication. Using an Acronym character style and an AcronymTitle marker, you can specify alternate text for acronyms. For example, if your document includes an acronym like NATO you can specify North Atlantic Treaty Organization as alternate text for the acronym. When you use an AcronymTitle marker and an Acronym character style to specify alternate text for an acronym, ePublisher adds the acronym alternate text you specify to the `title` attribute of the `acronym` tag in the output.

Following is an example of the HTML code produced when you specify North Atlantic Treaty Organization as alternate text for NATO.

```
<p><acronym title="North Atlantic Treaty Organization">NATO</acronym>  
is a military alliance.</p>
```

To assign alternate text to acronyms, your Stationery and template must have the following items configured:

- Acronym character style
- AcronymTitle marker type

Your output format must also support this feature.

The following procedure provides an example of how to specify alternate text for acronyms in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying alternate text for acronyms in Microsoft Word may be different in other versions of Microsoft Word.

#### **To specify alternate text for an acronym in a Microsoft Word source document**

1. In your Microsoft Word source document, locate the acronym for which you want to specify alternate text.
2. Apply the AcronymTitle character style to the abbreviation text.
3. Insert your cursor anywhere inside the abbreviation.
4. On the **WebWorks** menu, click **Markers**.
5. In the **Markers** field, select **AcronymTitle** from the list of markers.
6. In the **Value** field, type the acronym alternate text.
7. Click **OK**. ePublisher inserts the AcronymTitle marker into the abbreviation.
8. Save your Microsoft Word source document.
9. Generate output for your project. For more information, see "Generating Output".
10. Verify ePublisher assigned the acronym alternate text you specified when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.

- b. In the *TargetName* folder, open the page that has the acronym to which you assigned alternate text in Notepad, where *TargetName* is the name of your target.
- c. Verify that the alternate text you specified for the acronym is included in the `acronym` tag in the `title` attribute.

## Providing Citations for Quotes in Word

A **citation** is a reference or footnote to a book, article, or other material that specifies the source from which a quotation was borrowed. A citation contains all the information necessary to identify and locate the work. Using a Citation character style and the Citation marker, you can specify citations for quotes that enable users to go to a Web site that contains additional information about the quote.

Following is an example of the HTML code produced when you specify a citation for a quote.

```
<blockquote cite="http://shakespeare.mit.edu/lll/full.html">  
  <p>Remuneration! O! that's the Latin word for three farthings.  
  --- William Shakespeare (Love's Labor Lost).</p> </blockquote>
```

To provide citations for quotes, your Stationery and template must have the following items configured:

- Citation character style
- Citation marker type

Your output format must also support this feature.

The following procedure provides an example of how to specify citations for quotes in Microsoft Word source documents using Microsoft Word 2003. Steps for specifying citations for quotes in Microsoft Word may be different in other versions of Microsoft Word.

## To specify citations for quotes in a Microsoft Word source document

1. In your Microsoft Word source document, locate the quotation for which you want to specify a citation.
2. ***If the quotation is a phrase within a paragraph***, complete the following steps:
  - a. Apply the Citation character style to the quotation phrase.
  - b. Insert your cursor anywhere inside the quotation phrase.
3. ***If the quotation is a full paragraph***, insert your cursor into the paragraph.
  - c. On the **Special** menu, click **Marker**.
4. On the **WebWorks** menu, click **Markers**.
5. In the **Markers** field, select **Citation** from the list of markers.
6. In the **Value** field, type the URL for the citation.
7. Click **OK**. ePublisher inserts the Citation marker into the abbreviation.
8. Save your Microsoft Word source document.
9. Generate output for your project. For more information, see "Generating Output".
10. Verify ePublisher created the citation you specified when it generated output by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. In the *TargetName* folder, open the page that has the quotation for which you specified a quotation in Notepad, where *TargetName* is the name of your target.
  - c. Verify that the citation you specified for the quotation is included in the `cite` attribute.

## Troubleshooting Word issues

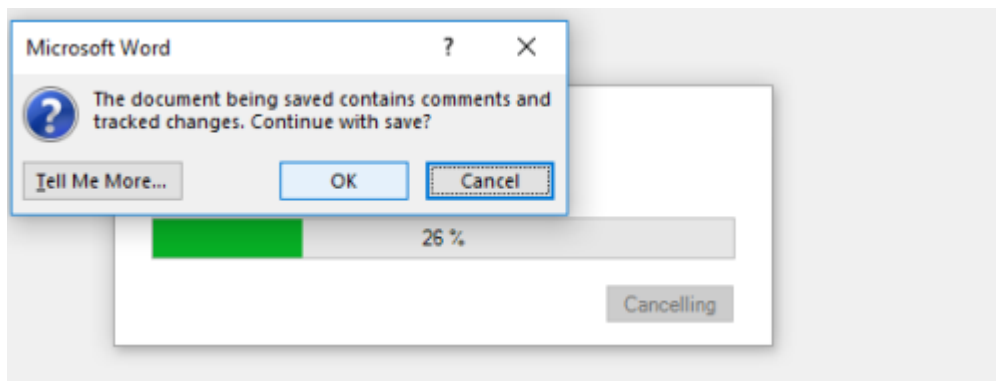
Occasionally there might be issues with the source documents you are using. Below is a list linking to the wiki solutions website that will help you troubleshoot each one:

<b>Issue</b>	<b>For more information, see to...</b>
If you are seeing hidden text in the output	<a href="#">Hidden text in output</a>
If you are seeing inconsistent bullets in output	<a href="#">Inconsistent bullet sizes</a>
If you are seeing an "infinite" number of transit menus in Word 2003	<a href="#">Transit Menu displays "infinite" menu syndrome</a>
If you are using relative images	<a href="#">Relative Images</a>
If you are using master docs	<a href="#">Using Microsoft Word Master Docs</a>
If conversions hang and your documents track changes or comments	<a href="#">Word warning dialogs that interrupt conversions</a>

## Word warning dialogs that interrupt conversions

Conversions in ePublisher being stuck because of warning dialogs or just hanging when you are using Word source documents and you have comments or tracking the changes, might be cause by a configuration in your Word application.

A warning dialog you might see during a conversion in ePublisher should look similar to the following figure.

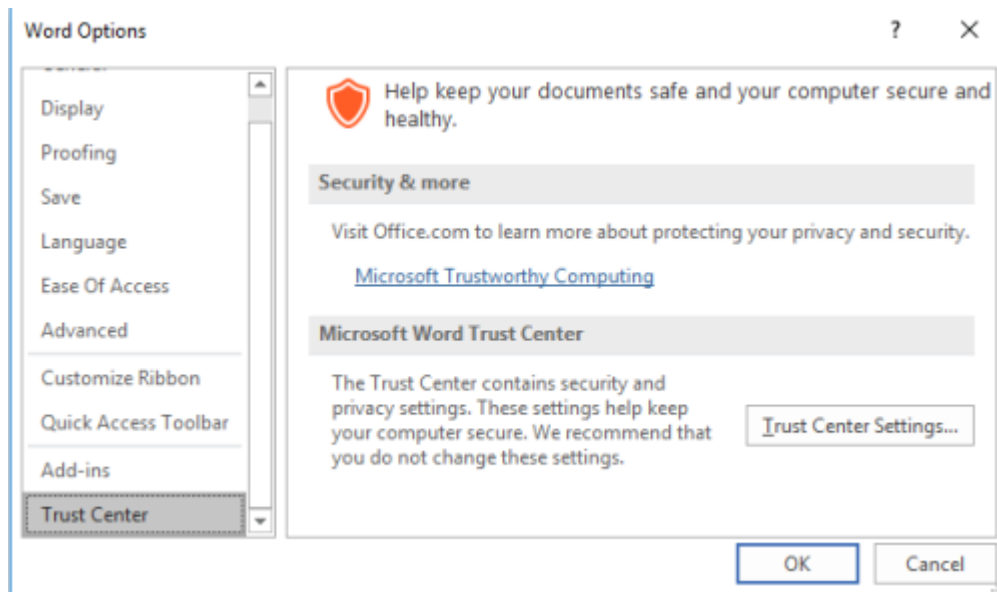


The following procedure provides an example of how to disable the "Warn before printing, saving or sending a file that contains tracked changes or comments" option in Microsoft Word source documents using Microsoft Word 2016.

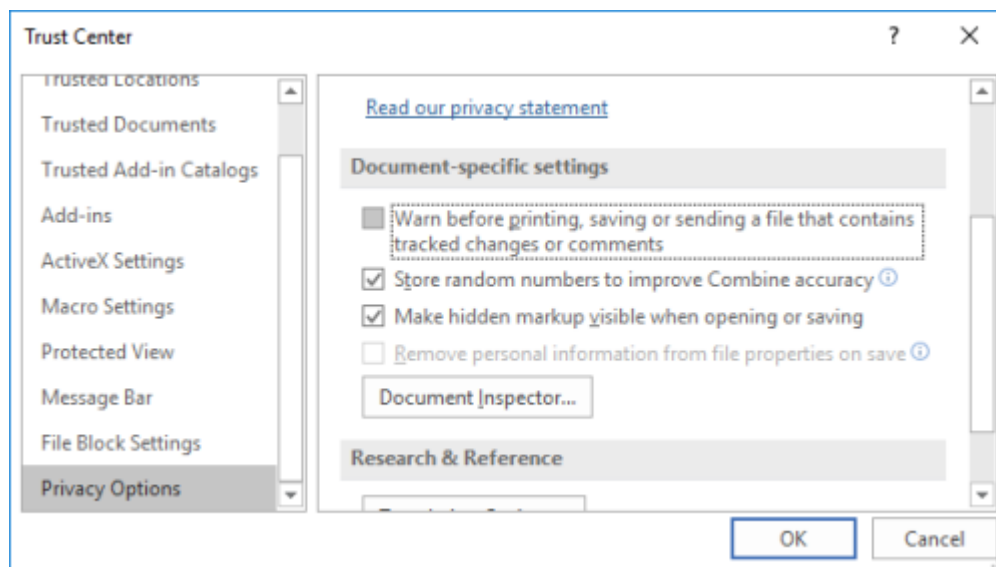


**To disable the “Warn before printing, saving or sending a file that contains tracked changes or comments” in a Microsoft Word source document**

1. Open a Microsoft Word document, or even a Blank document.
2. On the **Word** menu, click **File** and then **Options**.
3. In the window that pops out in the left side menu click on **Trust Center**, and on the right side select **Trust Center Settings**. You should see a window similar to the following figure.



4. After selecting **Trust Center Settings** you'll see a new window, click on the left side on **Privacy Options** and on the right panel uncheck **Warn before printing, saving or sending a file that contains tracked changes or comments**. You should see a window similar to the following figure.



5. And finally click **OK**.

# DITA - XML

- [DITA Usage Standards](#)
- [DITA Support](#)
- [Using Ditaaval files in DITA](#)
- [Using Passthrough outputclass in DITA](#)
- [Embedding a Video in DITA Source Documents](#)
- [Creating Context-Sensitive Help in DITA Source Documents](#)
- [Creating Hyperlinks in DITA Source Documents](#)
- [Creating Popups in DITA Source Documents](#)
- [Creating Related Topics in DITA Source Documents](#)
- [Creating See Also Links in DITA Source Documents](#)
- [Using the data element](#)
- [Assigning Custom Page Styles to Pages in DITA Source Documents](#)
- [Using Custom Graphic Styles for Images in DITA Source Documents](#)
- [Customizing TOC Entry in DITA](#)
- [Customizing Table of Contents Icons for Topics in DITA Source Documents Using Legacy Outputs](#)
- [Using markopen and markclose](#)
- [Troubleshooting DITA issues](#)

Before you can generate output using DITA source documents, you need to prepare your DITA source documents for output generation. This section explains how to prepare your DITA source files for output generation.

## DITA Usage Standards

DITA (Darwin Information Typing Architecture) is an XML-based format for creating and publishing technical content. DITA leverages the strengths of XML and provides a standard set of element definitions used to create technical content. Within the topic types based on the standard topic definition, DITA specifies the information elements used to define the content, such as the title, paragraph, table, and list elements.

This section describes the design usage considerations for DITA source documents. By reviewing how ePublisher processes DITA source documents, you can streamline single-sourcing processes and reduce your production and maintenance costs. This section does not describe all DITA details, but it focuses on the design and usage considerations related to ePublisher.

## DITA Standards for Single-Sourcing

ePublisher accepts ditamaps as input source documents. The ditamap identifies the structure and order of the XML files and the DTD. The DTD defines the classes for each element, such as topic/topic and task/task. ePublisher uses these classes to process the content. ePublisher also allows you to add individual XML files as source documents to a project. However, if you publish individual XML files without using a ditamap, links will not be generated in the output.

**Note:** Publishing individual XML files is only supported when using the DITA-OT version 1.8 and above.

## Mapping DITA Classes to ePubublisher Styles

The `default.wwconfig` file maps classes to styles in ePubublisher, defining both the style type and the style name. You can override the `default.wwconfig` file to specify your own style names and types. The override file does not need to be comprehensive. Your override file can build on the default file. When the same match is defined in both the customized override file and the default file, the match in the customized file overrides the match in default file.

You can override the `default.wwconfig` file for all output formats, a specific output format, or a specific target. The `default.wwconfig` file can also emit WIF information, such as `ww` content for bulleted and numbered list items. For more information, see the default `default.wwconfig` file in the following folder:

```
<ePublisher Installation Location>\ePublisher Designer\Adapters\xml  
\scripts\dita
```

For more information about override files and locations, see “Stationery, Projects, and Overrides”.

## To specify your own class mappings with an override file

1. ***If you want to override the class mappings for all output formats,*** complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
  - b. Create the `Formats\Adapters\xml\scripts\dita` folder in your project folder.
2. ***If you want to override the class mappings for one output format,*** complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
  - b. Create the `Formats\Adapters\formattype\xml\scripts\dita` folder in your project folder, where *formattype* is the name of the output format you want to override, such as `WebWorks Reverb 2.0`.
3. ***If you want to override the class mappings for a target,*** complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Target Override Directory**.
  - b. Create the `Targets\targetname\Adapters\xml\scripts\dita` folder in your project folder, where *targetname* is the name of the target you want to override.
4. Copy the `default.wwconfig` file from the following folder to the override folder you created within your project folder:  
`Program Files\WebWorks\ePublisher Designer\Adapters\xml\scripts\dita`
5. Open the `default.wwconfig` file you copied to your project override folder.
6. Remove any mappings you do not want to override.
7. Add the mappings you want to override, including a match statement that defines the style name and a match statement that defines the style type.
8. Save and close the `default.wwconfig` file you copied to your project override folder.

The `default.wwconfig` file provides two sections:

## **<Styles> section**

Provides the style match statements that map DITA classes to style names in ePublisher. Each style match statement, such as `<Style match=...`, defines the classes and parent classes to use for a match. When a class matches the style match statement, the XSL specifies the ePublisher style name to associate with that class.

## **<Types> section**

Provides the type match statements that map DITA classes to style types in ePublisher. Each type match statement, such as `<Type match=...`, defines the classes to use for a match. When a class matches the type match statement, the value attribute specifies the ePublisher style type for the style associated with that class.

# **Defining Online Features with DITA**

You assign formatting and features to styles in ePublisher. These style definitions are stored in your Stationery and mapped to DITA classes. For some online features, such as index entries, you use the standard DITA elements to implement the feature. For more information about creating Stationery and implementing online features, see “Designing, Deploying, and Managing Stationery”.

Review the following considerations to understand how to implement each online feature using DITA elements and the Stationery options within ePublisher:

- For index entries, use the standard DITA tag.
- For related topics, assign related topics options in ePublisher to paragraph styles for related-links elements, such as the Related Task and Related Concept styles.
- For expand/collapse sections, assign dropdown options in ePublisher to the appropriate paragraph styles. You cannot use a marker to identify the end of the expand/collapse section, so you need to use a paragraph style to identify the end.
- For conditions, use attributes and ditaval file filtering, for more information, See “Using Ditaval files in DITA”.
- For variables, use conref and ditaval file filtering.
- For popup windows, use the xref element for the link and define paragraph styles with the popup options in ePublisher.
- For specifying file names, use the othermeta element to define the marker with the name for the file.

- For specifying a topic alias for context-sensitive help links, use the othermeta element to define the marker with the value of the topic ID for that topic.
- For meta tag keywords, use the othermeta element to define the marker with the keywords you want to include in the meta tag in the generated output.
- For opening a topic in a custom window, use the othermeta element to define the marker with the name of the window in which to open the topic.
- For a custom TOC icon, use the othermeta element to define the marker with the name of the TOC icon to use for the generated topic.
- For See Also links, use related topics, which are available in more output formats. See Also link support is being reviewed for future enhancements.
- For accessibility features, such as image alternate text and long descriptions, use standard DITA elements and attributes.

## Configuring DITA Open Toolkit Version

In order to support DITA specialization and customization it is sometimes necessary to specify a specific version of the DITA Open Toolkit (OTK). In ePublisher, you can easily change which version of the OTK is used to preprocess all of your content on a project by project basis. You can even configure your stationery to use a specific version.

## To change your project's OTK version

1. In your Stationery design project, on the **Project** menu, click **Project Settings...**
2. Expand the **XML Input** tab and locate the row labeled: `DITA Open Toolkit version`.
3. Choose from one of the available versions and then select the **OK** button.

**Note:** Usually, it is fine to just use the latest version, however, it may be necessary to use an older version if you have specializations or other customizations that have not yet been migrated.

## Customizing the DITA DTD

You can associate an instance of the DITA-OT with your Stationery design project to apply and track DITA specialization through the Stationery.



## To customize the DITA DTD and apply the DITA specialization

1. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see "Stationery, Projects, and Overrides".
2. Create the `Formats\Helpers\dita-ot-<version>` folder in your *projectname* folder, where *projectname* is the name of your Stationery design project.
3. Copy the contents of the following folder to the override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Helpers\dita-ot-  
<version>
```

4. Open the `dita-ot-<version>` folder you copied to your project override folder and apply the DTD changes you need for your DITA specialization. Typically this means running the `ant integrate` task to install your DITA-OT plug-ins.
5. Save and close the modified files.
6. Regenerate your project to review the changes.

## DITA Specialization

Structure is mapped to style in dita, for more information, please refer to the following website:

<http://wiki.webworks.com/HelpCenter/Tips/DITA/All Versions/DITA Configuration>

## DITA Support

ePublisher supports all current OASIS DITA standards, which includes versions 1.1, 1.2, and 1.3. For information on the complete set of DITA 1.3 topic types, elements, attributes, and other specifications you may want to refer to:

[https://www.webworks.com/Documentation/DITA\\_1.3\\_Language\\_Reference/](https://www.webworks.com/Documentation/DITA_1.3_Language_Reference/).

## Keyref elements

You can use the "keyref" feature in DITA as an indirect addressing mechanism. Instead of linking directly to topics or maps, these can be given a symbolic name (key attribute) that points to a topic file path (href attribute). References to the topics are made using a key reference (keyref attribute). If the topic is relocated, the path needs to be updated only in the map where it is defined. All other references will automatically pick up the new location.

# Conref extensions

DITA uses of the conref element.

- Conref Range - Allows a single element to reference a range of elements.
- Conref Push - Most commonly used when your component adds to an existing component.
- Conkeyref - Allows you to use a key in the `conref` attribute so you can more easily change the target of the `conref`.

## Using Ditaval files in DITA

For conditional text, DITA uses filtering to determine the user or audience that will be viewing the help content. Using a file called a "ditaval" file, you can control the filtering of the content that will be generated as output. You can place ditaval files next to ditamap files using the name of the DITA map files that they are to be applied to similar to this:

`<ditamap name>.ditaval`

With ePublisher you have 3 additional ways to handle ditaval files. The ditaval definitions are looked for in the following order.

- Per Target: `Targets\<Target Name>\Adapters\xml\scripts\dita\default.ditaval`
- Per Format: `Formats\<Format Name>\Adapters\xml\scripts\dita\default.ditaval`
- Entire project: `Formats\Adapters\xml\scripts\dita\default.ditaval`

The following example shows an example of how filtering works with the ditaval markup:

```
<val>
  <prop att="audience" val="web" action="exclude" />
</val>
```

The attributes within prop determine the action and who sees the material that is defined in the output.

### action

include - Includes the content in output. This is the default behavior unless otherwise set.

exclude - Excludes the content from output (if all values in the particular attribute are excluded).

passthrough - Include the content in output, and preserve the attribute value as part of the output stream for further processing by a runtime engine, for example runtime filtering based on individual user settings

flag - include and flag the content on output (if the content has not been excluded).

## **att**

The attribute to be acted. This must be one of props, audience, platform, product, otherprops, or a specialization of props. If the att attribute is absent, then the prop element declares a default behavior for any conditional processing attribute.

## **val**

The value to be acted upon. If the val attribute is absent, then the prop element declares a default behavior for any value in the specified attribute.

# **Using Passthrough outputclass in DITA**

You may want to include some kind of HTML or Javascript in your output that would not be generated as normal output. For this example we are going to use a passthrough attribute on a paragraph tag in DITA. This will create a Passthrough paragraph style that you would switch to Passthrough in the paragraph options tab in ePublisher.

Below is an example of a Javascript alert in HTML using the CDATA attribute

```
<p outputclass="Passthrough">
<![CDATA[
<div><a href="alert('Yahoo!')">Click Me!</a></div>
]]>
</p>
```

Once you have entered in the paragraph that contains the passthrough attribute, re-scan the document and make sure that this passthrough paragraph has the "Passthrough" option enabled so that the HTML will not be processed.

Depending on the output, for example, browser-based (Dynamic HTML or WebWorks Help 5.0) versus PDF output, you may want to use ditaval filtering to ensure that only the web-based output is getting output. For this instance, you would want to have the following DITA markup

```
<p outputclass="Passthrough" audience="web">
<![CDATA[
<div><a href="alert('Yahoo!')">Click Me!</a></div>
```

```
]]>
</p>
```

For the PDF output, for example you would want to have the Target override:

```
Targets\PDF\Adapters\xml\scripts\dita
```

and the default.ditval information:

```
<val>
  <prop att="audience" val="web" action="exclude" />
</val>
```

For the Dynamic HTML output you would want to have the Target override:

```
Targets\Dynamic HTML\Adapters\xml\scripts\dita
```

and the default.ditval information:

```
<val>
  <prop att="audience" val="web" action="include" />
</val>
```

**Note:** Depending on what your Target name and input is like, the information may be different than what is listed above. For more information on Target overrides, See "Creating Target Overrides".

## Embedding a Video in DITA Source Documents

This section explains how you can embed videos in DITA source documents for ePubliher. **For embedded videos we only allow the mp4 and flv formats.**

### Embedding a Video file

**You can embed a video that you have on file into your DITA source document.**

## To embed a video from your file directory:

1. Create the an object tag for the video, for example: `<object> </object>`
2. Next you are going to want to specify certain attributes for your video object
  - If want to set a stylename, set the outputclass attribute to whatever stylename you want to give it.
  - Specify what width and height you want to give to your video in the width and height attributes
  - Set the data attribute to the path to the video file relative to the source file
  - For example: `<object outputclass="Foo" width="560" height="320" data="../../Video/small.mp4"> </object>`
3. Optional: if you want controls on your video include the following param tag inside your object tag: `<param name="controls" value="true">`

The following is a fully working code example of the following steps:

```
<object outputclass="Foo" width="560" height="320" data="../../Video/
small.mp4">
  <param name="allowFullScreen" value="true" />
  <param name="controls" value="true" />
</object>
```

## Linking to a Youtube Video

You can embed a Youtube video in your DITA source document.

## To embed a Youtube Video:

1. Create the an object tag for the video, for example: `<object> </object>`
2. Make sure that you are using the **Embed** URL and not the default URL provided in your browser when on the YouTube website. To obtain the proper URL, follow the instructions for obtaining a share link to embed in a website.
3. Next you are going to want to specify certain attributes for your video object
  - Specify what width and height you want to give to your video in the width and height attributes.
  - Set the data attribute to the link to the video file (optional).
  - For example: `<object width="560" height="320" data="https://www.youtube.com/embed/Fy1SB0ClS0k"> </object>`

**Note:** The proper URL should contain "embed" in the path.

4. Next you can set certain param tags for your video depending on your own specifications

**Note:** The `data` attribute on the `object` element is optional.

The following is a fully working code example of the following steps:

```
<object outputclass="Foo" width="560" height="320">
  <param name="movie" value="https://www.youtube.com/embed/
Fy1SB0ClS0k" />
  <param name="controls" value="true" />
</object>
```

# Creating Context-Sensitive Help in DITA Source Documents

This section explains how you can use ePublisher to create links to context-sensitive help content in DITA source documents

## Context-Sensitive Help

**Context-sensitive help** provides immediate assistance and information to users without requiring users to leave the context in which they are working. It helps answer questions like "What is this?" and "Why would I use this?", and provides information for a particular object and its context.

For example, in many applications, user interface controls such as windows and tabs have a help button. When users click on the help button, the application links

users to a help topic specific to the context of the window. Some applications also embedded context-sensitive help topics into the window itself as an HTML pane. The application relies on an identifier such as a topic ID or file name to identify the specific help topic to display.

There are several methods for creating context-sensitive help links. In addition, different use different mechanisms to support context-sensitive help links. For example, some, such as Microsoft HTML Help, create a map file using topic aliases. Applications then use the topic IDs in the map file to provide links to context-sensitive help topics from within the application. Other do not have a mapping mechanism. However, these may support creating links to context-sensitive help topics using file names.

## Map Files

Many application support the use of map files to deliver context-sensitive help. The topic IDs and map numbers are listed in a map file, which is a text file that typically has a `.h` extension. Applications can use the information in the map file to link users to the appropriate context-sensitive help topic.

**Note:** Some developers may use the term header file instead of map file.

There are some variations in the way context-sensitivity works depending on which supported ePublisher output format you use. For example, Microsoft HTML Help, Sun JavaHelp, and Oracle Help use map files.

**Note:** WebWorks Help, WebWorks Reverb, Dynamic HTML Help, and XML+XSL do not use map files.

When an application calls a context-sensitive help topic, it relies on the topic IDs and map numbers to identify the specific topic to display. Therefore, the topic IDs and map numbers must be embedded both in the application code and in the help system. If the topic IDs and map numbers do not match, the wrong topic (or no topic) displays when the user requests Help.

Following is a typical example of a Microsoft HTML Help map file:

```
#define IDH_WDWTYPE      1001
#define IDH_WDWENTER     1002
#define IDH_WDWCANCEL    1003
```

In this example, IDH\_WDWTYPE is a topic ID, and 1001 is the corresponding map number. These topic IDs and map numbers must be embedded in the software application and in your source documents.

Following is a typical example of a Sun JavaHelp and Oracle Help map file:

```
<mapID target="ch1_hm_999374" url="ch1.htm#999374">
<mapID target="ch2_hm_999640" url="ch2.htm#999640">
<mapID target="ch9_hm_999786" url="ch9.htm#999786">
```

In this example, `ch1_hm_99374` is a topic ID, and `ch1.htm#99374` is the target URL for the topic ID. These topic IDs must be embedded in the software application and in your source documents.

## Planning for Context-Sensitive Help

Creating context-sensitive help requires you to collaborate with application developers. Because topic IDs and map numbers must be embedded in both the software application and in your source documents, you and the application developers must agree in advance on the values to use.

Before you create context-sensitive help topics, complete the following steps:

1. Confirm with your application developers that the application supports context-sensitive help.
2. Meet with your application developers to identify each context-sensitive help topic you need to create.
3. Determine if you will use topic IDs or file names to create links to context-sensitive help topics.
4. Discuss the process for referencing context-sensitive help topics from the application with your application developers. Writers and application developers have the following options for creating context-sensitive help links:
  - The writer chooses the topic IDs or file names and embeds them in the source documents. If the generated output supports map files, the writer performs the following steps:
    - \_ The writer uses topic IDs inserted into source documents and ePublisher to generate a map file, also known as a header file, that contains the topic IDs defined by the writer and automatically generated mapping IDs.
    - \_ The writer supplies the generated map file to the application developers to implement.

**Note:** The writer must supply the header file along with the help system to the developers each time the writer updates the help system. This ensures correctly identified context-sensitive help topics each time.

- Application developers choose the topic IDs or file names and then give the topic IDs or filenames to the writer to embed in the source documents. If the generated output supports map files, the application developers perform the following steps:



- \_ Application developers create the map file, or header file.
- \_ Application developers give the writer a copy of the map file, or header file, and the writer embeds the topic IDs from the map file into the source documents.

**Note:** The group context must be unique so that if there are the same topic ID's in a help system, the context sensitive pointer will go to the correct place in the help.

## Topic ID and File Name Requirements

If you are creating context-sensitive help topics using topic IDs, topic IDs must follow these guidelines:

- Must be unique
- Must begin with an alphabetical character
- May contain alphanumeric characters
- May not contain special characters or spaces, with the exception of underscores ( \_ )

Dynamic HTML, Eclipse Help, Microsoft HTML Help, Oracle Help, Sun, WebWorks Help, and WebWorks Reverb support the use of topic IDs to create context-sensitive help.

If you are creating context-sensitive help topics using file names, file names must follow these guidelines:

- Must be unique
- Must begin with an alphabetical character
- May contain alphanumeric characters
- May not contain special characters or spaces

XML+XSL support the use of file names to create context-sensitive help.

## Output Formats that support Creating Context-Sensitive Help Links In DITA Source Documents

The following table lists available for creating context-sensitive help links.



### **that Support Context-Sensitive Help Links**

You can use ePublisher to create context-sensitive help links for the following :

- Eclipse Help
- Microsoft HTML Help
- Oracle Help
- Sun JavaHelp 2.0
- WebWorks Help
- WebWorks Reverb
- WebWorks Reverb 2.0

## **Specifying Context-Sensitive Help Links in DITA Source Documents**

Before you specify context-sensitive help links, review context-sensitive help link requirements. For more information about context-sensitive help and context-sensitive help link requirements, see “Context-Sensitive Help” and “Output Formats that support Creating Context-Sensitive Help Links In DITA Source Documents”.

### To specify a topic alias for a topic in a DITA source document:

1. In your DITA source document, locate your topic's meta information container element.
2. For this element, you use the `<othermeta>` element to define the topic alias, for example: `<othermeta name="TopicAlias" content="helpid"/>`

**Note:** You can also use the `<data>` element to define topic alias values, for example: `<data name="TopicAlias" value="helpid"/>`

3. Save the DITA document
4. Generate output for your target. For more information, see "Generating Output"

Steps for creating context-sensitive help links in DITA may be different in other versions of DITA.

## Creating Hyperlinks in DITA Source Documents

You will have to use DITA elements to create links in your documents so that ePublisher can create links to other topics or files in the online help output of your choice.

**To create an a hyperlink to a topic in the same DITA file or another DITA file, complete the following steps**

1. Identify your link type, as it will determine the markup that is required.
2. For a hyperlink that is going the same file create the following markup:  
`<xref href="#yourtopicID">Your xref link text</xref>`
3. For a hyperlink that is going to a different file create the following markup:  
`<xref href="file.xml#yourtopicID">Your xref link text</xref>`
4. For a hyperlink that is going to a PDF create the following markup:  
`<xref format="PDF" href="sample.PDF">Your xref link text</xref>`

## Creating Popups in DITA Source Documents

This section explains how to create popups in DITA source documents.

### Popups

A **popup window** is a window that is smaller than standard windows and typically does not contain some of the standard window features such as tool bars or status bars. Popup windows display when users hover over or click on a link. The popup topic closes automatically as soon as the users clicks somewhere else.

A typical use of popups is to display glossary terms. For example, in a printed document, terms and definitions are typically grouped in a separate glossary document. However, in a help system, you can display glossary definitions in popups. When you create glossary popups, users can choose whether they want to view the definition of an unfamiliar term. If they want additional information about the term, they can view the definition in a click.

You create popups by creating link between the word or phrase in a topic and the content you want to display in the popup, and then you use `<othermeta>` elements or paragraph styles to create popups.

#### Popup and Popup Append paragraph styles

Specifies that content displays both in popup windows and in standard help topics. You apply the Popup paragraph style to the first paragraph of content you want displayed in the popup window. If you have more than one paragraph of content you want to display, you apply the Popup Append style to the additional paragraphs.

For example, if you apply a glossary term and glossary definitions style for a glossary using the Popup and Popup Append styles, the terms and definitions in your output display in both a popup window and in a glossary topic that contains the definitions.

### **Popup Only and Popup Only Append paragraph styles**

Specifies that content displays only in popup windows. You apply the Popup Only paragraph style to the first paragraph of content you want displayed in the popup window. If you have more than one paragraph of content you want to display, you apply the Popup Only Append style to the additional paragraphs.

For example, if you apply a glossary term and glossary definition style for a glossary using the Popup Only and Popup Only Append paragraph style, the terms and definitions in your output display in only popup windows. The content is not displayed in an additional glossary topic that contains the definitions.

## **Requirements for Creating Popups in DITA Source Documents**

You prepare popups using <othermeta> elements or paragraph formats. Before you create popups in your source documents, verify that your , templates, and stationery meet popup requirements. The following table lists requirements for creating popups.

	Requirement
Output Format	<p>You can use ePublisher to create popups for the following :</p> <ul style="list-style-type: none"> <li>• Microsoft HTML Help</li> <li>• Oracle Help</li> <li>• Sun JavaHelp</li> <li>• WebWorks Help</li> <li>• WebWorks Reverb 2.0</li> </ul>

## Creating Popup Links in DITA Source Documents

Steps for creating links in DITA may be different in other versions of DITA.

## Using Paragraph Styles to Create Popups in DITA Source Documents

You can use Popup paragraph formats in your DITA source documents to create popups. To use Popup paragraph styles to create popup windows, your Stationery and DITA file must have the following items configured:

- Popup and Popup Append paragraph style behaviors if you want your content to display both in popup windows and in standard help topics.
- Popup Only and Popup Only Append paragraph style behaviors if you want your content to display only in popup windows.

The following procedure provides an example of how to use Popup paragraph formats to create popup windows in DITA source documents.

## To create popup windows using Popup paragraph styles in DITA source document

1. In your DITA source document, create a link between a word or phrase in the topic and the content you want to display in the popup window and ensure that the link resolves in the document.
2. Save your DITA source document.
3. In the ePublisher **Style Designer**, configure the destination paragraph styles with the appropriate popup behavior via the **Options** panel.
4. Generate output for your project.
5. In Output Explorer, go to the page where you created the popup window and verify that ePublisher created the popup window and that the popup window displays the content you specified.

# Creating Related Topics in DITA Source Documents

This section explains how to create related topics in DITA source documents.

## Related Topics

**Related topics** provide a list of other topics that may be of interest to the user viewing the current topic. For example, you could have a section called Creating Web Pages in your help. You may also have many other topics, such as HTML Tags and Cascading Style Sheets, that related to creating Web pages. Identifying these related topics for users can help them find the information they need and identify additional topics to consider. However, providing these types of links as cross-references within the content itself may not be the most efficient way to present the information. By utilizing related topics links, you combine the capabilities of cross-references with the efficiency of a related topics button.

Related topics and See Also links provide similar capabilities, but there are several important differences:

- Related topics can link to headings in a Help system that do not start a new page.
- Related topics links are static and defined in the source documents as links. You must have all the source documents to create the link and generate the output.



- If a related topics list contains a broken link in the source document, that link is broken in the generated output. In a See Also link list, the broken link is not included in the output.

The stationery designer can configure related topics to display in the following ways:

- Display in a popup window when the user clicks a button,
- Included in a list in the topic itself and then displayed in a popup window when the user clicks a button.

**Note:** If a related topic link is broken in the source document, in most cases that link is broken in the generated output. WebWorks Help provides an additional feature by removing broken links from related topics lists that are displayed in a popups window when a user clicks the Related Topics button.

## Requirements for Creating Related Topics Links in DITA Source Documents

You can create related topics using a paragraph format. Before you create related topics links in your DITA source documents, verify that your , templates, and stationery meet related topics links requirements. The following table lists requirements for creating related topics links.

	Requirement
Output Format	<p>You can use ePublisher to create related topics for the following :</p> <ul style="list-style-type: none"> <li>• Eclipse Help</li> <li>• Microsoft HTML Help</li> <li>• Oracle Help</li> <li>• Sun Java Help</li> <li>• WebWorks Help</li> <li>• WebWorks Reverb</li> <li>• WebWorks Reverb 2.0</li> </ul>

## Specifying Related Topics Links in DITA Source Documents

Create related topics links by applying the Related Topics paragraph format to cross-references you create in your DITA source documents. Before you create related topics in your source documents, review related topics links requirements. For more information about related topics and related topics links requirements, see “Related Topics” and “Requirements for Creating Related Topics Links in DITA Source Documents”.

The following procedure provides an example of how to create related topics links in DITA source documents using DITA 1.4. Steps for creating related topics links in DITA may be different in other versions of DITA.

## To create a related topics list in a DITA source document

1. Identify the topic in which you would like to insert a related topics list.
2. Identify the different topics you want to link to from this topic.

**Note:** Generally, you should only create one related topics list for each section of your source document that corresponds to a help topic. For example, if you have specified in your ePublisher project that there will be a page break at each Heading 1 section, then you should only create one related topics list for each Heading 1 section within your source document.

# Creating See Also Links in DITA Source Documents

This section explains how to create See Also links in DITA source documents.

## See Also Links

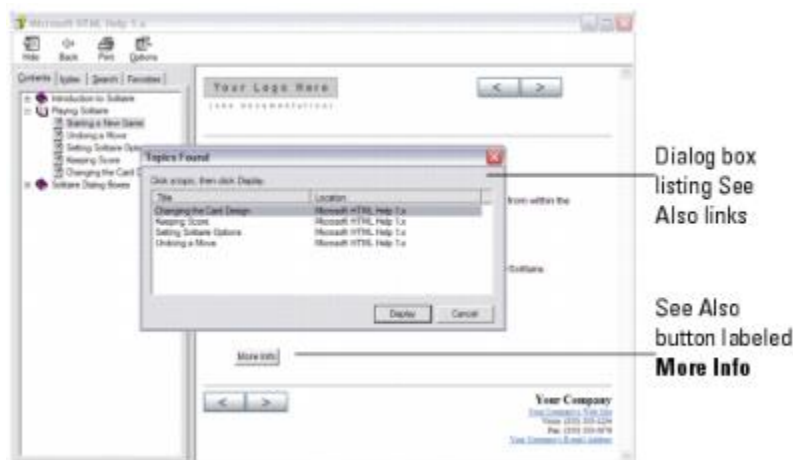
**See Also** links, also known as **ALinks**, or **associative links**, are links that may be of interest to the user viewing the current topic. These links use internal identifiers to specify the links and the link list is built dynamically based on the topics available when the user clicks to display the links. See Also links are important to use with larger help sets and merged help sets.

Related topics and See Also links provide similar capabilities, but there are several important differences:

- See Also links must link to styles that start a new topic, such as a heading.
- See Also links are dynamic and the lists of links are built at display time instead of during help generation.
- Since see Also link lists are dynamically built, they do not include links to topics that are not available when the user displays the links. If a list contains a broken link in the source document, that link is broken in the generated output for most .

See Also links are useful if you plan to merge help systems. For example, if you have a multiple help systems that you merge into one main help system at run time and if your topics in the merged help systems contain See Also keywords that are also used in the main help system, links to those topics are included in the See Also lists in the main project.

You can create See Also links as buttons or as inline text links in Microsoft HTML Help and WebWorks Help. The following example shows how the two different types of See Also links display in a Microsoft HTML Help system.



To create See Also links in your generated output, use a See Also paragraph format or character format defined by the stationery designer and through <othermeta> elements.

## Requirements for Creating See Also Links in DITA Source Documents

You create See Also links using a paragraph or character format and through <othermeta> elements. Before you create See Also links in your DITA source documents, verify that your , templates, and stationery meet See Also link requirements. The following table lists requirements for creating See Also links.

	Requirement
Output Format	<p>You can use ePublisher to create See Also links for the following :</p> <ul style="list-style-type: none"> <li>• Microsoft HTML Help</li> <li>• WebWorks Help</li> </ul>

## Specifying See Also Links in DITA Source Documents

Create See Also links by applying the See Also paragraph format or character format to text in your DITA source documents and through `<othermeta>` elements into your DITA source documents. Before you create See Also links in your source documents, review See Also link requirements. For more information about See Also links and See Also link requirements, see "See Also Links" and "Requirements for Creating See Also Links in DITA Source Documents".

Steps for creating See Also links in DITA may be different in other versions of DITA.

## Using the data element

This section explains how to use the `<data>` element in DITA source documents. The data element allows insertion of arbitrary marker style data similar to the `<othermeta>` element. Unlike the `<othermeta>` element, the `<data>` element can be used throughout a topic, just like a marker.

Using the data element to insert markers

The `<data>` element represents a property within a DITA topic or map. You can use the `<data>` element to insert a marker type anywhere on your page. Here is an example of the `<data>` element: `<data name = "TOCIcon" value = "red.png"/>`. In this example, our marker name is TOCIcon, and is using the value of red.png.

For more information on the `<data>` element see <https://docs.oasis-open.org/dita/v1.2/os/spec/langref/data.html>

## Assigning Custom Page Styles to Pages in DITA Source Documents

This section explains how to assign custom page styles to specific pages in DITA source documents.

## Page Styles

By default, each page generated by ePublisher is associated with the default page style defined in the stationery used by your ePublisher project. This means that typically you do not need to specify a page style for pages when you generate output.

For example, you may want to use one page style in your help system for all concept and procedure topic pages, and another page style for all context-sensitive window description topic pages in your help system. In this example, you can use the default page style for all of your concept and procedure topic pages, and then you can use a second custom page style defined in your stationery for all context-sensitive window description topic pages in your help system.

## Requirements for Specifying Custom Page Styles for Pages in DITA Source Documents

You specify page styles for pages using `<othermeta>` elements. Before you specify page styles for pages by using `<othermeta>` elements in your DITA source documents, verify that your , templates, and stationery meet image style requirements. The following table lists requirements for specifying page styles for pages.

	Requirement
Output Format	<p>You can use ePublisher to specify page styles for specific images for the following :</p> <ul style="list-style-type: none"> <li>• Dynamic HTML</li> <li>• Eclipse Help</li> <li>• eBook - ePub 2.0</li> <li>• Microsoft HTML Help</li> <li>• Oracle Help</li> <li>• Sun JavaHelp</li> <li>• WebWorks Help</li> <li>• WebWorks Reverb</li> <li>• WebWorks Reverb 2.0</li> </ul>

## Specifying Custom Page Styles for Pages in DITA Source Documents

For more information about page styles and page style requirements, see “Page Styles” and “Requirements for Specifying Custom Page Styles for Pages in DITA Source Documents”.

### To specify a Page Style for a topic in a DITA source document:

1. In your DITA source document, locate your topic's meta information container element.
2. For this element, you use the `<othermeta>` element to define the Page Style, for example: `<othermeta name="PageStyle" content="stylename"/>`
3. Save the DITA document
4. Generate output for your target. For more information, see "Generating Output"

Steps for specifying custom page styles for pages in DITA may be different in other versions of DITA.

## Using Custom Graphic Styles for Images in DITA Source Documents

This section explains how to use custom graphic styles for images in DITA source documents. It also explains how graphic styles are assigned by default using implicit image properties in the source content.

### Assigning Graphic Styles

By default, each image generated by ePublisher is associated with one of several possible graphic styles that all begin with **Default**. This means that typically you do not need to specify a graphic style for images when you generate output.

For example, you may want to create a special style called "thumbnail" for handling very large images. In this example, each graphic that you assign the "thumbnail" style would have those defined properties. All other graphics would use one of the **Default** graphic styles.



## To use a custom graphic style for images in your DITA source documents:

1. In your DITA source document, locate the image/s that will be assigned the custom graphic style.

Example: `<image href="myimage.png" />`

2. Within this image element, insert the `outputclass` attribute to assign the custom graphic style to the image. Assign the `outputclass` value as the name of the custom graphic style.

Example:

`<image href="myimage.png" outputclass="CustomGraphicStyle" />`

3. Define the graphic style in ePublisher Designer.

## Default Graphic Styles for DITA

For DITA, ePublisher will automatically use predefined graphic styles based on the properties of the graphic that is being generated. Below are the list of predefined graphic styles that ePublisher uses for DITA images when no custom graphic style has been assigned.

Graphic Style Name	Example DITA Code to Produce
<b>Default</b>	<pre>&lt;image href="file.png"/&gt;</pre> <pre>&lt;image placement="break" href="file.png"/&gt;</pre>
<b>DefaultLeft</b>	<pre>&lt;image placement="break" align="left" href="file.png"/&gt;</pre>
<b>DefaultCenter</b>	<pre>&lt;image placement="break" align="center" href="file.png"/&gt;</pre>
<b>DefaultRight</b>	<pre>&lt;image placement="break" align="right" href="file.png"/&gt;</pre>
<b>Default Scalefit</b>	<pre>&lt;image scalefit="yes" href="file.png"/&gt;</pre>

## Customizing TOC Entry in DITA

Use these steps to customize a TOC entry in your **Reverb 2.0** output. Your DITA file must have a nested heading structure for TOC Icons to appear.

1. Locate the `<prolog>` element in your DITA document. If your document does not have a `<prolog>` element, you can insert one near the top of the document under the opening `<concept>`, `<task>`, or `<topic>` element. A `<metadata>` element must be nested within the `<prolog>` element. An `<othermeta>` element must be nested within the `<metadata>` element. The end result should look like this:

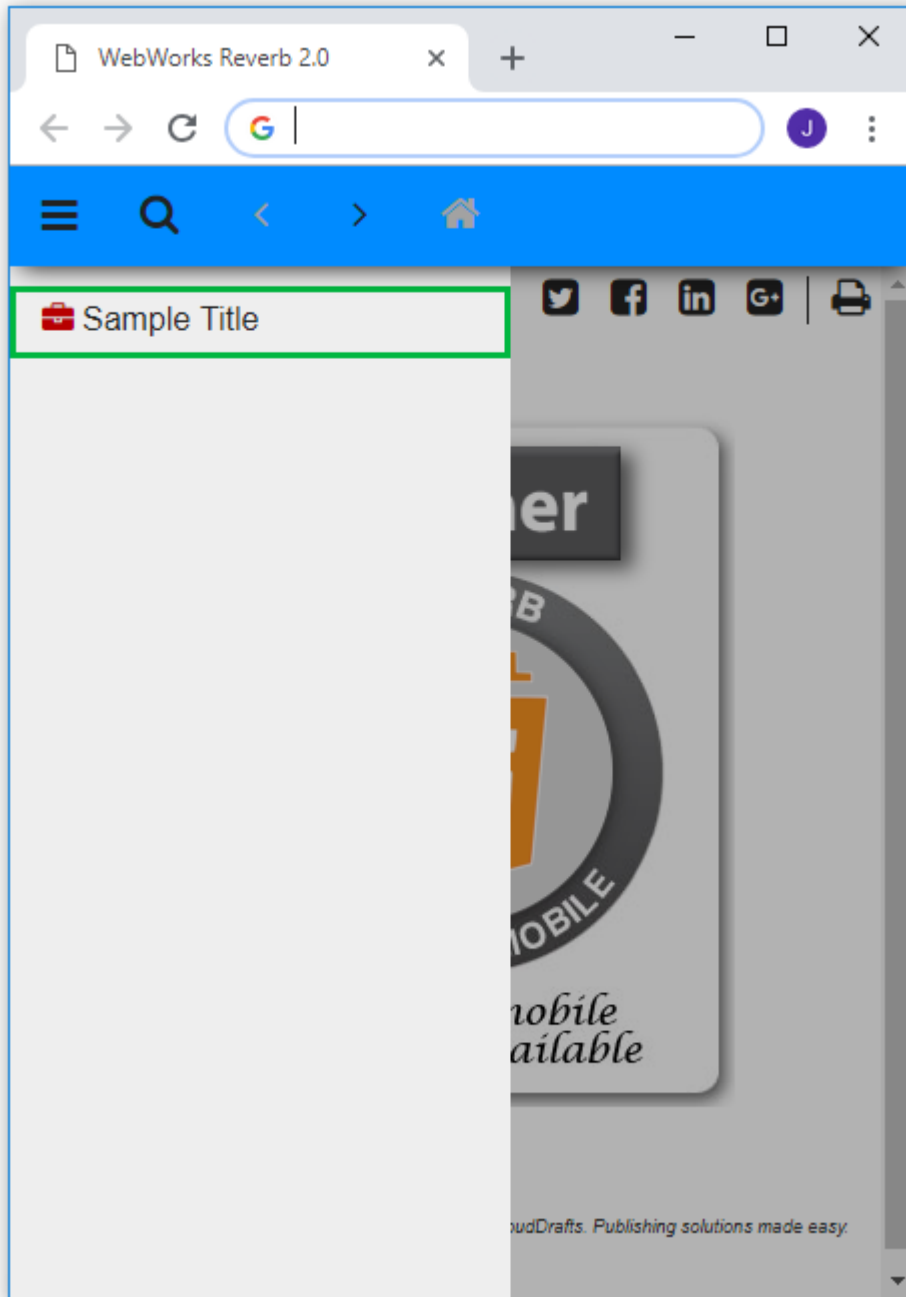
```
<prolog>
  <metadata>
    <othermeta name="TOCEntryClass" content="folder_icon" />
  </metadata>
</prolog>
```

2. The `<othermeta>` tag requires a name and content attribute. The value of the name attribute will be "TOCEntryClass". The value of the content attribute can be whatever value you would like to use. The value of the content attribute will become the name of the CSS class that you will customize in an override of the `*.scss` files.
3. Save your DITA document.
4. Scan the document in ePublisher Designer.

5. Open the **Style Designer**.
6. Open **Marker Styles**.
7. Locate the **Marker Type Option** from the **Options** tab and set its value to `TOC Entry Class`.
8. In this example, the assigned class for the Menu TOC entry will be the value of the marker: `folder_icon`.
9. Add the following to a target override of `icons.scss`. Notice how the CSS class is the name of the value given in the Marker Text Window. In this example we change the icon color and the icon of the TOC entry. You are able to make other customizations such as adding a border, or changing the background color.

```
.folder_icon {  
  
    > div > span > i {  
  
        color: black;  
  
        &:before {  
  
            content: $folder_icon;  
  
        }  
  
    }  
  
}
```

10. Save your project and generate the output.



## Customizing Table of Contents Icons for Topics in DITA Source Documents Using Legacy Outputs

This section explains how to customize the appearance of table of contents icons for topics in Microsoft HTML Help, Sun JavaHelp, Oracle Help, and WebWorks help systems. For information on how to customize Menu TOC entries in Reverb 2.0, see “Customizing TOC Entry in DITA”.

## **Requirements for Specifying Custom Table of Contents Icons in DITA Source Documents**

	Requirement
Output Format	<p>You can use ePublisher to specify custom table of contents icons in the following :</p> <ul style="list-style-type: none"> <li>• Microsoft HTML Help</li> <li>• Oracle Help</li> <li>• Sun JavaHelp</li> <li>• WebWorks Help</li> </ul>

## Specifying Custom Table of Contents Icons in DITA Source Documents

For more information about custom table of contents icons, see “Requirements for Specifying Custom Table of Contents Icons in DITA Source Documents”.

## To specify a Custom Table of Contents Icon for a topic in a DITA source document:

1. In your DITA source document, locate your topic's meta information container element.
2. For this element, you use the `<othermeta>` element to define the Page Style, for example: `<othermeta name="TOCIcon" content="blue.png"/>`.

**Note:** You can also use the `<data>` element, for example: `<data name="TOCIcon" value="blue.png"/>`.

3. Save the DITA document
4. Generate output for your target. For more information, see "Generating Output"

Steps for customizing table of contents icons for topics in DITA may be different in other versions of DITA.

## To specify a custom table of contents icon in a DITA source document

5. **If you want to specify a custom table of contents icon for Microsoft HTML Help**, identify the number of the image you want to use for the table of contents image for the topic in the `.hhp` file for your Microsoft HTML Help project by completing the following steps:
  - a. On the **View** menu, click **Output Directory**.
  - b. Open the `ProjectName` folder, where *ProjectName* is the name of your project.
  - c. Open the `ProjectName.hhp` file where *ProjectName* is the name of your project.
  - d. On the **Contents** tab, select a table of contents entry, and then click the **Pencil** icon.
  - e. On the **Advanced** tab, in the **Image index** field, use the up and down arrows to identify the table of contents image you want to use for the topic.
  - f. Note the number of the image you want to use for the table of contents image for the topic.

For example, if you want to use a question mark icon with a red star for the table of contents icon for new topics, note that the number for this icon is 10.
  - g. Close HTML Help Workshop.
6. **If you want to specify a custom table of contents icon for Oracle Help or Sun JavaHelp**, create the graphic file for the custom table of contents icon in `.gif` format. The default graphics used as Sun JavaHelp or Oracle Help table of contents icons are 17 x 17 pixels. The custom graphics you create for Sun JavaHelp or Oracle Help table of contents icons should also be 17 x 17 pixels. You can assign any name to the graphic files.
7. **If you want to specify a custom table of content icon for WebWorks help**, create graphics files containing the collapsed and expanded versions of the icons you want to use, then save the graphic files in `.gif` format. The default graphics used as WebWorks Help table of contents icons are 17 x 17 pixels. The custom graphics you create for WebWorks Help table of contents icons should also be 17 x 17 pixels. You can assign any name to the graphic files.
8. Copy the graphic files you want to use as icons in the table of contents into the following folder:



**Note:** If the folder does not exist, first create the folder using the specified folder structure and then copy the graphic files you want to use as icons into the folder. You do not need to perform this step when specifying custom table of contents icons for Microsoft HTML Help.

- **If you are generating Oracle Help**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Oracle Help\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating Sun JavaHelp 1.1.3**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Sun Java Help 1.1.3\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating Sun JavaHelp 2.0**, copy the graphic files you want to use into the following folder:

`ProjectName\Formats\Sun Java Help 2.0\Files\images` folder, where *ProjectName* is the name of your project.

- **If you are generating WebWorks Help**, in your `ProjectName\Files` folder, where *ProjectName* is the name of your project, create a `wwhelp\images` subfolder and copy the graphic files you want to use into this folder. Your project file structure should be similar to the following structure:

`ProjectName\Files\wwhelp\images`

## Using markopen and markclose

This `default.wwconfig` modification specifies the start and a stop of an element. This is especially useful for the implementation of the dropdown elements in WebWorks Help 5 or WebWorks Reverb output. ePublisher has preconfigured items such as the Definition Term Elements, below is an example of the sample DITA markup:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE concept PUBLIC "-//OASIS//DTD DITA Concept//EN"
"concept.dtd">
<concept id="simplelist" xml:lang="en-us">
  <title>Definition List</title>
  <conbody>
    <p>Definition list</p>
    <dl>
      <dlhead>
```

```

    <dthd>Image File View Selection</dthd>
    <ddhd>Resulting Information</ddhd>
</dlhead>
<dlentry>
    <dt>File Type</dt>
    <dd>Image's file extension</dd>
</dlentry>
<dlentry>
    <dt>Image Class</dt>
    <dd>Image is raster, vector, metafile or 3D</dd>
</dlentry>
<dlentry>
    <dt>Number of pages</dt>
    <dd>Number of pages in the image</dd>
</dlentry>
<dlentry>
    <dt>Fonts</dt>
    <dd>Names of the fonts contained within a vector image</dd>
</dlentry>
</dl>
<p>Content after the definition list.</p>
</conbody>
</concept>

```

Without the use of the `markclose` in the `default.wwconfig`, the last paragraph would be underneath the last Definition Term element. To make sure that no other elements are inside the dropdown, you can assign `markclosed` to it by creating an override to `default.wwconfig` located in `[project directory]\Formats\Adapters\xml\scripts\dita`. For more information, refer to “Creating Format Overrides” Below is an example of the modification needed to get the desired output:

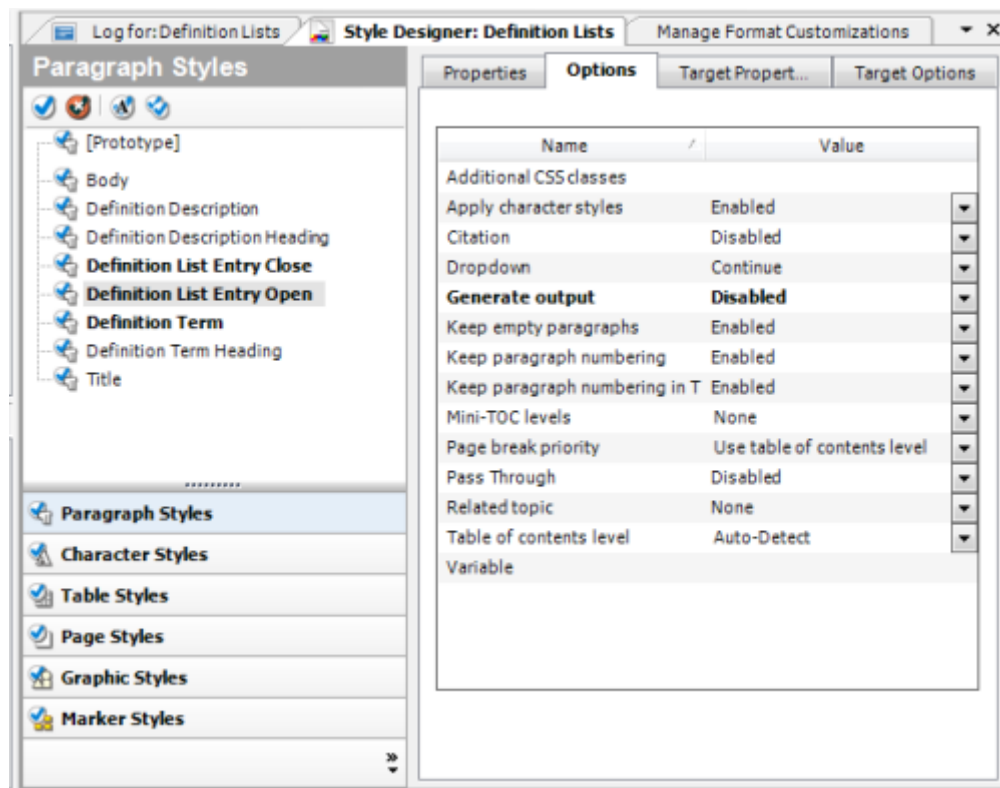
```

<!-- Mark open/close on definition entries -->
<!--                                     -->
<Style match="//*[contains(@class, ' topic/dlentry ')]">
    <xsl:variable name="VarName" select="'Definition List Entry'" />
    <wwditaconfig:Head name="{ $VarName}" markopen="{ $VarName} Open"
markclose="{ $VarName} Close" />
</Style>

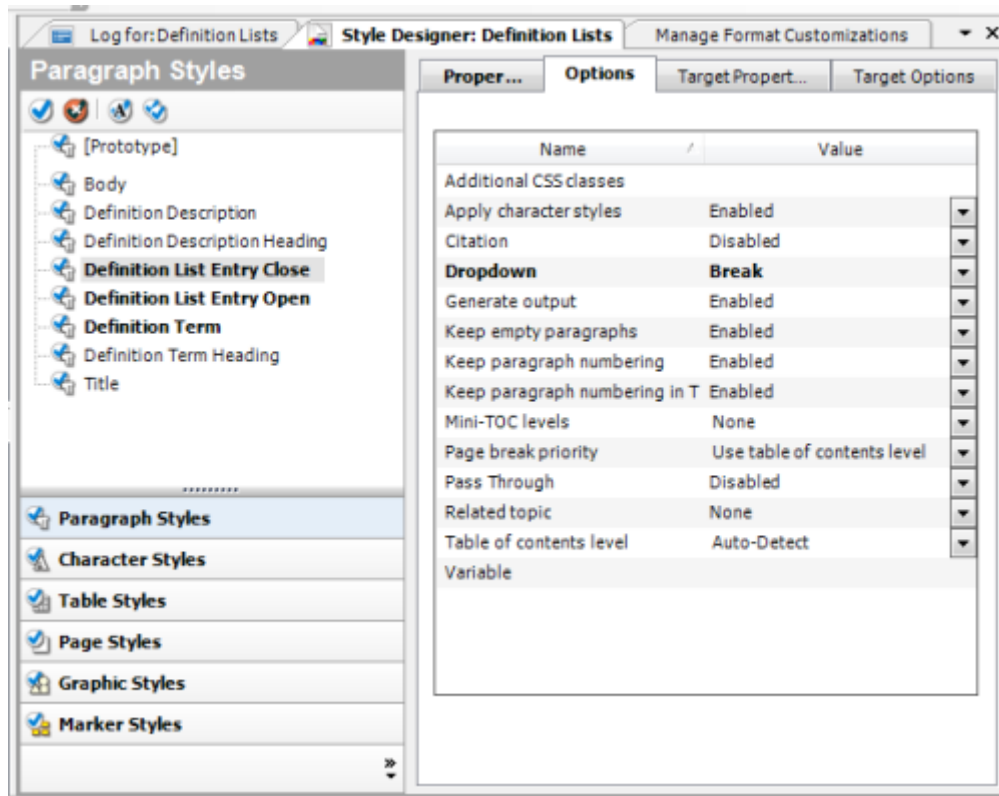
```

## Configuring markopen and markclose entries for dropdowns in ePublisher

After you have created the override, you will still need to scan for the newly created paragraph styles in ePublisher. Using our created override, you would see something like this in the Style Designer:



The output is set to disabled because the markopen just serves as a placeholder text. In the closed entry, we set the Options to have the Dropdown to Break:



We do this because that is how the content will emit, below is an example of the sample output's HTML that shows the paragraph styles as div classes:

```

20 <div class="Definition_List_Entry_Close"><a name="3_2_2_4_8c">&nbsp;</a></div>
21 <div class="Definition_Term" onclick="WebWorks_ToggleDIV(WebWorksRootPath,
    &quot;wwdd3_2_2_4_10_2&quot;);"><a name="3_2_2_4_10_2">Fonts</a><script
    type="text/javascript" language="JavaScript1.2">WebWorks_WriteArrow(WebWorksRootPath,
    "wwdd3_2_2_4_10_2", false);</script> <a
    href="javascript:WebWorks_ToggleDIV('wwdd3_2_2_4_10_2');"></a></div>
22 <script type="text/javascript"
    language="JavaScript1.2">WebWorks_WriteDIVOpen("wwdd3_2_2_4_10_2", false);
</script><div id="wwdd3_2_2_4_10_2" style="visibility: visible; display: block;">
23 <div class="Definition_Description"><a name="3_2_2_4_10_4">Names of the fonts
    contained within a vector image</a></div>
24 <script type="text/javascript" language="JavaScript1.2">WebWorks_WriteDIVClose();
</script></div>
25 <div class="Definition_List_Entry_Close"><a name="3_2_2_4_10c">&nbsp;</a></div>
26 <div class="Body"><a name="3_2_2_6">Content after the definition list.</a></div>
27 </blockquote>

```

Note that the Content after the definition list is now in a Body paragraph, so that it does not become included in the dropdown.

## Troubleshooting DITA issues

Occasionally there might be issues with the source documents you are using. Below is a list linking to the wiki solutions website that will help you troubleshoot each one:

Issue	Solution
If you are having issues with whitespace when authoring with FrameMaker using DITA	<a href="#">White space in FrameMaker</a>
If you are having issues with cross references when authoring with FrameMaker using DITA. In particular, if you are using the fm-xref element, generated output will not properly update the generated numbering. There is a solution to fix this behavior in FrameMaker.	<a href="#">DITA Cross References in FrameMaker</a>
If you are receiving a Cannot Duplicate Document	<a href="#">Cannot duplicate document error message</a>
If you are having issues outputting cross references from DITA	<a href="#">Cross Reference to same topic not working</a>
If you are having issues using a custom DTD	<a href="#">DTD Issues</a>
If you are using an older or newer version of the OTK	<a href="#">Different versions of the OTK</a>
If you are trying to use conditional text	<a href="#">DITA conditions</a>
If you are trying to localize figure/table paragraphs	<a href="#">Localization of the word "table" or "figure"</a>

Issue	Solution
If you are getting a Java error when generating	<a href="#">Java Error</a>
If you are trying to put a list in a table	<a href="#">List in a Table</a>
If you are getting a Resample WIFError upon generation	<a href="#">ResampleWIF Pipeline Error Blocks Generation of Output</a>
If your elements are not being populated in the style designer	<a href="#">DITA styles are not being read in the Style Designer</a>
If your Overview topics are showing up on the same level as their children	<a href="#">DITA Overview Topics Showing on Same Level as Children Topics</a>

# Markdown++ Output Format

Introduction

Setting up a Markdown++ Target in ePublisher

Basic Workflow for Adding and Converting Documents

Configuration Tasks for Markdown++

Markdown++ Tables and Layout

## Introduction

The **Markdown++ output format** is a versatile output target designed as both an output and an interoperable source format within ePublisher 2024.1. Unlike traditional output formats, Markdown++ is tailored for integration with tools that utilize Markdown, such as OpenAI's ChatGPT, GitHub, and others. It provides an open format that streamlines transitions from proprietary authoring systems—such as FrameMaker, Word, or DITA-XML—to Markdown, supporting transparent, collaborative technical writing in Markdown-compatible systems.

### Primary Uses:

- **Migrating Content:** Seamlessly migrate source files from FrameMaker, Word, or DITA-XML into Markdown++.
- **Interoperability:** Facilitate content creation and maintenance in an accessible Markdown format for enhanced collaboration.
- **AI Compatibility:** Create documentation directly accessible for AI systems, including OpenAI's ChatGPT.

## Setting up a Markdown++ Target in ePublisher

Setting up a Markdown++ output target in ePublisher is straightforward. Using ePublisher's Designer interface, you can convert existing source documents into Markdown++ format and manage outputs with ease.

### Steps to Set Up

#### 1. Create a New Project or Use Existing Stationery:

- Open **ePublisher Designer** and either start a new project or select existing Stationery.
- Markdown++ can be added as a target to an existing project, enabling conversion from any compatible source format.

#### 2. Add Markdown++ as an Output Format:

- In your Stationery Design Project, choose **Add Target**.
- Select **Markdown++** from the available output formats.

### 3. **Scan Documents:**

- After adding Markdown++ as a target, **Scan All Documents** to ensure that all styles, variables, and elements from the source documents are recognized and mapped for the Markdown++ output.

### 4. **Generate Output:**

- Once your documents are scanned and the target is configured, select **Generate Output**. ePublisher will create Markdown++ files for each source document, organized in a Markdown-friendly structure.

## Basic Workflow for Adding and Converting Documents

To add and convert documents to Markdown++:

- **Add Source Documents:** Use the Document Manager to add source files.
- **Set Markdown++ as the Active Target:** In the Manage Targets menu, ensure Markdown++ is selected as the active target.
- **Generate and Review Output:** Use the **Preview Window** to check the output structure.

**Note:** Markdown++ files generated by ePublisher are immediately ready for further editing or maintenance, making it easy to shift from complex authoring tools to Markdown-based workflows.

## Workflow for Generating Markdown++ Output

Markdown++ enables round-tripping, where content can be converted from Markdown back into Markdown++. This feature aids in standardizing code, styles, and document structures across projects.

1. **Prepare Source Files:** Add your source files (from Word, FrameMaker, or DITA-XML) to the project.
2. **Select Markdown++ as Target:** Specify Markdown++ as the active output target in the Document Manager.
3. **Generate Markdown++ Files:** Review the generated files, which will include Markdown files for each topic and structured elements that adhere to Markdown++ specifications.



# Configuration Tasks for Markdown++

Markdown++ in ePublisher offers powerful configuration options to enhance the versatility and control of your Markdown output. By fine-tuning settings such as the **Markdown Syntax** and **Indentation Level** properties for paragraph and character styles, you can produce highly structured, readable, and stylistically consistent Markdown documents. These configuration tasks empower technical writers to create sophisticated layouts that retain compatibility with other Markdown tools while leveraging Markdown++'s advanced capabilities. This section provides guidance on these essential configuration tasks, helping you set up a seamless Markdown workflow that meets both organizational standards and the demands of complex technical documentation.

## Configuring Each Paragraph and Character Style's Markdown Syntax Property

The **Markdown Syntax** property in ePublisher's Markdown++ output format allows you to control the specific Markdown representation for both **paragraph** and **character** styles. While the default setting of **Auto-Detect** often yields acceptable results, configuring each style's Markdown syntax explicitly ensures consistent formatting and improves output quality, especially for production workflows.

### Setting the Paragraph Style's Markdown Syntax Property

By default, the **Markdown Syntax** property for paragraph styles is set to **Auto-Detect**, which infers the appropriate Markdown based on the source paragraph's structure and formatting. For greater control, especially when preparing final documentation, you can specify an explicit Markdown syntax from the following options:

- **Title 1, Title 2, Heading 1, Heading 2, Heading 3, Heading 4, Heading 5, Heading 6:** Corresponds to Markdown heading syntax (e.g., `#`, `##`, etc.). Use these options to explicitly set headers for different sections.
- **Unordered List, Ordered List:** Configures paragraphs to render as list items, aligning with Markdown's list syntax (`-`, or numbered).
- **Blockquote:** Formats paragraphs as blockquotes using the `>` character in Markdown.
- **Code Fence:** Renders content within a fenced code block, using backticks (```).
- **None:** Renders the content as a plain paragraph without additional Markdown syntax, recommended when there is no specific formatting required.

To configure a paragraph style's Markdown Syntax property:

1. In **ePublisher Designer**, open the **Style Designer** for your project.

2. Select the **Paragraph Style** you wish to configure.
3. Locate the **Markdown Syntax** property.
4. Choose the desired setting from the dropdown list, or keep **Auto-Detect** if unsure.
5. Save your settings to apply changes to the Markdown++ output.

**Tip:** If you're uncertain of the specific Markdown syntax, set the property to **None** for a plain paragraph appearance.

## Setting the Character Style's Markdown Syntax Property

The **Markdown Syntax** property for character styles defines inline formatting, such as bold or italic text. The available options are:

- **Bold:** Formats text in bold (`**text**`).
- **Italic:** Formats text in italics (`*text*`).
- **Strikethrough:** Formats text with strikethrough (`~~text~~`).
- **Code:** Renders text within inline code format using backticks (``text``).
- **None:** Renders the text without additional Markdown syntax.

To configure a character style's Markdown Syntax property:

1. In **ePublisher Designer**, open the **Style Designer** for your project.
2. Select the **Character Style** you wish to configure.
3. Locate the **Markdown Syntax** property.
4. Choose an explicit setting (e.g., **Bold**, **Italic**), or use **Auto-Detect** if unsure.
5. Save your settings to apply changes to the Markdown++ output.

**Recommendation:** For best results in production, avoid **Auto-Detect** and explicitly configure each style's Markdown Syntax property. This ensures clarity and consistency across your Markdown++ documentation.

## Configuring Each Paragraph Style's Markdown Indentation Level Property

The **Indentation Level** property in ePublisher's Markdown++ output format allows you to create nested paragraph structures, providing flexibility in organizing content hierarchically. This property applies only to **paragraph styles** and controls whether a paragraph nests under its **preceding sibling paragraph**—the paragraph immediately before it in the document structure.

By default, **Indentation Level** is set to **None**, preventing any indentation relative to the preceding paragraph. However, by configuring this property to a value of **1** or higher, you can create multi-level structures, such as nested lists, indented blockquotes, or even complex outline formats.

## How Indentation Works

When set to **1** or more, the **Indentation Level** property will nest the paragraph one level beneath its preceding sibling, but only if:

1. The preceding sibling paragraph is configured as a **container paragraph**—that is, a paragraph that allows nesting.
2. The preceding sibling's **Markdown Syntax** property is set to one of the following container-compatible formats:
  - **Unordered List**
  - **Ordered List**
  - **Blockquote**

**Note:** A paragraph's indentation level cannot exceed one level beneath the preceding sibling, but it can nest as deep as the structure allows, based on the indentation of previous paragraphs.

## Setting Indentation Levels

To configure a paragraph style's indentation level:

1. In **ePublisher Designer**, open the **Style Designer** for your project.
2. Select the **Paragraph Style** you wish to configure.
3. Locate the **Indentation Level** property.
4. Set the **Indentation Level** to the desired number (1 or greater) to enable nesting.
5. Save your settings to apply changes to the Markdown++ output.

## Example Use Cases

The **Indentation Level** property provides a powerful way to structure Markdown content. Here are some examples:

- **Nested Lists:** Create lists within lists by setting the **Indentation Level** on each successive list item style. This structure is ideal for outlines, multi-tiered content, or Harvard-style lists.
- **Blockquote Content Islands:** Place an indented paragraph within a blockquote, allowing for multiple paragraphs nested under a single quote. This technique is effective for formatting testimonials, quoted sections, or side content within a primary narrative.

The **Indentation Level** property, combined with other Markdown++ formatting options, allows for sophisticated document structures that can elevate your Markdown presentations.

## Markdown++ Tables and Layout

Markdown++ extends traditional Markdown with advanced table support and flexible layout options, enabling you to create professional, structured documents with ease. The **multi-line pipe table** feature, a standout capability of Markdown++, supports complex content within table cells, including multi-line text, images, and rich formatting. This section introduces the configuration options available for Markdown++ tables and provides insights on maximizing their layout potential. By mastering these features, you can adapt Markdown++ to meet diverse documentation needs, from data-rich tables to intricate, styled layouts, ensuring seamless output across Markdown-aware tools and ePubublisher's full range of output formats.

### Introduction to Markdown++ Multi-line Pipe Tables

One of the most powerful features of the Markdown++ output format is its advanced support for tables, allowing technical authors to create professional, complex table layouts within Markdown. Markdown++ offers two table-rendering options under the **Table Style** setting, called **Table Rendering**:

- **Pipes Multiline** (default): Supports multi-line cell content, along with embedded Markdown++ elements such as paragraphs, character styles, images, and markers.
- **Pipes:** Traditional Markdown pipe tables, suitable for simpler tables or datasets.

### Using Multi-line Pipe Tables

The **Pipes Multiline** option is recommended for most users because it offers complete support for multi-line content within cells. This setting preserves the

readability and formatting of complex tables, allowing for robust styling and layout configurations that are compatible with Markdown syntax.

### Key Advantages of Multi-line Pipe Tables:

- **Supports Advanced Content:** Each cell can contain multi-line content, including nested Markdown++ paragraphs, images, and markers.
- **Compatibility with Markdown Tools:** While fully compatible with most Markdown-aware tools, tools that do not support Markdown++ will render a basic version of the table without advanced styling. Multi-line content will appear as individual rows in non-Markdown++ viewers, preserving basic readability.
- **Migration Capabilities:** Use multi-line pipe tables to transfer complex tables from FrameMaker, Word, or DITA-XML, retaining layout fidelity within Markdown while making content more accessible and transparent.

## Configuring Table Rendering

To configure the table rendering mode:

1. In **ePublisher Designer**, open the **Style Designer** for your project.
2. Select the **Table Style** you wish to configure.
3. Locate the **Table Rendering** option.
4. Choose either:
  - **Pipes Multiline:** For full-featured tables that support multi-line and styled content.
  - **Pipes:** For simple, single-line tables with basic content.
5. Save your settings to apply changes to the Markdown++ output.

## Benefits of Markdown++ Tables Across ePublisher Outputs

Markdown++ tables can be styled with a **Markdown++ style name**, providing consistent and rich styling options when generating other ePublisher outputs such as PDF, WebWorks Reverb 2.0, or HTML. This allows authors to leverage Markdown++'s Markdown-based transparency while still achieving high-quality output across multiple formats.

For simpler tables focused on plain data representation, the **Pipes** option provides a straightforward solution, ideal for basic data tables or scenarios where simplicity is a priority.

Markdown++'s multi-line pipe tables offer an unparalleled combination of compatibility and fidelity, bridging the gap between traditional authoring tools and the flexibility of Markdown. This unique feature empowers authors to retain full control over table layouts, bringing Markdown-based tables to a professional standard without sacrificing compatibility.

# WebWorks Reverb 2.0

- [Choosing a Skin](#)
- [Using SASS To Customize Reverb Interface](#)
- [Previewing Reverb Output](#)
- [Delivering Reverb Output](#)
- [Top-Level Groups in Reverb](#)
- [Searching Output](#)
- [Using Baggage Files](#)
- [Searching with URL Method](#)
- [TOC or Index with URL Method](#)
- [Launching Context Sensitive Help for WebWorks Reverb 2.0](#)
- [End-user requirements in WebWorks Reverb 2.0](#)
- [Analytics Event Tracking in Reverb 2.0](#)
- ['Was This Helpful?' Buttons](#)
- ['Was This Search Helpful?' Buttons](#)
- [Document Last Modified Date in Reverb](#)
- [Drop-down Expand/Collapse All Toggle Button](#)
- [Google Translate Button](#)
- [Customizable Header and Footer](#)
- [Custom TOC Menu Items](#)
- [Customizing a Bullet Icon using Font Awesome](#)
- [Using the url\\_maps.xml reference file](#)

WebWorks Reverb 2.0 has many of the features found in WebWorks Help, and goes one step further by providing high-performance load times combined with optimal handling of the end-user's device resolution. This means that users viewing this type of output on a mobile device will be able to view the files in a fast, light-weight manner that is optimal for lower resolutions and/or touch operated screens. Furthermore, users on a more traditional higher-resolution computer monitor will still have the same advantages offered by a complete online help system. In other words, Reverb adapts to the traits of the device that is doing the reading of the content. Additionally, the JavaScript code for WebWorks Reverb 2.0 has been written in an inline-safe manner, which makes the output even more secure on the web.

WebWorks Reverb 2.0 also offers a commenting and end-user feedback mechanism using the Disqus commenting and discussion platform. This platform is a leading solution on the Internet for comment handling and has no cost and provides automatic hosting of your end-users comments in a way that is transparent and effective.

Reverb also provides easy Google Analytics integration. Using Google Analytics in your web files will be very easy to configure with this format.

If you have end-users that require localized help files, WebWorks Reverb 2.0 offers a seamless integration with the Google Translate Element. This feature is very powerful and can be used instead of an expensive and burdensome set of translated files and help volumes, which requires a separate set of content for each language that you are supporting. Using WebWorks Reverb 2.0, you can now create

instantaneously localized help volumes without any translation requirements and only one set of help files. For more information see “Google Translate Button”.

## Choosing a Skin

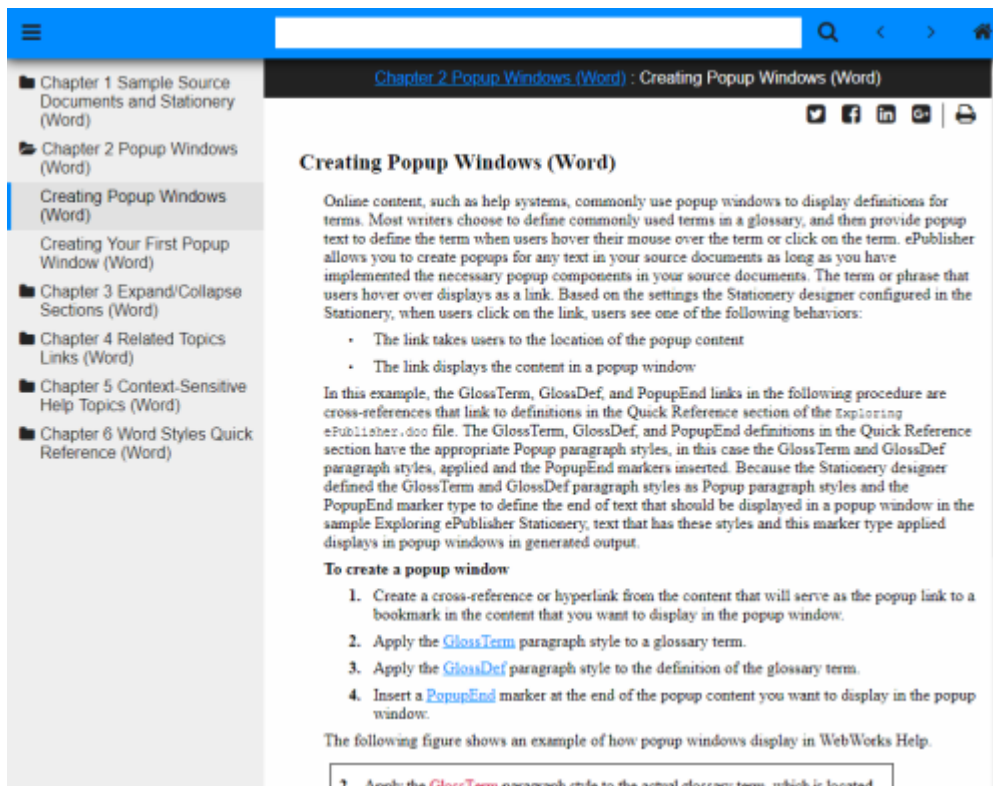
You can select from a number of predefined skins that are available for the WebWorks Reverb 2.0 output format. Each skin has been professionally designed with assistance from both graphic designers and web developers so that your users can get an ideal experience when browsing your documentation.

**Note:** If you plan on customizing the Reverb 2.0 toolbar or menu, we recommend not using the skins to simplify future upgrades.

You may select among the following types of skins:

### Neo

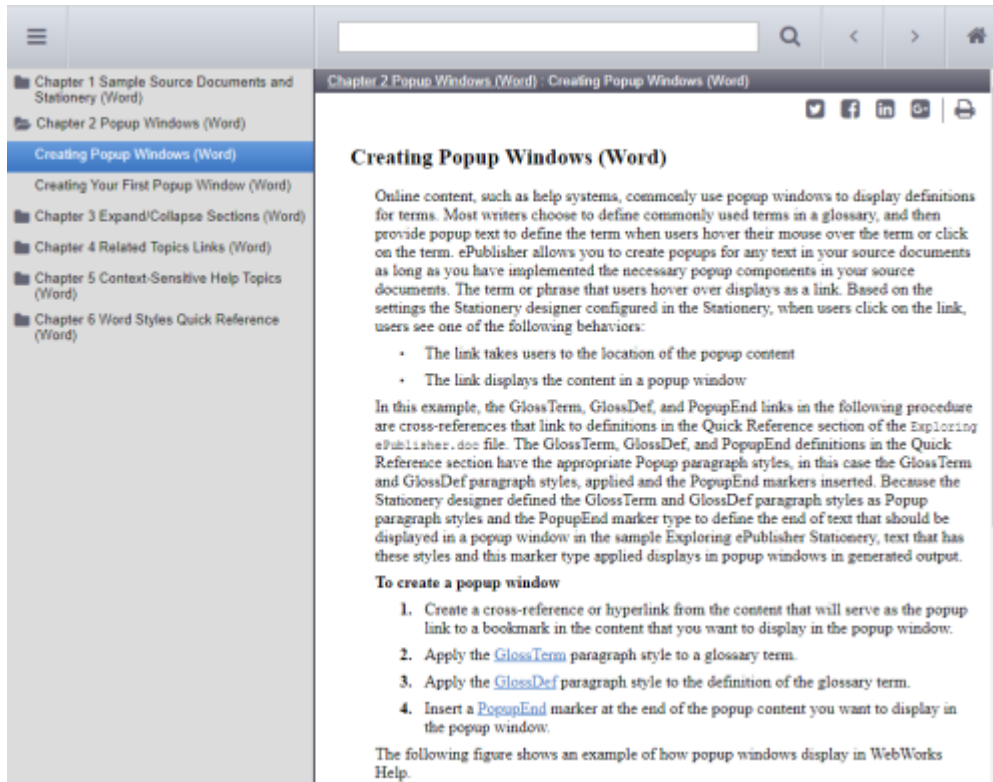
The default skin for WebWorks Reverb 2.0, Neo is the latest design for Reverb output and was designed with current web aesthetics in mind. It features a simplistic layout that looks great across many devices. This skin is very versatile and well suited for most purposes.



### Classic

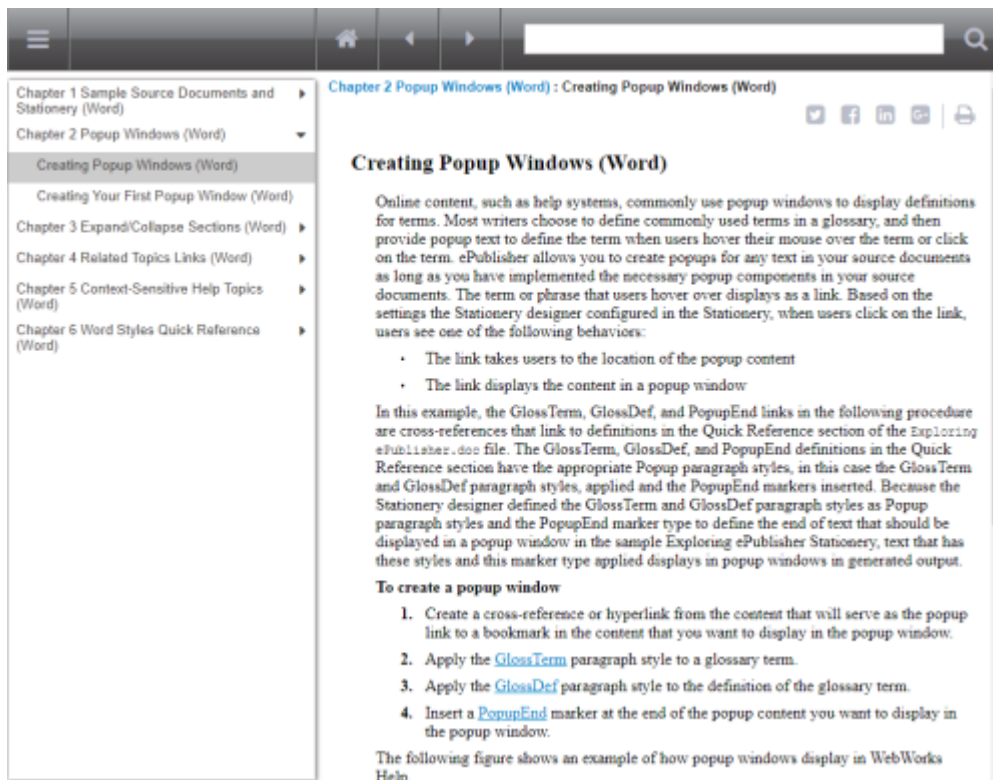


Based off of the original WebWorks Reverb skin, Classic makes a return with some modifications in WebWorks Reverb 2.0. This skin features a larger toolbar, and gradients across the layout to produce a traditional help look and feel. This skin is great for many purposes and offers existing users a similar skin if they were previously using the Classic skin with WebWorks Reverb.



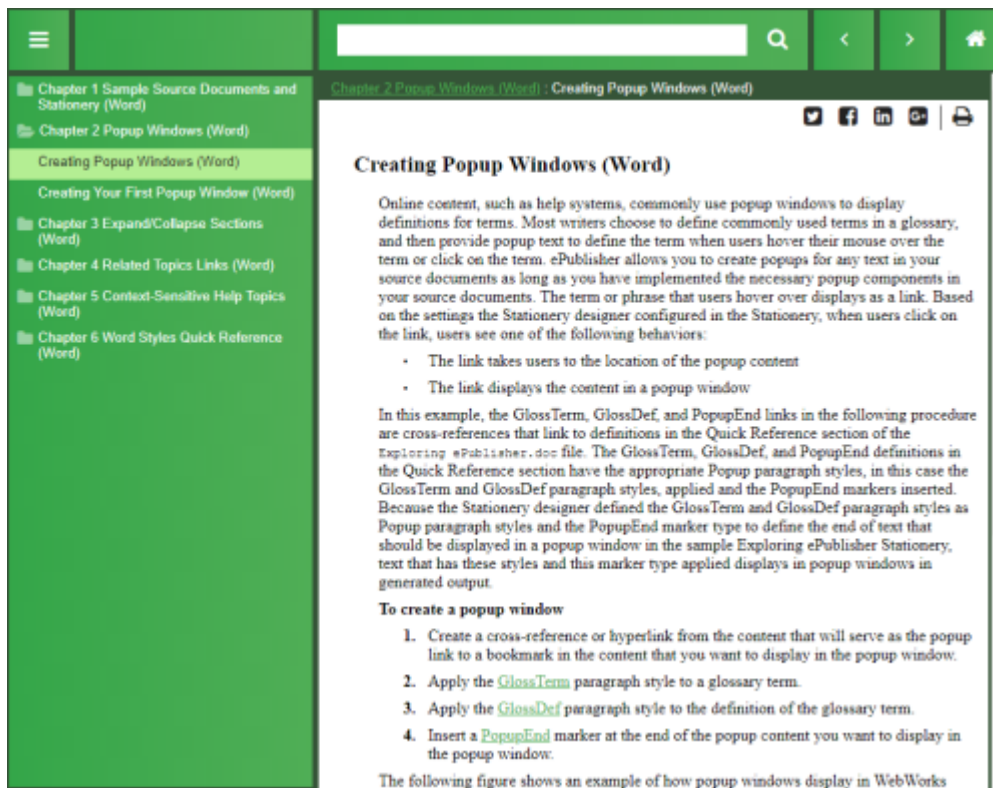
## Corporate

The Corporate skin has been brought into the WebWorks Reverb 2.0 Format, with an updated yet familiar look to it. This skin has a polished look and would suit the needs of a high-profile technology company's help set.



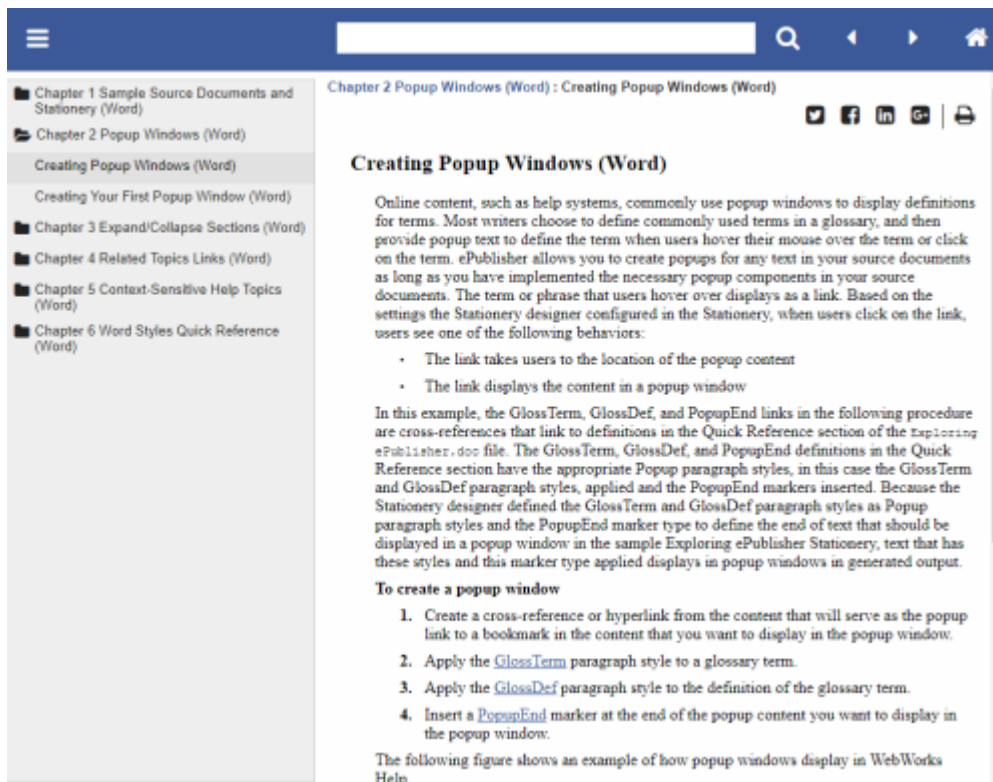
## Metro

The Metro skin has returned in WebWorks Reverb 2.0. This skin was designed in reference to the Windows 8.1 Metro interface design pattern. The Metro skin is useful for users wanting migrate from WebWorks Reverb and have a starting point that looks and feels like their previous Metro skin.



## Social

The *Social* skin has been designed to provide a user experience most similar to that of social networking websites. It provides a familiar interface that is intuitive and casual enough to maximize your end-user participation.



## Using SASS To Customize Reverb Interface

The WebWorks Reverb 2.0 Format uses a technology called SASS to present the user with a dynamic and responsive visual experience. To summarize, SASS is useful because it lets the designer make use of devices that are typically not available in the CSS language. SASS lets the user create and reuse variables across their style sheet, as well as create functions and mixins to enable efficiency and ease of access that other programming languages typically get to enjoy. Once a SASS design has been created, it is then processed and transcompiled to CSS for use in HTML layouts.

Because of the benefits of this, the WebWorks Reverb 2.0 design process is able to be simple and robust. With a large collection of intuitively-named variables, users are able to change settings like fonts and colors in one spot and watch it propagate throughout the layout.

For those interested, SASS is well documented and WebWorks' implementation of this technology conforms with the standards implemented by the creator. Below are a list of resources that may be useful in priming the user to interacting with SASS for the first time.

SASS Basic Guide: <https://sass-lang.com/guide>

SASS Reference: [https://sass-lang.com/documentation/file.SASS\\_REFERENCE.html](https://sass-lang.com/documentation/file.SASS_REFERENCE.html)

CSS Tutorial: <https://www.w3schools.com/Css/>

## Using Custom SASS files in Reverb Projects

The WebWorks Reverb 2.0 Format now allows you to use your own custom SASS files for your projects which allows for even more customization. To do this simply add your SASS file as a Format or Target override.

## Previewing Reverb Output

You can view the generated files directly in your browser, however, the social-media functionality will not be present when viewing WebWorks Reverb 2.0 files in this manner. In order to preview the social-media functionality, you will need to view the files from a web server. With ePublisher, you can preview your WebWorks Reverb 2.0 files through a web server without having to configure a separate web server.

To preview the output, simply select the top level group in the **Document Manager**. This will display an entry in the **Output Explorer** called: `View Output`. Double-click on this entry and your help system will be opened in your default browser connected to a web server built into ePublisher.

## To fully preview WebWorks Reverb 2.0 using the built-in web server

1. Open your project and make sure that your WebWorks Reverb 2.0 target is active and fully generated.
2. In the **Document Manager** select the top level group.
3. In the **Output Explorer** underneath **Merge Output**, double click **View Output**.
4. A browser window will be opened using your default browser and a URL to a `localhost` web address will be displayed. You can now browse the entire help volume as if it were deployed on a dedicated web server.

## Delivering Reverb Output

When you generate output for your project, ePublisher creates the `Output` folder in your project folder, and then creates a folder in the `Output` folder for each generated target. Then, ePublisher creates a folder named for the project itself in the target folder. For example, when you generate output, ePublisher creates the following folder structure:

```
<Project Area>\<Project Name>\Output\<Target Name>\<Project Name>
```

In this folder structure, `<Project Area>` is the name of the ePublisher component you are using, such as `ePublisher Express`, `<Project Name>` is the name of your ePublisher project, and `<Target Name>` is the name of your ePublisher target, such as `WebWorks Reverb`. To deliver your WebWorks Reverb generated output, you need to deliver all the files and subfolders in the `<Target Name>\<Project Name>` folder.

The `<Target Name>\<Project Name>` folder contains the entry-point file, `index.html` by default, which establishes the help set appearance. When the user opens the entry-point file, the browser uses all the files in the `<Target Name>\<Project Name>` folder to display the help, including all the topic files, generated `.css` files, `.pdf` files, images, and WebWorks Reverb components.

## Top-Level Groups in Reverb

Top-Level Groups are used for many organizational and functional purposes in WebWorks Reverb 2.0. Once a help set is generated as a WebWorks Reverb 2.0 Output, Top-Level Groups are converted into a hierarchical structure to form the `Parcels` that make up the Table of Contents, and the structure in the file system. These `Parcels` can then be selected by the user when using Scoped Search to filter search results, and also using the URL method to filter a help set into a subset that contains a specific section of the entire set.

# Searching Output

Use the following guide to assist users in finding terms in your help system:

## Boolean

All search words and phrases have an implicit AND

## Word Search

For example, searching: `eggs bacon` returns all documents containing "eggs" and "bacon"

## Phrase Search

For example, searching `"Good Morning"` returns all documents containing "good" and "morning". However, the two words will not necessarily be adjacent. In other words, at this time, phrase search returns the same results as a multi-word search.

# Using Baggage Files

Baggage files are `PDF`, `HTML` and `ZIP` files that are not part of the ePublisher source files to be converted. ePublisher makes this determination by examining the file extension. If the file extension matches one of the following listed below then it's considered a baggage file.

```
.pdf
.html
.htm
.shtml
.shtm
.xhtml
.xhtm
.zip
```

In order to a file to be processed as a Baggage file ePublisher must know about it. This can be achieved in one of two ways.

- (1) Any link from a source file being converted by ePublisher.
- (2) Entry in the `baggage file info list`. For more information see "Indexing Baggage Files and External URLs".

Baggage Files will be packaged within your Output folder.

**Note:** In **Target Settings > Links > Baggage File Target**, specify whether you want the link to open in the same browser window or in a different window.

**Note:** Set **Target Settings > File Processing > Insert Mark of the Web (MOTW) = Disabled**. Otherwise, the link will fail in Internet Explorer on the local file system. It will work in other browsers. It will work in IE on a web server.

## Indexing Baggage Files and External URLs

With WebWorks Reverb 2.0, files can be indexed to produce as search results with the user's help set. An indexable Baggage File in this context is any PDF or HTML file that is linked from a source document that will be included in the generated output for producing useful search results. For more detailed information on baggage files, see "Targets".

**Note:** In order to determine what baggage files are indexed, ePublisher examines the file extension and if it matches one on the following then it will be indexed.

```
.pdf  
.html  
.htm  
.shtml  
.shtm  
.xhtml  
.xhtm
```

Baggage files are indexed in the same way that source documents are. Indexable baggage files will be indexed as long as the **Index baggage files** Target Setting is **Enabled**. External URLs will be downloaded & indexed as long as the **Index external links** Target Setting is **Enabled**.

## Using Tidy for Indexing HTML Pages

In order to index an HTML baggage file, Reverb creates an XHTML copy of the file using Tidy (tool for cleaning up HTML files) to get a valid XML file that ePublisher can read. As useful as Tidy is, there may be times where it does not recognize a tag or generates something improperly. Tidy is configurable and can be adjusted to convert the HTML in the proper way.

When Tidy does not recognize a tag in an HTML file, an error like the following is produced:

```
line 33 column 3 - Error: <not_recognized_tag> is not recognized!
```



This error means that Tidy wasn't able to generate an XHTML copy of the HTML file, and therefore ePublisher won't be able to index it as a baggage file. With the right adjustments, this can be fixed.

## Configuring Tidy To Recognize New Tags

1. Go to your Tidy directory under the installation directory in your local computer: `...\WebWorks\ePublisher\<VERSION>\Helpers\tidy\`
2. Create a Format override of this helper. To do this: in the sub-folder of your project called: `Formats`, where the Format overrides live, create a new folder called `Helpers` and copy the entire folder called `tidy` (from step 1) to this new folder.
3. In the newly created `tidy` folder, open your `config.txt` file.
4. Depending on the kind of tag you want to add, you'll have to uncomment line 8 or 10, or maybe both in the `config.txt` file.
5. Substitute the placeholder we put there and after the colon, with your new tag name (for example: `not_recognized_tag`).
6. Save and close the file.

To know more about how to customize Tidy go to <https://www.w3.org/People/Raggett/tidy/>.

## Assigning Relevance Weight to Your Source Documents Styles

Search results are displayed in the Search tab when a user types a word to search for. The search results are sorted by a relevancy ranking, which, in the case of source documents, is calculated based on the **Search relevance weight** option defined in your **Paragraph** and **Marker Styles**. By default, WebWorks Reverb 2.0 assigns relevance weight of 1 to all styles.

## To Modify the Relevancy Ranking in Source Documents for Search Results

1. Open your project with ePublisher Designer.
2. Scan the document, to pull all styles into the Style Designer.
3. Open the **Style Designer** (**F10** or **View > Style Designer**).
4. Select the style you want to assign a weight to (either in **Paragraph Styles** or **Marker Styles**).
5. Open the **Options** window.
6. Change the **Value** of the **Search relevance weight** option to a decimal number you determine or you can just ignore it (which is going to be 0), meaning that the style is not going to be shown in your results.

## Assigning Relevance Weight to Your HTML and PDF Baggage Files

The search results are sorted by relevancy ranking, which, in case of HTML baggage files, is calculated based on the scoring preference defined for the HTML tags in the `search_settings.xml` file. By default, WebWorks Reverb 2.0 assigns relevancy rankings based on where in a topic a particular item is found.

## To Modify the Relevancy Ranking in Baggage Files for Search Results

1. Open your project with ePublisher Designer.
2. ***If you want to override the relevancy ranking for all WebWorks Reverb 2.0 targets***, create the `Formats\WebWorks Reverb 2.0\Transforms` folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
3. ***If you want to override the relevancy ranking for one WebWorks Reverb 2.0 target***, create the `Targets\WebWorks Reverb 2.0\Transforms` folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
4. Create a customization of your `search_settings.xml` file.
5. You'll see the following block of code:

```
<Settings version="1.0" xmlns="urn:WebWorks-Settings-Schema">
```

```
<ScoringPrefs default-weight="0.05" pdf-weight="0.05">
```

```
<meta name="keywords" weight="1.0"/>
```

```
<meta name="description" weight="1.0"/>
```

```
<meta name="summary" weight="1.0"/>
```

```
<title weight="1.0"/>
```

```
<div class="myclass" weight="0.05"/>
```

```
<div weight="0.05"/>
```

```
<h1 weight="0.1"/>
```

```
<h2 weight="0.1"/>
```

```
<caption weight="0.1"/>
```

```
<h3 weight="0.1"/>
```

```
<th weight="0.1"/>
```

```
<h4 weight="0.1"/>
```

```
<h5 weight="0.1"/>
```

```
<h6 weight="0.1"/>
```

```
<h7 weight="0.1"/>
```

```
<p weight="0.05"/>
```

```
</ScoringPrefs>
```

```
</Settings>
```

6. Modify the `weight` attributes for any tags, such as `h1` and `h2`, you want to change. You can also specify additional tags with or without class attributes to further refine weights for your HTML baggage files. You may use decimal values to modify the `weight` attribute value.

**Note:** If you wish to set a default weight to tags that are not defined in this file simply update the `default-weight` attribute value.

**Note:** You can change the default weight for all of the text in a PDF file by changing the `pdf-weight` attribute value.

7. Save and close the `search_settings.xml` file.

8. Regenerate your project to review the changes.

## Search Highlighting in Baggage Files

When you click on a result in your Search Results, you'll open the associated source document or baggage file. If what you are clicking is a baggage file and you want to get the highlighting feature in your baggage file, you'll have to copy next to your file and then reference the `reverb-search.js` script in the `<head>` tag of your HTML file. The `reverb-search.js` file lives in the installation directory at `...\WebWorks\ePublisher\<VERSION>\Formats\WebWorks Reverb 2.0\API\reverb-search.js`.

## To Reference the `reverb-search.js` File From Within Your HTML File (After Copying the Script Next to Your HTML File)

1. Open your HTML document.
2. Locate the `<head>` tag.
3. Create the following line inside the `<head>` tag, pointing to the script you just copied:

```
<script type="text/javascript" src="../../../reverb-search.js"></script>
```

4. Save your HTML file.
5. You can either Enable in your **Target Settings** under **Baggage Files** the **Copy baggage file dependents** (this will copy the script to the Output folder, see "Copy baggage file dependents"), or you can manually copy the file to the Output directory, next to your baggage file.

## Searching with URL Method

Search actions can be initiated via URL. For example you can go to the link here:

<http://www.webworks.com/Documentation/Reverb/index.html#search/ePublisher>

When clicked, the link above searches for the term "ePublisher". This can be changed to any search term, which will produce results accordingly.

## TOC or Index with URL Method

You can toggle between the Table of Contents or the Index in the Reverb 2.0 output. For example, you can go to:

<http://www.webworks.com/Documentation/Reverb/index.html#toc/>

[http://www.webworks.com/Documentation/DITA\\_1.2\\_Specification/index.html#index/](http://www.webworks.com/Documentation/DITA_1.2_Specification/index.html#index/)

Clicking on these links will bring up that part of the help as opposed to the regular view.

## Launching Context Sensitive Help for WebWorks Reverb 2.0

WebWorks Reverb 2.0 allows you to open a specific topic. Context sensitive help is a way to identify a topic by using a standard location combined with a topic identifier (topic alias marker). For example, you can go to:

<http://www.webworks.com/Documentation/Reverb/index.html#context/welcome/whatsnew>

You can also use just the topic name without the group context.

<http://www.webworks.com/Documentation/Reverb/index.html#context/whatsnew>

## End-user requirements in WebWorks Reverb 2.0

End-users must have JavaScript enabled in their browser. If not, the user will be prompted with a localized message asking them to enable JavaScript to view the content. Local deployments must have DOM storage enabled in the browser. Web server deployments do not require DOM storage to be enabled. However, some features will be disabled.

End-users must also consume a WebWorks Reverb 2.0 help set with a browser that is not prior to Internet Explorer 11. Modern browsers are recommended for the best possible experience. If an end-user tries to access a WebWorks Reverb 2.0 help set with an unsupported browser, they will be presented with a message instructing them that the browser they are using is not supported.

It is possible to customize these messages by overriding the proper Page Templates (`.asp`).

## Analytics Event Tracking in Reverb 2.0

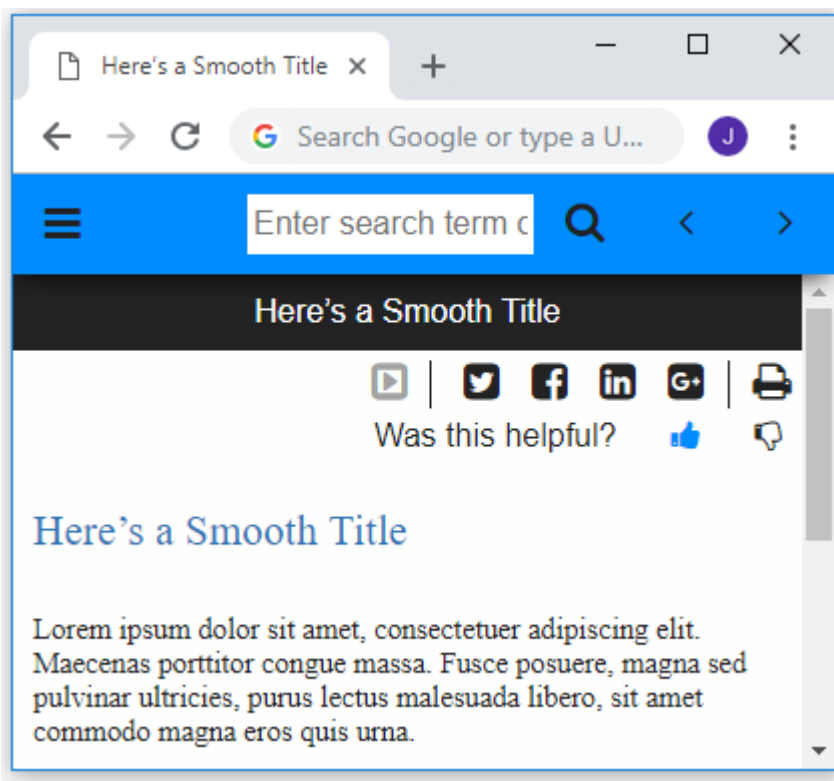
In WebWorks Reverb 2.0, analytics has been expanded to anonymously track several types of events that better allow you to understand the performance of your content with your end-users. The updated analytics features include:

- All page views are tracked with easy to read page names.
- New categories and labels have been introduced to highlight high performing and low performing pages in your reports. For more information, see “Was This Helpful?” Buttons”.
- All Reverb interface components, such as toolbar buttons and menu items have their interactions tracked as events.
- All search queries, inputs, cancellations, and search result views are tracked and recorded.

- Search page results including the **Was This Helpful?** buttons are tracked for easy reporting. For more information, see “Was This Search Helpful? Buttons”.

## ‘Was This Helpful?’ Buttons

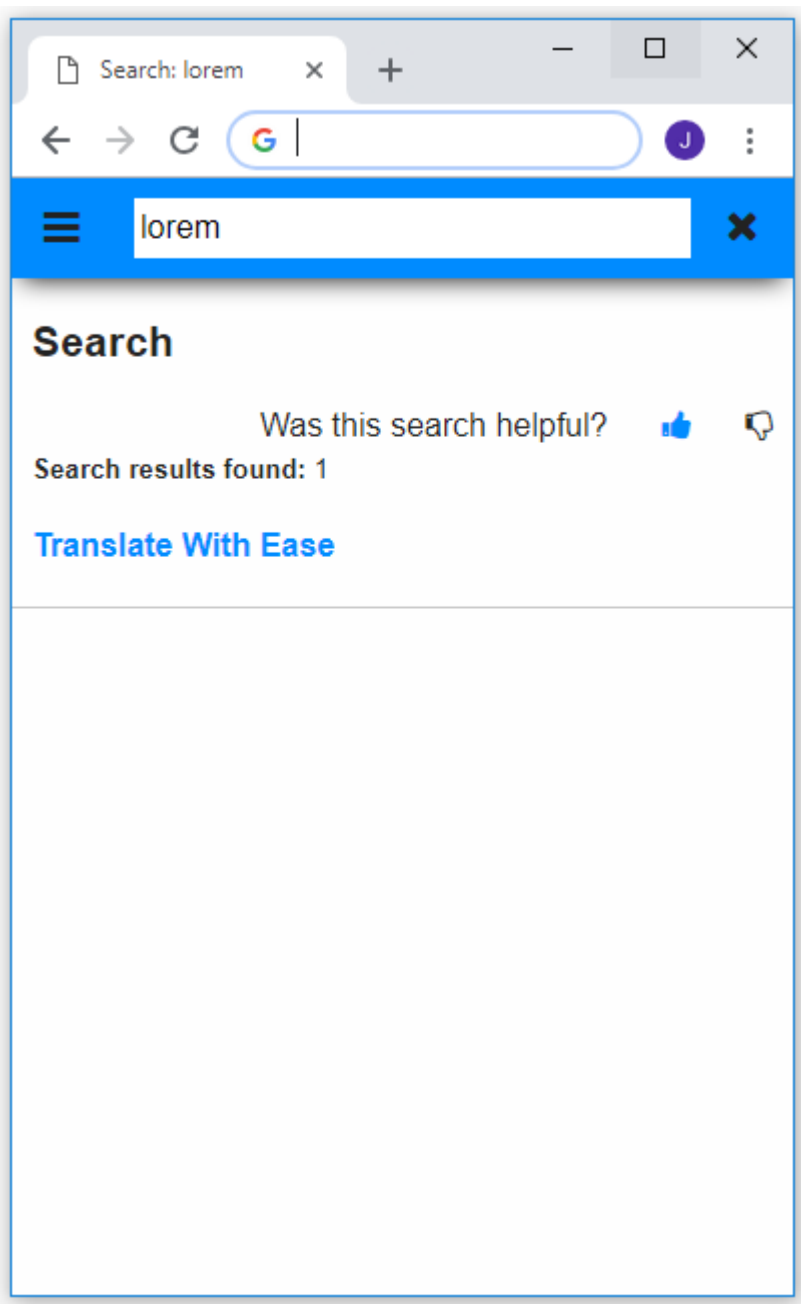
The **Was This Helpful?** buttons feature allows you to provide your end-users with the ability to give anonymous feedback about their experience. The **Was This Helpful?** buttons feature can be used to record feedback on the current page being viewed. A Google Tracking ID is required for these buttons to record analytic events.



## ‘Was This Search Helpful?’ Buttons

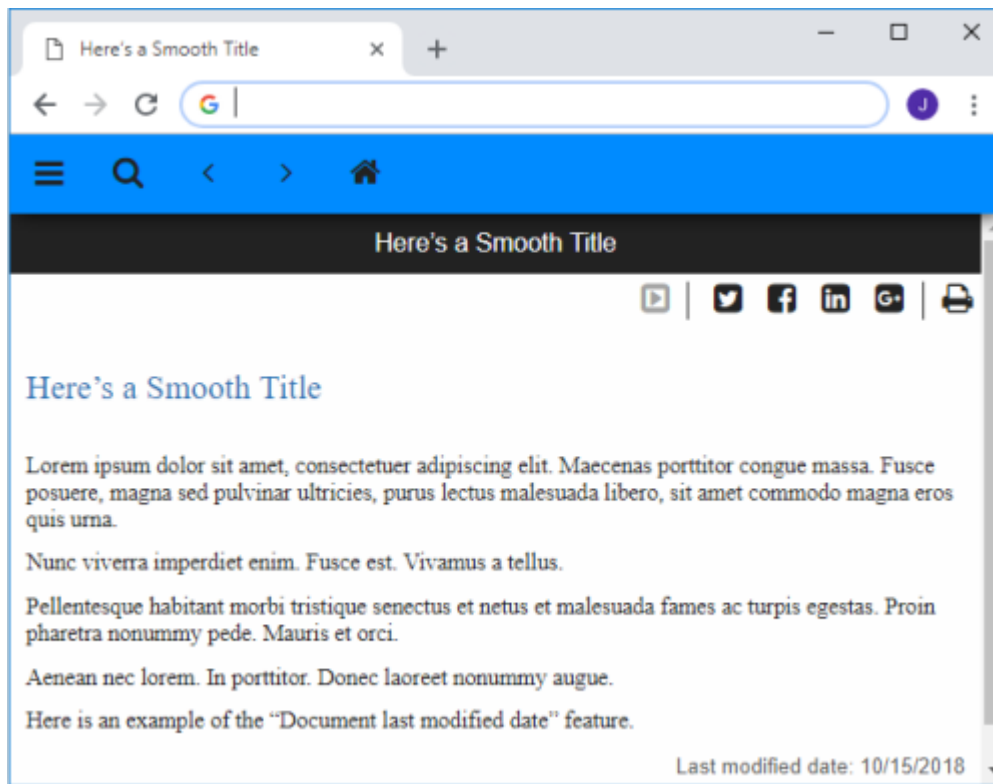
The **Was This Search Helpful?** buttons feature allows you to provide your end-users with the ability to give anonymous feedback about their search results. The **Was This Search Helpful?** buttons feature can be used to record feedback and search queries of the search results page. A Google Tracking ID is required for these buttons to record analytic events.





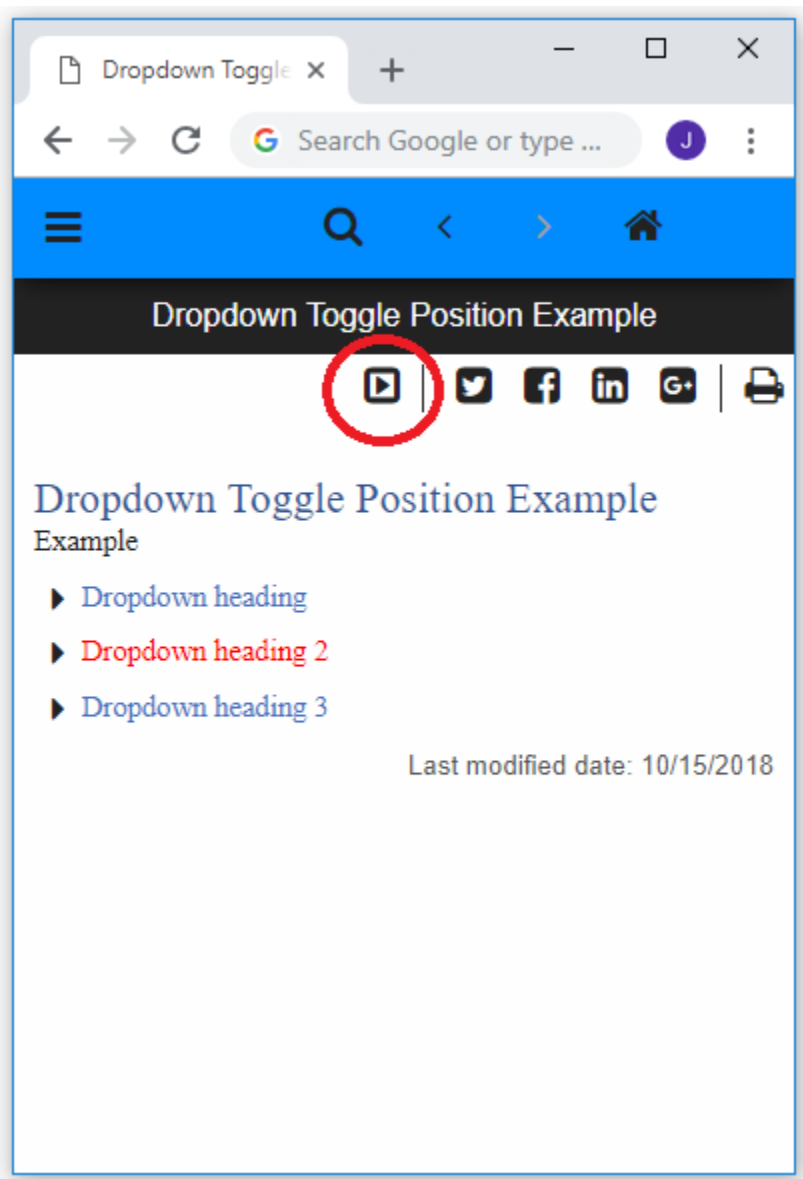
## Document Last Modified Date in Reverb

In WebWorks Reverb 2.0, there is a target setting that allows you to display the **Last modified date** of each document in your generated output pages. If you enable this target setting, your end-users will be able to see the date of that last time the document that generated a particular page was edited. If you examine the image below, you can see the date string in a light colored gray.



## Drop-down Expand/Collapse All Toggle Button

In WebWorks Reverb 2.0, you can centrally control all of the drop-down paragraphs on a page using the **Dropdown Expand/Collapse Toggle** button. It is enabled by default, but can be disabled in the target settings. The image below highlights the toggle button.

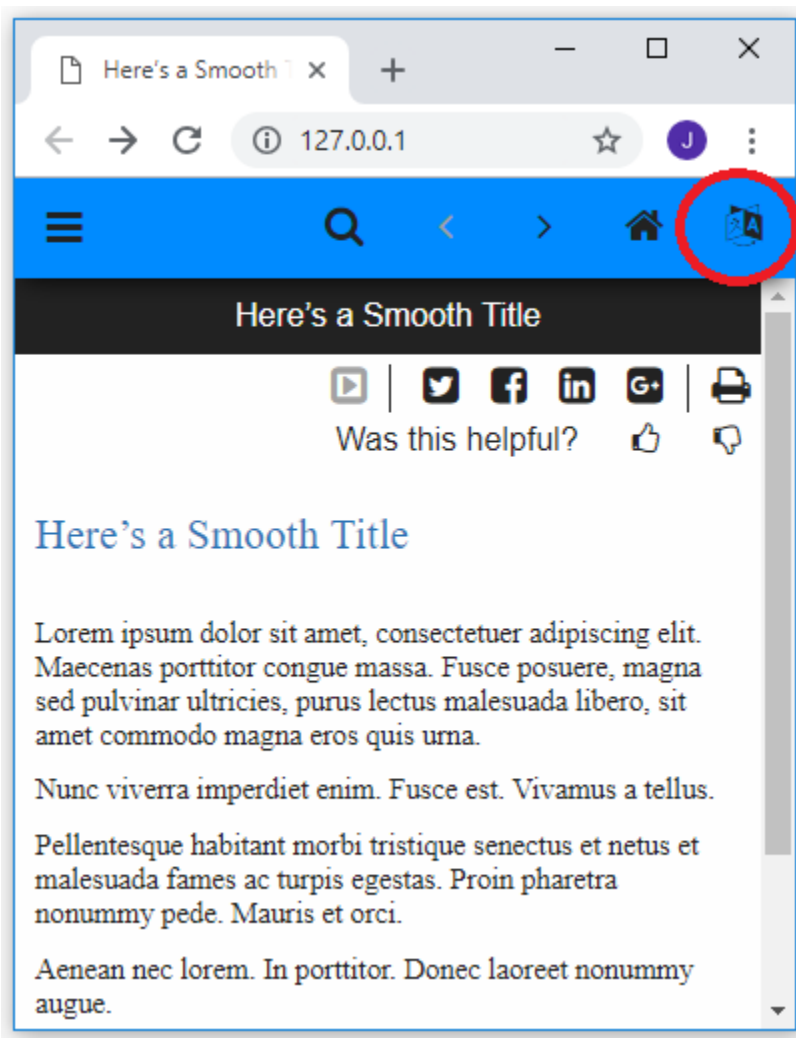


## Google Translate Button

In WebWorks Reverb 2.0, you can use the target settings to enable a Google Translate button to be part of your toolbar. When your end-users select this toolbar button, all of the text in the content area will be translated using Google's Translate web service. The navigation menus however are not translated, just the content area. This feature will only be displayed when the output is deployed to a web server, which is a requirement of the Google Translate web service.

**Note:** As an added benefit, Reverb 2.0 is smart enough to know when to display the Google Translate button and when to suppress it. By not displaying the Google Translate button when loading directly from a file

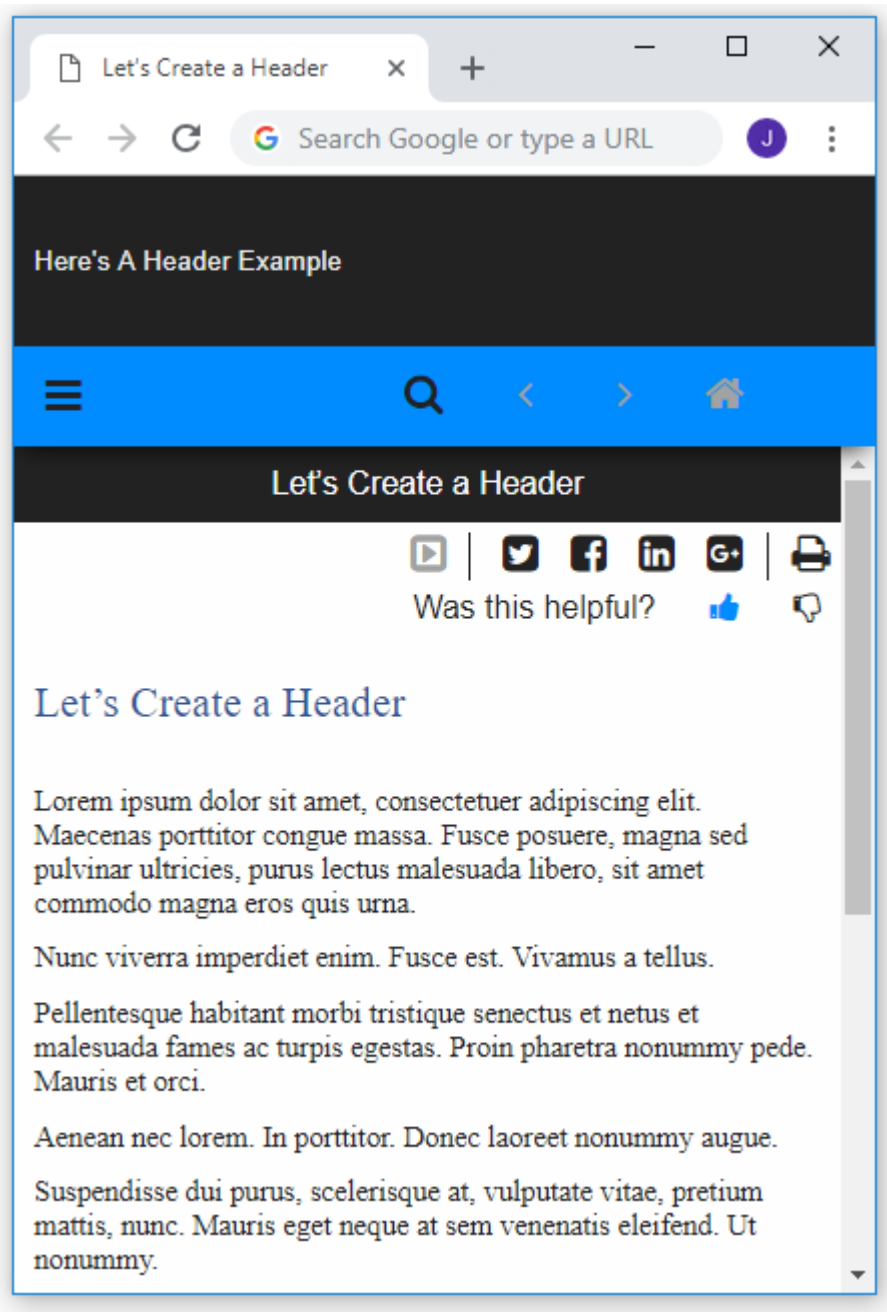
system (i.e. as packaged help), you will need only one configuration (target) to support both your web deployment and your packaged help deployments.



## Customizable Header and Footer

In WebWorks Reverb 2.0, using target settings, a customizable Header and Footer can be enabled. Each of these parts uses its own ASP file that can be customized. In addition, you can specify that company information be displayed or another logo image of your choosing.

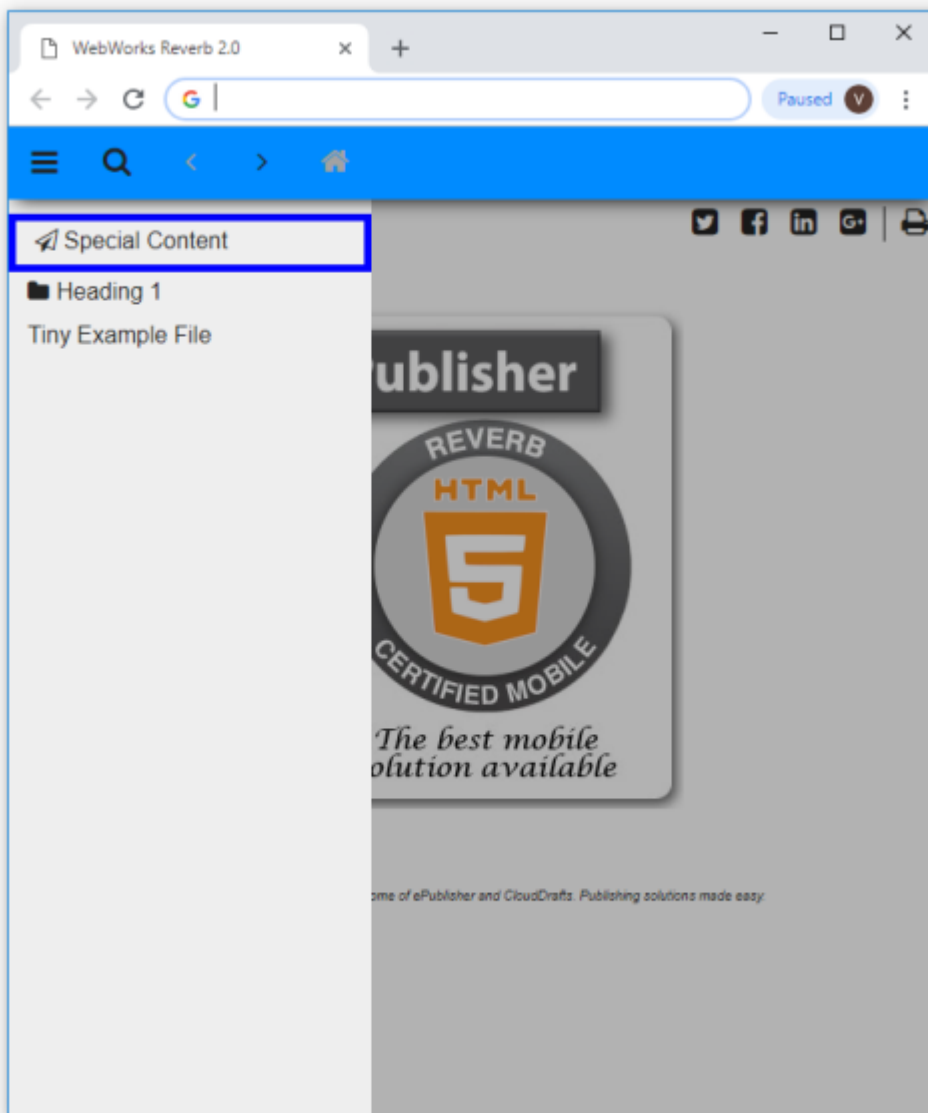
The following image demonstrates a custom header above the Reverb 2.0 toolbar.



## Custom TOC Menu Items

In WebWorks Reverb 2.0, you can customize specific TOC (table of contents) Menu items based the target paragraph that the item links to. For example, if a TOC item links to a paragraph that contains a marker assigned to the ePublisher marker type called: TOC Entry class, then that TOC item will inherit the value of the marker as a CSS class name. Using the class name, you can completely change the appearance

of the TOC item in the Menu. See the image below for an example of a TOC Menu item that has a different icon and blue borders.



## Customizing a Bullet Icon using Font Awesome

It is possible to use a Font Awesome icon for the bullet in a list. In order to do this, take the following steps:

1. In the Style Designer, create a new Character Style that will represent the Font Awesome bullet.

2. In the Character Style Options, under Additional CSS Classes, add `fa`, a whitespace, and the class of the icon you want to use. (For example: `fa fa-warning`) For a reference of all the icons available, refer to the Font Awesome Cheatsheet at:  
<https://fontawesome.com/v5/cheatsheet/free>
3. Select the Paragraph Style that is to have the bullet added.
4. In the Bullet Properties area for the Paragraph Style, under the Property called Character Style, select the name of the Character Style that has the Font Awesome icon applied to it.
5. (Optional) If the Paragraph Style was initially an unordered list in the source document, it is also necessary to add a single whitespace to the Bullet Property called Text so the default bullet does not emit from the source document.

## Using the `url_maps.xml` reference file

In the Output Explorer, the `url_maps.xml` file can be used to view the links available in an output helpset. When opened in a text editor, this file contains all of the available links in the output. This can be useful for web designers or anyone needing a reference to use these links externally, such as on a website or a custom page. The file can also be used for sitemap generation if you know how to work with XML.

The TopicMap element contains all of the Topic Alias Links in the output. The PageMap element contains all of the Page URL links available in the output.

# PDF - XSL-FO

Why use PDF - XSL-FO output format?

PDF-XSL-FO Page Regions

PDF XSL-FO Font Inclusions

With ePublisher, you can quickly and easily create PDF output from Adobe FrameMaker, Microsoft Word, and DITA-XML source documents. Based on your requirements, you can either create PDF output so that it mirrors the styling and layout of your source document's pre-configured settings (using the *PDF* output format), or you can choose to manage all these settings through ePublisher using the *PDF - XSL-FO* output format.

## Why use PDF - XSL-FO output format?

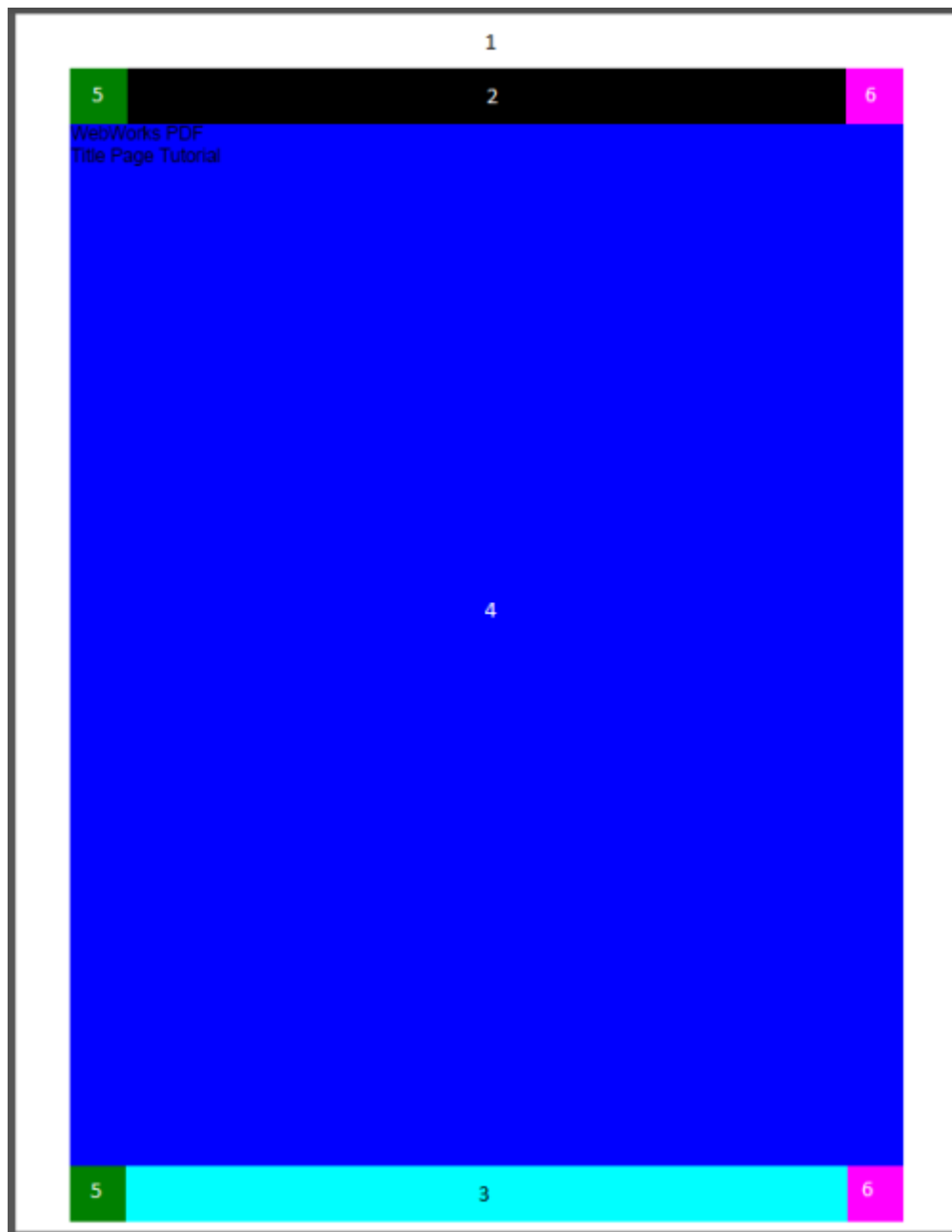
If you want to keep your original document's style and layout settings, then you will want to configure ePublisher to use the *PDF* output format. However, if you want to directly control your document's style and layout settings, then you will need to configure ePublisher to use the *PDF - XSL-FO* output format.

The PDF - XSL-FO format creates a PDF file for each of your source files, a single PDF file containing all source files within a group, or both. PDF - XSL-FO is recommended for producing a PDF output file that conforms to the settings you configure in ePublisher. These settings can be controlled independent of your source file's authoring environment or they can be used to work in conjunction with your source file's paragraph and character level settings. This format allows you to create and deliver PDF files that conform to the style and layout settings that you can centrally control in ePublisher. This format may be especially useful when mixing documents from different input formats or different layouts and/or styles.

## PDF-XSL-FO Page Regions

When creating a PDF-XSL-FO output, it can be useful to understand the regions of the page when styling a PDF document.





The numbered regions corresponding with the list items below, which can be modified in the Page Styles section of the Style Designer.

- 1.** Master Page Margins (Top, Left, Right, Bottom).
- 2.** Master Page: Before Region - This is the region where a header will be rendered

3. Master Page: After Region - This is the region where a footer will be rendered
4. Master Page: Body Region - This is where the content of the page is rendered
5. Master Page: Start Region
6. Master Page: End Region

**Note:** It should be noted that the Start and End regions are actually rectangular regions that span from the top to the bottom of the page. In this image the Body Region is covering the middle of the Start and End regions, giving the appearance of two small squares in the corners of the document.

The **Extent** property can be used to set the sizing of these different regions, with the exceptions of the Master Page margins and the Body Region.

## PDF XSL-FO Font Inclusions

To ensure non-standard fonts are included into your generated PDF, you may need to modify the Apache FOP configuration files, `apache-fop-2.6.xconf`.

To do this, you must either create a Format or Target Override, Create the override in one of these places, for more information on overrides, Depending on how you prefer to set up your project, you will create one of the following:

Per target:

```
Targets\<target name>\Helpers\apache-fop-2.6.xconf Targets\<target
name>\Helpers\apache-fop-2.6.xconf
```

Per format:

```
Formats\<format name>\Helpers\apache-fop-2.6.xconf Formats\<format
name>\Helpers\apache-fop-2.6.xconf
```

Project-wide:

```
Formats\Helpers\apache-fop-2.6.xconf Formats\Helpers\apache-
fop-2.6.xconf
```

Once the override has been created, open the file in a text editor of your choosing, and you will see the following markup:

```
<fonts>
  <!-- Example -->
  <!--
    <font kerning="yes" embed-url="file:///C:/Windows/Fonts/
REFSAN.TTF">
```

```

        <font-triplet name="Microsoft Sans Serif" style="normal"
weight="normal" />
    </font>
    <font kerning="yes" embed-url="file:///C:/Windows/Fonts/
REFSAN.TTF">
        <font-triplet name="Microsoft Sans Serif" style="italic"
weight="normal" />
    </font>
    -->
    <!-- automatically detect operating system installed fonts -->
    <auto-detect/>
</fonts>

```

Modify the lines in between the comments to reflect the font you are wanting and the directory path in which it is located. Save this file, and now you can add them by using ePublisher Designer.

### To add fonts in ePublisher Designer

1. In ePublisher Designer, select the **View -> Style Designer** menu
2. With the **Paragraph Styles** visible, select the **[Prototype]** style
3. On the **Properties** tab, select **Font** then click the icon for **Family**
4. If your fonts are not listed, then manually add the names of your desired font using the **Custom Font Family** text box
5. Select **OK**

# Dynamic HTML

Dynamic HTML Output Viewer  
Delivering Dynamic HTML

There are several HTML-related standards, such as HTML 3.2, HTML 4, and XHTML 1.0. HTML 4 is also referred to as Dynamic HTML (DHTML). In addition, there are multiple browsers and device types that can display content based on these standards. With all these variables, getting the content you need formatted the way you need it for your specific environment and requirements is critical. ePublisher provides the Dynamic HTML output format to allow you to generate the HTML-based output you need. You can also customize this formatting to meet your requirements.

You can use the Dynamic HTML format to produce XHTML output that conforms to XHTML 1.0 standards and uses cascading style sheets that conform to the CSS1 standard. XHTML became a W3C recommendation in 2000. Dynamic HTML is recommended to produce output that you will publish on a Web server and provide to users running a Web browser, such as Firefox or Internet Explorer. You can also customize the Dynamic HTML output format to create a powerful, full-featured web site.

As the standards evolve, browsers and device platforms adjust to support the newer standards. With mobile devices, additional platforms and browsers have been introduced, which can complicate the decision about which standards your content can use.

The Dynamic HTML output format allow you to generate HTML content to integrate into your Web site. You can also create content for HTML-based release notes and content for hand-held devices, such as PDAs. These output formats provide the flexibility and control you need, with the ability to add a basic table of contents, index, and browse navigation. You decide whether to use all the aspects of DHTML and XHTML in the Dynamic HTML output format, or to use simplified HTML to support a wide range of browsers and platforms, including mobile devices and PDAs.

The Dynamic HTML output format produces HTML content that conforms to the HTML 4 and XHTML 1.0 standards, and uses cascading style sheets that conform to the CSS1 standard. Dynamic HTML (DHTML) is a collection of technologies developed to make HTML more dynamic and interactive. DHTML uses the following technologies to give the content developer control over the appearance and behavior of HTML elements in a browser window:

## Static markup language (HTML 4)

HTML 4 extends HTML with mechanisms for style sheets, scripting, frames, embedding objects, improved support for right-to-left and mixed-direction

text, richer tables, enhancements to forms, and improved accessibility for people with disabilities.

### **Presentation definition language (cascading style sheets)**

Cascading style sheets (CSS) provide style definitions, such as fonts, colors, spacing, and positioning to HTML documents.

### **Client-side scripting language, such as JavaScript**

JavaScript and other scripting languages provide compact, object-based scripting support for developing client and server Web applications.

### **Document Object Model**

The Document Object Model provides a standard that allows programs and scripts to dynamically access, process, and modify the content of a page.

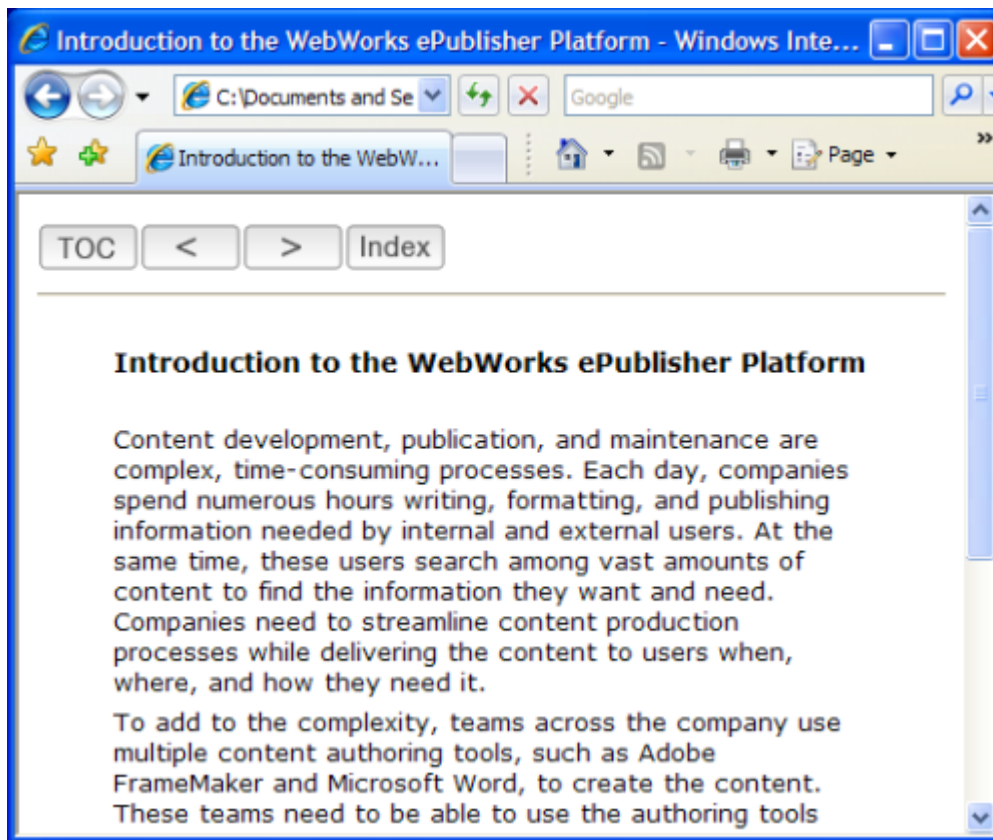
XHTML is an abbreviation for Extensible Hypertext Markup Language. XHTML 1.0 is similar to HTML 4, with tagging rules that conform to the requirements of XML. If you modify the page templates or styles in a Dynamic HTML project, make sure your changes conform to the XHTML requirements for future maintenance. However, as long as you create valid HTML, most current browsers can correctly display your output.

To determine whether the Dynamic HTML output format is what you need, review the following considerations:

- If all your users have current browsers and their viewing environment is not restricted, use the Dynamic HTML output format.
- If all your users have current browsers and you want to provide enhanced navigation controls, such as an expand/collapse table of contents, full text search, and related topics buttons, consider using the WebWorks Help or WebWorks Reverb output format.

## **Dynamic HTML Output Viewer**

Dynamic HTML does not provide a multi-pane viewer. This output format is displayed in a browser. By default, ePublisher adds a navigation bar at the top of the files it creates. ePublisher also creates a table of contents page and an index page.



You can specify whether to include the navigation bar at the top or the bottom of the page. You can also add company information to the page, and define the table of contents and index pages.

## Delivering Dynamic HTML

When you generate output for your project, ePublisher creates the `Output` folder in your project folder, and then creates a folder in the `Output` folder for each generated target. Then, ePublisher creates a folder named for the project itself in the target folder. For example, when you generate output, ePublisher creates the following folder structure:

```
<Project Area>\<Project Name>\Output\<Target Name>\<Project Name>
```

In this folder structure, `<Project Area>` is the name of the ePublisher component you are using, such as `ePublisher Express`, `<Project Name>` is the name of your ePublisher project, and `<Target Name>` is the name of your ePublisher target, such as `Dynamic HTML`. To deliver your Dynamic HTML generated output, you need to deliver all the files and subfolders in the `<Target Name>\<Project Name>` folder.

The `<Target Name>\<Project Name>` folder contains the entry-point file, `toc.html` by default, which displays the table of contents. When the user opens the entry-

point file, the browser uses all the files in the *<Target Name>\<Project Name>* folder to display the help, including the topic files and images.



# ePUB

ePUB Platforms  
ePUB Considerations

ePublisher can deliver content to your mobile users and eReader platforms with the IDPF ePUB 2.0 eBook standard. This format makes sense for long form content or reference materials which are not suitable for web delivery. eBooks enable users to access content offline, track their reading progress, and otherwise enjoy the benefits of traditional press books.

## ePUB Platforms

ePUB eBooks can be accessed and viewed on a variety of desktop and mobile devices. To ensure maximum usability, this release was tested against the following ePUB readers:

- Adobe Digital Editions  
<http://www.adobe.com/products/digitaleditions/>  
Microsoft Windows
- Apple iBooks  
<http://itunes.apple.com/us/app/ibooks/id364709193?mt=8>  
Apple iPad, Apple iPhone, Apple iPod Touch
- Lexcycle Stanza (Apple iPad, iPhone, and iPod Touch)  
<http://www.lexcycle.com/iphone>  
Apple iPad, Apple iPhone, Apple iPod Touch, Microsoft Windows (Beta)

In addition UX tests performed on these eBook platforms, generated output was validated using the publicly available EpubCheck tool during the development cycle.

- EpubCheck 1.0.5  
<http://code.google.com/p/epubcheck/>

By closely tracking the ePUB standard and conducting eReader specific tests, ePublisher attempts to deliver high quality eBooks across a wide range of ePUB capable platforms.

## ePUB Considerations

The ePUB standard specifies a container and page definition syntax. ePUB eBook readers will render this information as best they can given available screen real-estate and platform considerations. Users who wish to maximize the ePUB reading experiences for a broad audience should pay careful attention to ePUB specific details.

# Meta Data

You can specify standard ePUB eBook meta data in the Target Settings dialog.

- Author Name
- Author Name (File As)

# Book Title and ID

All ePUB books contain a unique identifier. ePublisher will synthesize a unique ID for your book, though you may require explicit control over this value. You can specify an explicit ID via the Merge Settings dialog with the Group Context control. Further, localized book titles can be specified in the Merge Settings dialog as well.

# Long Content

The ePUB standard recommends breaking long content into smaller chunks to speed display rendering and reduce memory requirements on mobile devices. This can be accomplished with ePublisher's standard page break controls. You may specify a page break priority for paragraphs and tables in the Options panel of the ePublisher Style Designer.

# Page Styles

Content page styles can be controlled using ePublisher's standard page style support. For the ePUB format, ePublisher also allows users to select a page style for use with the cover and index pages. These styles can be specified in the Target Settings dialog under the Cover and Index sections.

# Tables

By default, ePublisher's ePUB format renders tables using paragraph markup. Users may enable true table markup via the table styles Options panel in the ePublisher Style Designer. Reasons for rendering tables using paragraph markup include:

- All tested eBook readers lacked the capability to view tables larger than the available screen space.
- The Apple iBooks reader, prior to version 2.0, fails to render images inside of table cells.  
ePublisher emits conversion warnings when an image is rendered inside a table cell.

Tables can be used effectively inside of ePUB eBooks, though authors should be aware of the inherent space limitations associated with mobile reading devices. The

addition of proportional (percentage based) table cell widths, used in conjunction with a table width set to 100%, may provide users with sufficient control over table rendering behavior. The behavior can be specified in the Options tab for table styles in the ePublisher Style Designer.

## Cover

ePUB books support the display of a cover page or image. This format allows users to select a cover image via the Target Settings dialog. Additionally, users may customize the cover page by creating overrides for the `Pages\Cover.asp` page template.

When creating your cover page and image, keep the following points in mind:

- eBook icons
  - \_ Adobe Digital Editions uses the cover page for the eBook icon
  - \_ Apple iBooks and Lexcycle Stanza use the cover image for the eBook icon
  - \_ ePublisher will embed the selected cover image into the cover page to ensure correct operation across platforms
- Cover image limitations
  - \_ Certain readers, such as the Apple iBook reader, prefer images with a 3x4 aspect ratio. Otherwise, the eBook will not display a custom icon in iTunes.
  - \_ Certain readers, such as the Apple iBook reader, prefer 600x800 pixel images. Otherwise, the eBook will not display a custom icon in iTunes.

## Syncing with Apple iPad

1. Ensure iTunes 9.1 or greater is installed on your system. Ensure the Apple iBooks application is installed on your iPad.
2. Launch iTunes.
3. Click "Add File to Library" and select your generated ePUB eBook.
4. Attach your iPad.
5. Sync your iPad normally.
6. When synchronization completes, disconnect the iPad from your computer.
7. Launch the iBooks application on the iPad.

- 8.** Within iBooks, you will see your new book on the virtual bookshelf.

These instructions also apply when adding ePUB eBooks to Apple iPhone and Apple iPod Touch devices.

# Eclipse Help

Eclipse Help Viewer  
Delivering Eclipse Help

To deliver help on multiple platforms, you need a solution that runs on all those platforms. This solution helps you avoid complications of each platform and allows you to deliver one solution that meets the needs on all your platforms. This common solution can save development time and effort.

ePublisher supports several Java-based help output formats, such as Eclipse Help, Oracle Help, and Sun JavaHelp, that provide a common solution for multiple platforms without JavaScript support. WebWorks Help and WebWorks Reverb also provide a cross-platform solution, but it requires JavaScript support. If you are delivering a Java-based application, or if your environment does not support JavaScript, a Java-based help solution can help you deliver your help content.

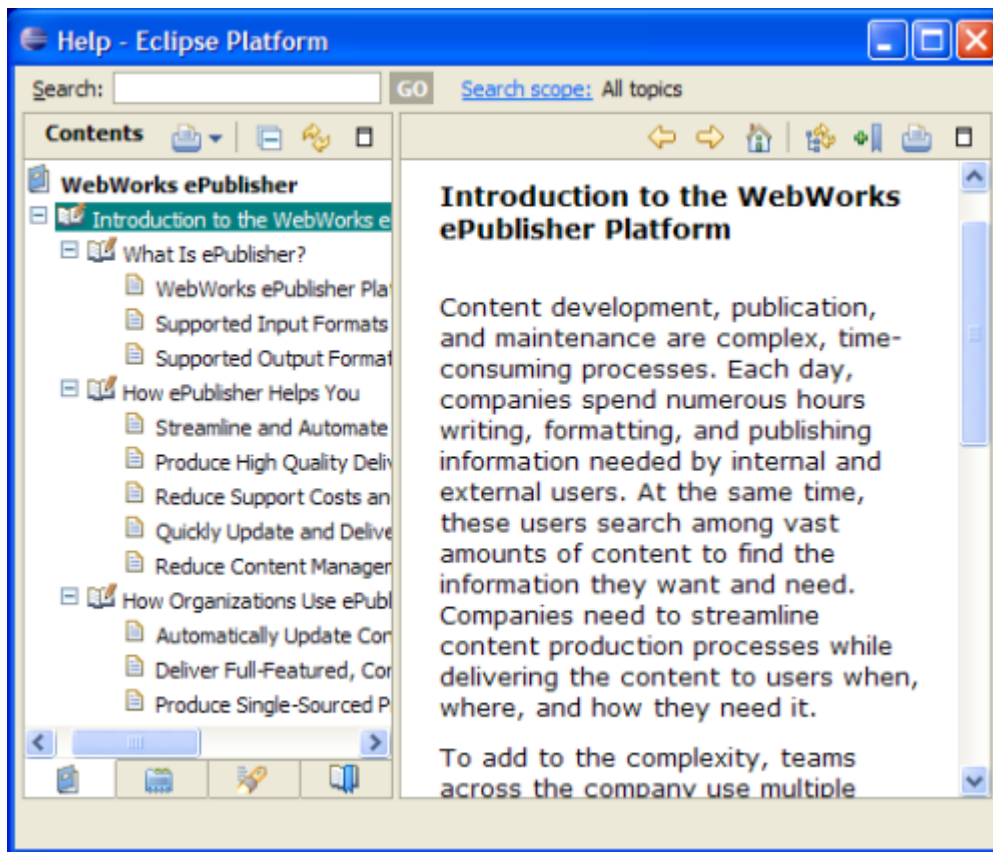
The Eclipse Help format uses a Java-based delivery environment to provide a comprehensive help viewer. Eclipse Help delivers content in HTML files and uses an XML-based table of contents. Once you install the Eclipse platform, you can view Eclipse Help files. ePublisher provides a standalone viewer for Eclipse Help so you can develop and view your Eclipse Help.

Eclipse Help requires the Eclipse integrated development environment (IDE), which makes it a good output format choice for products that already install and use the IDE. For more information about Eclipse Help, see the Eclipse SDK documentation at: [help.eclipse.org/help33/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/ua\\_help.htm](http://help.eclipse.org/help33/index.jsp?topic=/org.eclipse.platform.doc.isv/guide/ua_help.htm).

You can also use Eclipse Help to deliver an Infocenter. You can use all the supported input formats to develop content for an Infocenter. Then, ePublisher allows you to publish that content in the required format. For more information about creating an Infocenter, see [dita.xml.org/wiki/setting-up-the-eclipse-help-infocenter-for-publishing-dita-content](http://dita.xml.org/wiki/setting-up-the-eclipse-help-infocenter-for-publishing-dita-content).

## Eclipse Help Viewer

The Eclipse Help viewer uses an embedded Apache Tomcat server. Similar to other help viewers, the Eclipse Help viewer provides a navigation pane with multiple tabs and a topic pane.



## Delivering Eclipse Help

You can provide a help system in Eclipse Help format either as a collection of individual files or in a single, compressed Java archive `.jar` file. A `.jar` file, similar to a `.zip` file, compresses and stores a collection of files.

When you generate output for your project, ePublisher creates the `Output` folder in your project folder, and then creates a folder in the `Output` folder for each generated target. Then, ePublisher creates a folder named for the project itself in the target folder. For example, when you generate output, ePublisher creates the following folder structure:

```
<Project Area>\<Project Name>\Output\<Target Name>\<Project Name>
```

In this folder structure, `<Project Area>` is the name of the ePublisher component you are using, such as `ePublisher Express`, `<Project Name>` is the name of your ePublisher project, and `targetname` is the name of your ePublisher target, such as `Eclipse Help`. To deliver your Eclipse Help generated output, you can provide the complete contents of the `<Target Name>\<Project Name>` folder to your Java application developers, and they can determine whether to deliver the `.jar` file or the individual files to application users.

# HTML Help

[Benefits of Microsoft HTML Help](#)  
[Restrictions and Requirements for Microsoft HTML Help](#)  
[HTML Help Viewer](#)  
[Topic Only View in HTML Help](#)  
[HTML Help Workshop](#)  
[Delivering HTML Help](#)

Delivering help information in a consistent manner on Windows computers was an important concern for Microsoft. Initially, Microsoft provided the WinHelp format for help content delivery. As the Windows platform advanced, Microsoft introduced the Microsoft HTML Help format. To help authors deliver content through the HTML Help Viewer provided with the Windows platform, Microsoft provides a help compiler and toolkit, HTML Help Workshop, that allows you to build Microsoft HTML Help `.chm` files.

The Microsoft HTML Help output format provides a standard help format for products on computers running the Windows operating system. The Microsoft HTML Help format delivers a single, compiled file that includes multiple source files compressed into one `.chm` file. Unlike the RTF-based Microsoft WinHelp format, the Microsoft HTML Help format is based on HTML/XHTML. Microsoft HTML Help is recommended to produce online help for 32-bit applications that run on a computer running Microsoft Windows 95 or later, including Windows Vista:

Microsoft HTML Help provides a comprehensive help format, including a table of contents, index, full-text search, and favorites in one integrated viewer window. The HTML Help Viewer uses standard Internet Explorer components and supports many Web technologies, such as HTML, ActiveX, Java, JavaScript, JScript and other scripting languages, and the Web image formats, such as `.gif`, `.jpg`, and `.png`.

In HTML Help, writers list a collection of source files in a help project `.hhp` file with other project-related settings. The writers then use HTML Help Workshop to compile the help and create the `.chm` file. During compilation, HTML Help Workshop uses the help project `.hhp` file to determine how HTML topic files, image files, contents `.hhc` files, index `.hhk` files, and any other elements appear in the single, compressed help file. ePublisher automates this process and integrates it into help generation.

## Benefits of Microsoft HTML Help

The Microsoft HTML Help output format generates output files that conform to the HTML 4 and XHTML 1.0 standards. The content uses Cascading Style Sheets that conform to the CSS1 standard. This output format produces all the files required to create the Microsoft HTML Help `.chm` file.

ePublisher streamlines the help generation process. ePublisher uses your Stationery and project settings to transform your source documents to the input files needed

by Microsoft HTML Help Workshop. Then, ePublisher uses the help compiler to build the help project and create the `.chm` file. This integrated process simplifies the authoring environment, but it also creates all the underlying files to help you customize the process, if needed.

Microsoft HTML Help provides several important benefits:

- You have a single, compressed `.chm` output file to deliver.
- Your audience can read your content in the standard Windows HTML Help Viewer.
- The HTML Help Viewer provides a powerful search engine that supports partial words, synonym searching, and other advanced search options.

## Restrictions and Requirements for Microsoft HTML Help

The Microsoft HTML Help output format is recommended to produce standard help for 32-bit applications that run on a computer running Windows 95 or later. For more information, see “Requirements”. Review the following considerations when deciding whether to deliver HTML Help:

- Your audience cannot read the `.chm` file over a network or over the Web. The `.chm` file must be installed on the local computer.
- On some systems, CHMs cannot be viewed if they were generated in an ePublisher project that is located on a network drive. To resolve the problem, move the entire ePublisher project to a local drive, on the same computer where ePublisher is installed. It is OK to keep the source documents on a network drive.
- Some security settings can interfere with `.chm` file use.
- A `.chm` file supports many Web technologies, such as scripting languages, which also allow them to contain and transport viruses and security risks. For this reason, many email systems remove `.chm` files when attached to an email message.

## HTML Help Viewer

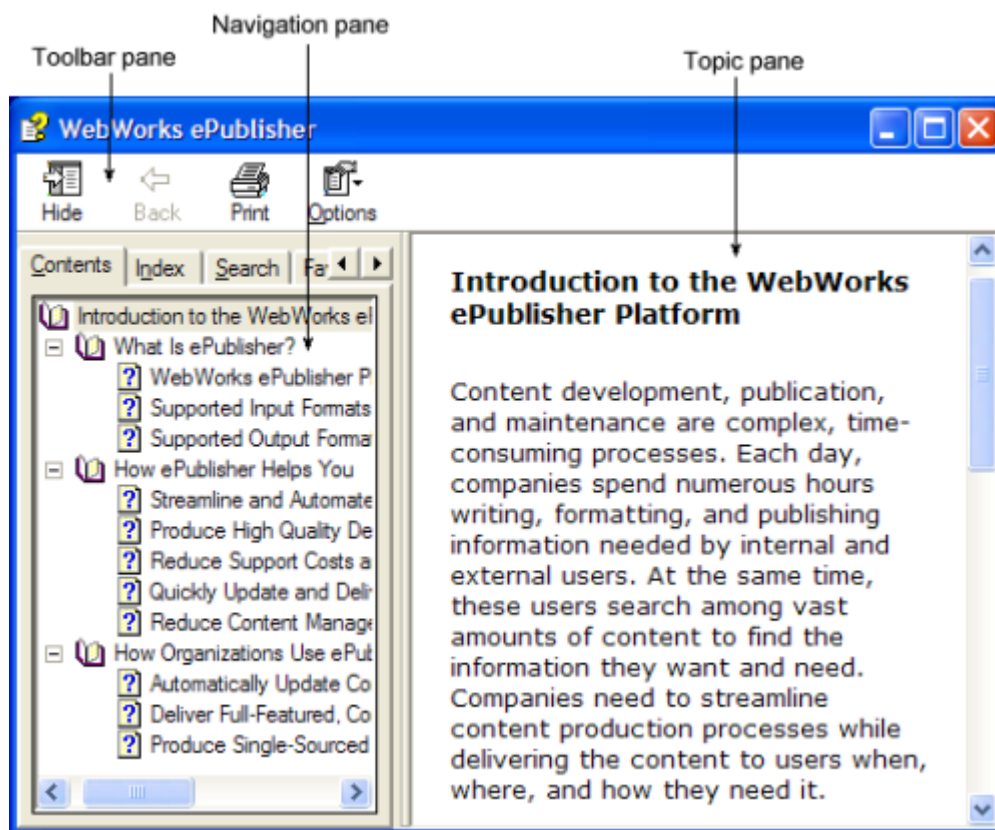
When you open a Microsoft HTML Help `.chm` file, Windows displays the content in the Help Viewer window. Microsoft HTML Help Workshop allows the writer to compile and customize the `.chm` file. ePublisher uses your source documents to generate all the files needed to compile the `.chm` file.



By default, the tripane HTML Help Viewer includes a table of contents, index, full-text search, and favorites in one, integrated viewer window. You can customize this viewer window. The user also has options to show or hide the navigation pane, based on the options you defined.

The default tripane view includes the following panes:

- The **toolbar pane** displays buttons that provide additional functions and navigation options to the user. HTML Help Workshop allows you to select which buttons to include in your help file.
- The **navigation pane** displays tabs to access the table of contents, index, full-text search, and favorite topics. HTML Help Workshop allows you to select which tabs to include in your help file.
- The **topic pane** displays the information contained in the source documents.



## Toolbar Pane in HTML Help

The toolbar pane provides several buttons that provide additional functions, such as returning to the previously viewed topic, showing and hiding the navigation pane, and printing a help topic. The writer can customize which buttons are included

in the toolbar pane through HTML Help Workshop. The following table lists the supported buttons.

<b>Button</b>	<b>Description</b>
Hide/Show	Opens or closes the Navigation pane.
Back	Displays the previously viewed topic like the Back button in a browser.
Forward	Displays the previously viewed topic if the user clicked the Back button. This button functions like the Forward button in a browser.
Stop	Stops retrieving the file information and contents.
Refresh	Updates the topic that is currently displayed in the topic pane.
Home	Displays the default topic you defined for the help file.
Options	Opens a menu that provides many commands, such as Home, Show, Back, Stop, Refresh, Print, Search Highlight On/Off, and Internet Options. This menu also includes commands for all the buttons included on help windows.
Print	From the Contents tab, this button allows the user to print the selected topic, and optionally all subtopics. From the Index or Search tab, opens the Print dialog box to print the current topic.
Locate	Displays in the table of contents the location of the current topic is not listed in the table of contents, this command will not work.
Jump 1	Jumps to an author-designated Web page or help topic.
Jump 2	Jumps to an author-designated Web page or help topic.

# Navigation Pane in HTML Help

By default, the navigation pane provides the following tabs, which you can include or exclude from your help output through HTML Help Workshop:

## Contents tab

Displays the table of contents in the form of an expand/collapse tree view. The table of contents includes all paragraph styles that you assigned a TOC level in Style Designer. When the user selects an entry in the Contents tab, the information from that topic is displayed in the topic pane. You can customize the icons used for specific topics within the table of contents.

## Index tab

Displays an alphabetical list of keywords associated with topics in the source documents. Writers use their source document authoring tool to create standard index markers or field codes that define these keywords in their source documents.

## Search tab

Provides a powerful full-text search feature with several advanced search options. The user can type a search string and then select **Display** to view a list of topics that contain the word or phrase specified.

## Favorites tab

Displays a list of topics in the help that the user has added to his or her personal list of favorites. This tab allows users to bookmark help topics they often use or want to quickly find in the future. When the user selects a topic on this tab, the information from that topic is displayed in the topic pane.

# Topic Pane in HTML Help

The output pages generated from your source documents are displayed in the topic pane. When a user selects a topic on any of the tabs in the navigation page, the content of that topic is displayed in the topic pane.

# Topic Only View in HTML Help

The show/hide button in the toolbar pane allows users to add or remove the navigation pane. The writer can also define a window in HTML Help Workshop that does not include the navigation pane, or with the navigation pane hidden by default. Without the navigation pane, more screen area is available to display the help topic content, or for the application itself. However, the navigation pane helps

users view where they are in the organization of topics, and quickly browse and access other topics in the table of contents.

## HTML Help Workshop

HTML Help Workshop is a help authoring tool that allows you to create and manage Microsoft HTML Help projects and their related files. You can use this tool, which is installed with ePublisher, to further customize your help, such as changing the appearance of the Contents and Search tabs. You can also create an override for your project `.hhp` file in your ePublisher project, which allows you to implement the custom HTML Help Viewer window you need. For more information about Microsoft HTML Help and its related files and customization options, see the help for HTML Help Workshop.

## HTML Help Project File (.hhp)

The HTML Help project `.hhp` file identifies all the elements of an HTML Help project. This project file is separate from the ePublisher project. The HTML Help project file contains the information HTML Help Workshop needs to combine the source files, images, index, and table of contents into a single, compiled help `.chm` file.

The HTML Help project file also defines the appearance and behavior of the HTML Help Viewer window. ePublisher creates the HTML Help project file based on the `template.hhp` file and the settings in your ePublisher project. You can override the default `template.hhp` file to adjust the default appearance of your HTML Help, such as the default size and position of the HTML Help Viewer window or the buttons displayed in the toolbar pane. You can also define additional windows in the `template.hhp` file, such as a window without the navigation pane. For more information, see “Adjusting the HTML Help Viewer Window Size and Toolbar Buttons” and “Creating an Additional HTML Help Window Definition”.

## HTML Help Contents File (.hhc)

The HTML Help contents `.hhc` file defines the table of contents for the help. This file includes entries for each occurrence of the paragraph styles for which you assigned a TOC level in your ePublisher project. HTML Help displays the contents of this file on the Contents tab of the navigation pane. ePublisher generates the `toc.hhc` file for your project using the `template.hhc` file.

## HTML Help Index File (.hhk)

The HTML Help index `.hhk` file defines the index for the help. This file includes entries based on the index markers and field codes in your source documents. HTML Help displays the contents of this file on the Index tab of the navigation pane. ePublisher generates the `index.hhk` file for your project using the `template.hhk` file.

# HTML Help Mapping File (.h)

The HTML Help mapping `.h` file identifies the mapping information for context-sensitive help links. This file includes an entry for each TopicAlias marker or field code in your source documents. Application developers build this file with their project and use the defined aliases to display the appropriate topic in the help. This file is linked in the `MAP` section of the `.hhp` file, and its entries coordinate with the entries in the `ALIAS` section of the `.hhp` file.

## Delivering HTML Help

When you generate output for your project, ePublisher creates the `Output` folder in your project folder, and then creates a folder in the `Output` folder for each generated target. Then, ePublisher creates a folder named for the project itself in the target folder. For example, when you generate output, ePublisher creates the following folder structure:

```
<Project Area>\<Project Name>\Output\<Target Name>\<Project Name>
```

In this folder structure, `<Project Area>` is the name of the ePublisher component you are using, such as `ePublisher Express`, `<Project Name>` is the name of your ePublisher project, and `<Target Name>` is the name of your ePublisher target, such as `Microsoft HTML Help 1.x`.

By default, ePublisher compiles the `.chm` and stores it in the `<Target Name>` folder. To deliver your HTML Help, you need to deliver the `.chm` file.

If you created context-sensitive help, your application developer needs to build with the `<Target Name>\<Project Name>\<Project Name>.h` file. If you created What's This field-level help, your application developer also needs to build with the `<Target Name>\<Project Name>\whatisthis.h` file. For more information about these mapping files, see "Using Context-Sensitive Help in HTML Help" and "Defining What's This (Field-Level) Help in HTML Help".

# Oracle Help

Oracle Help Viewer  
Oracle Help Files  
Delivering Oracle Help

To deliver help on multiple platforms, you need a solution that runs on all those platforms. This solution helps you avoid complications of each platform and allows you to deliver one solution that meets the needs on all your platforms. This common solution can save development time and effort.

ePublisher supports several Java-based help output formats, such as Eclipse Help, Oracle Help, and Sun JavaHelp, that provide a common solution for multiple platforms without JavaScript support. WebWorks Help and WebWorks Reverb also provide a cross-platform solution, but it requires JavaScript support. If you are delivering a Java-based application, or if your environment does not support JavaScript, a Java-based help solution can help you deliver your help content.

The Oracle Help format is a Java-based help format that produces the complete set of files required to deliver online help based on the Oracle Help for Java technology. Oracle Help for Java is a set of Java components and an API for developing and displaying HTML-based help content in a Java environment. Oracle Help is recommended to produce help for an application written in the Java programming language.

**Note:** Oracle also offers a separate technology called Oracle Help for the Web. ePublisher does not support Oracle Help for the Web.

To view Oracle Help, users must have a Java Virtual Machine (JVM) installed. If you produce help for a Java application, the JVM will be installed on computers running the application. The Oracle Help components must also be installed on the computer.

The Oracle Help output format produces content that conforms to the HTML 4 and XHTML 1.0 standards and uses Cascading Style Sheets that conform to the CSS1 standard. This output format also produces files that are specifically required for Oracle Help. For more information about the Oracle Help technology, see the Oracle Web site at [www.oracle.com/technology/tech/java/help](http://www.oracle.com/technology/tech/java/help).

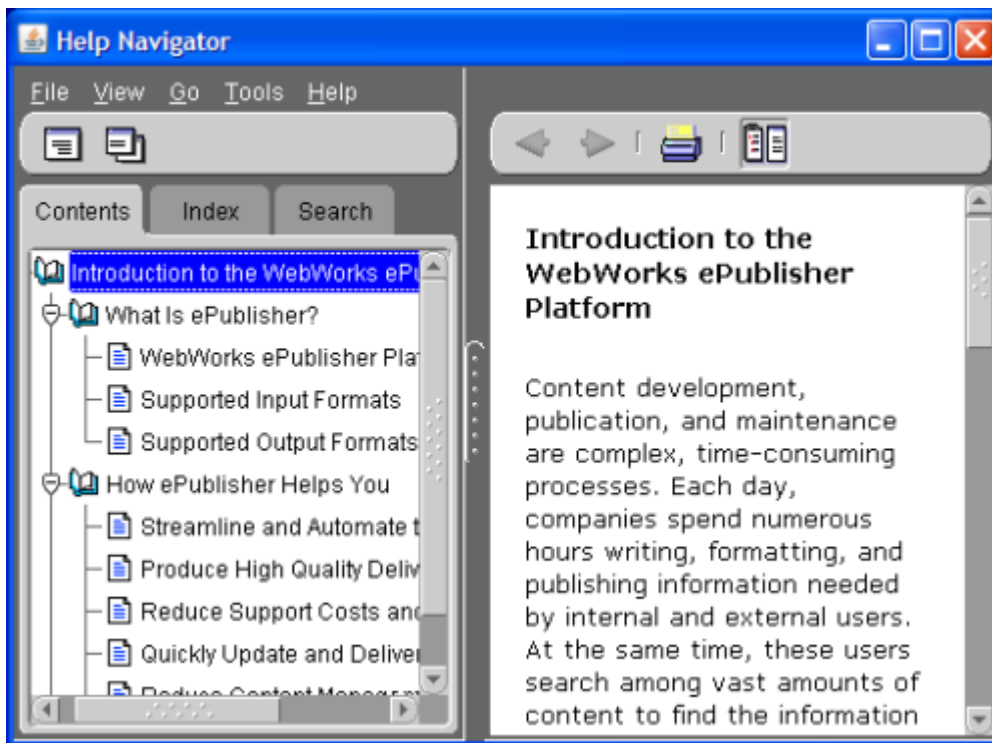
When deciding whether to use the Oracle Help output format, review the following considerations:

- Oracle Help is one of the output formats you can use to produce help for a Java application. You can also use the Oracle Help output format to produce content for the Web if all your users have, or are prepared to install, a Java Virtual Machine.

- If you need to produce help for a pure Java application, you may also consider using the Eclipse Help and Sun JavaHelp output formats. Each output format has slightly different requirements, appearance, and behavior.
- If you need to produce help for a Web-based application hosted on a Web server or on an intranet server that is not a pure Java application, you may consider using the WebWorks Help or WebWorks Reverb output format.

## Oracle Help Viewer

The Oracle Help output format produces Java-based online help. Oracle Help includes a navigation pane and a topic pane that is similar to other help viewers.



## Oracle Help Files

Oracle Help has several files that work together to define and create the Oracle Help solution. ePublisher incorporates these files into the compressed helpset file, so you do not need to distribute them to users.

### Helpset .hs File in Oracle Help

The helpset `.hs` file contains descriptive information about your Oracle Help helpset. ePublisher generates this file using the `template.hs` page template. For more information about the helpset file, see the Oracle Help documentation.



# Control .xml Files

The control `.xml` files define the table of contents, index, and topic aliases in your Oracle Help project. These files are named `projectTOC.xml`, `projectMap.xml`, `projectLinks.xml`, and `projectIndex.xml`, where *project* is the name of your ePublisher project. For more information about these files, see the Oracle Help documentation.

## Full Text Search .idx Index File

The internal full text search `.idx` index file provides support for the full text search feature.

## Manifest .mft File

ePublisher automatically generates the manifest `.mft` file. This internal file is used to create the final Java archive `.jar` file that you distribute to users.

## Delivering Oracle Help

You can provide a help system in Oracle Help format either as a collection of individual files or in a single, compressed Java archive `.jar` file. A `.jar` file, similar to a `.zip` file, compresses and stores a collection of files.

When you generate output for your project, ePublisher creates the `Output` folder in your project folder, and then creates a folder in the `Output` folder for each generated target. Then, ePublisher creates a folder named for the project itself in the target folder. For example, when you generate output, ePublisher creates the following folder structure:

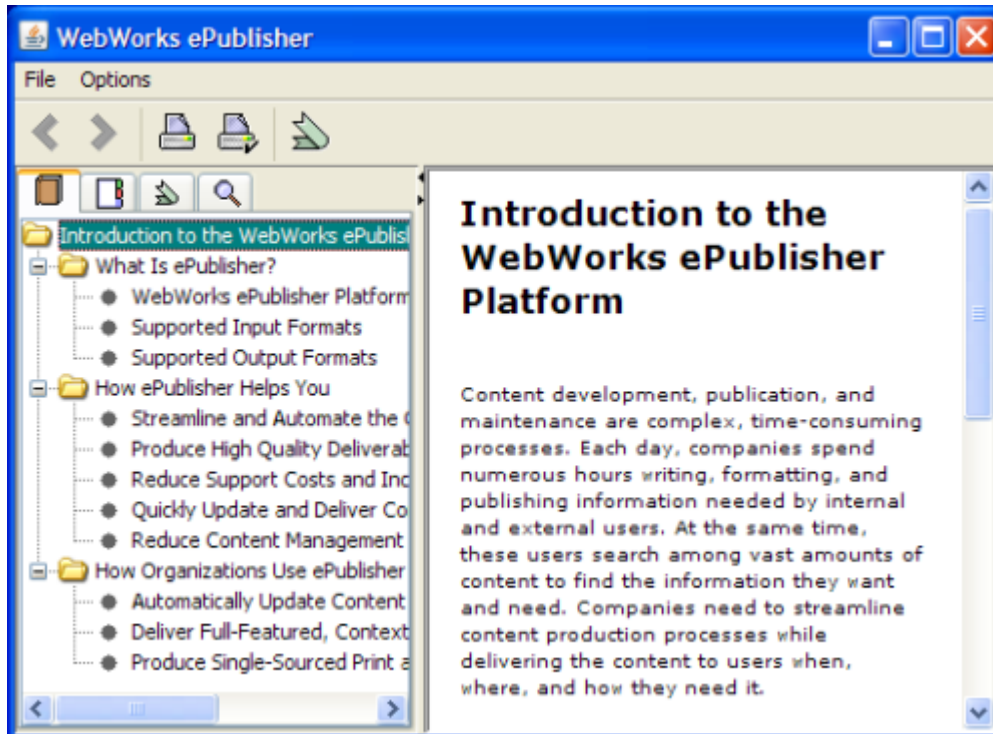
```
<Project Area>\<Project Name>\Output\<Target Name>\<Project Name>
```

In this folder structure, *<Project Area>* is the name of the ePublisher component you are using, such as `ePublisher Express`, *<Project Name>* is the name of your ePublisher project, and *<Target Name>* is the name of your ePublisher target, such as `Oracle Help`. To deliver your Oracle Help generated output, you can provide the complete contents of the *<Target Name>\<Project Name>* folder to your Java application developers, and they can determine whether to deliver the `.jar` file or the individual files to application users.

# Sun JavaHelp

Sun JavaHelp Files  
Delivering Sun JavaHelp

Similar to other help systems, Sun JavaHelp provides a navigation pane and a topic pane. The navigation pane provides Contents, Index, Favorites, and Search tabs. The topic pane displays the content of the help topic selected on a tab in the navigation pane.



## Sun JavaHelp Files

Sun JavaHelp delivers content in HTML files. The table of contents, index, and helpset `.hs` files use the Extensible Markup Language (XML) format.

## Helpset .hs File in Sun JavaHelp

The helpset `.hs` file contains configuration information in XML format that Sun JavaHelp needs to display your help system. The following figure shows a sample helpset file:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE helpset PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp
HelpSet Version 1.0//EN" "http://java.sun.com/products/javahelp/
helpset_1_0.dtd">
```

```

<helpset version="1.0">
  <title>Interesting help project</title>
  <maps>
    <homeID>home</homeID>
    <mapref location="test.jhm" />
  </maps>
  <view>
    <name>TOC</name>
    <label>Interesting help project</label>
    <type>javax.help.TOCView</type>
    <data>testt.xml</data>
  </view>
  <view>
    <name>Index</name>
    <label>Index</label>
    <type>javax.help.IndexView</type>
    <data>testi.xml</data>
  </view>
  <view>
    <name>Search</name>
    <label>Search</label>
    <type>javax.help.SearchView</type>
    <data engine="com.sun.java.help.search.DefaultSearchEngine">
      JavaHelpSearch
    </data>
  </view>
</helpset>

```

The helpset file begins with an XML declaration. The next part of the file, as shown in the following sample, defines the help version, the title of the help, and the location of the mapping file that is used to support context-sensitive help:

```

<helpset version="1.0">
  <title>Interesting help project</title>
  <maps>
    <homeID>home</homeID>
    <mapref location="test.jhm" />
  </maps>

```

The remaining sections of the file define several views, such as the table of contents (TOC), index, and search. Each `<view>` tag includes several tags that define the view, such as the `<name>` tag that specifies the name of the view and the `<data>` tag that specifies the data file for the view.

You can also define your own views in Sun JavaHelp by overriding the `template.hs` file. For example, you can create an alternate list of topics and provide that list as a separate view.

## Contents toc.xml File in Sun JavaHelp

The contents `toc.xml` file defines the items in the table of contents as shown in the following example file:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE toc PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp TOC
  Version 1.0//EN" "http://java.sun.com/products/javahelp/toc_1_0.dtd">
<toc version="1.0">
  <tocitem target="tutorial_htm_1003580" text="Hybrid Web/CD:">
  <tocitem target="tutorial_htm_1004263" text="Roadmap"/>
  <tocitem target="tutorial_htm_1005107" text="Explosion"/>
  <tocitem target="tutorial_htm_999374" text="Bandwidth"/>
  ...
</tocitem>
</toc>
```

## Index ix.xml File in Sun JavaHelp

The index `ix.xml` file lists your index entries as shown in the following example file:

```
<?xml version='1.0' encoding='ISO-8859-1' ?>
<!DOCTYPE index PUBLIC "-//Sun Microsystems Inc.//DTD JavaHelp
  Index Version 1.0//EN" "http://java.sun.com/products/javahelp/
index_1_0.dtd">
<index version="1.0">
  <indexitem target="tutor4_htm_999680" text="1996 Telecom Act"/>
  <indexitem target="tutor11_htm_999661" text="active catalog"/>
  <indexitem target="tutor10_htm_999632" text="ATM"/>
  <indexitem target="tutor10_htm_999634" text="audio synchronization"/>
  >
  <indexitem text="bandwidth">
  ...
</index>
```

## Map .jhm File in Sun JavaHelp

Sun JavaHelp supports context-sensitive help through a mapping `.jhm` file. The mapping file contains information that links your Sun JavaHelp topics to particular locations in the Java application. For more information, see “Using Context-Sensitive Help in Oracle Help and Sun JavaHelp”.

## Delivering Sun JavaHelp

You can provide a help system in Sun JavaHelp format either as a collection of individual files or in a single, compressed Java archive `.jar` file. A `.jar` file, similar to a `.zip` file, compresses and stores a collection of files.

When you generate output for your project, ePublisher creates the `Output` folder in your project folder, and then creates a folder in the `Output` folder for each generated target. Then, ePublisher creates a folder named for the project itself in the target folder. For example, when you generate output, ePublisher creates the following folder structure:

```
<Project Area>\<Project Name>\Output\<Target Name>\<Project Name>
```

In this folder structure, *<Project Area>* is the name of the ePublisher component you are using, such as `ePublisher Express`, *<Project Name>* is the name of your ePublisher project, and *<Target Name>* is the name of your ePublisher target, such as `JavaHelp`. To deliver your Sun JavaHelp generated output, you can provide the complete contents of the *<Target Name>\<Project Name>* folder to your Java application developers, and they can determine whether to deliver the `.jar` file or the individual files to application users.

# WebWorks Help

[The Frameset View in WebWorks Help](#)  
[Topic Only View in WebWorks Help](#)  
[WebWorks Help Output Files](#)  
[Delivering WebWorks Help](#)  
[Searching WebWorks Help](#)

Delivering help content on multiple platforms and browsers can be a challenge. You need a solution that provides advanced help features, such as full-text search, expand/collapse sections, browse and breadcrumb navigation, related topics, and a table of contents tree control. WebWorks Help delivers these features and others on multiple platforms and browsers. Whether your help needs to be viewed on the Macintosh or Windows operating system, displayed with Firefox, Safari, or Internet Explorer, WebWorks Help provides the flexible, adaptable help solution you need.

**Note:** Users considering a web help output should first consider a Reverb 2.0 output before choosing Reverb or WebWorks Help. Reverb 2.0 is designed to work across modern browsers, and includes our latest features. WebWorks Help 5 is recommended for web server use only.

The WebWorks Help format combines standard XHTML, CSS1, and CSS2 with JavaScript to provide many online help navigation controls and features, including an expandable/collapsible table of contents, an index, full text search, and Related Topics menus. You can easily integrate WebWorks Help with applications written in C, C++, Java, or Visual Basic, as well as Web-based applications. This comprehensive help format allows you to quickly create and deploy context-sensitive help and comprehensive help that you can fully customize to match your corporate branding.

WebWorks Help enables you to publish your content in a variety of ways:

- On a Web server
- On a network server available to multiple users through a share
- On the local computer where the application is installed
- On a CD-ROM

## The Frameset View in WebWorks Help

When you open WebWorks Help, you can choose whether to display the navigation pane and the content, or only the content in the browser. You can control the way WebWorks Help opens based on whether the user selected a context-sensitive help link or opened the comprehensive help. This flexibility allows you to deliver your content the way the user needs it.

The frameset view divides the browser window into several areas:

### Navigation pane

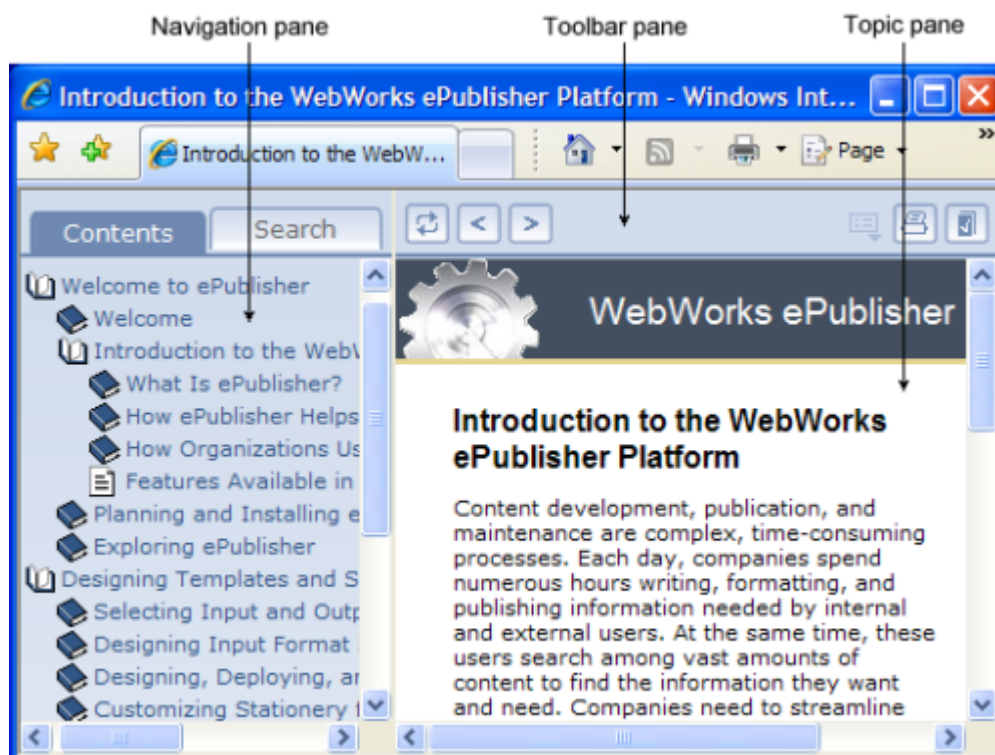
Provides multiple tabs for navigation elements, such as the table of contents and full-text search.

### Toolbar pane

Provides several standard toolbar buttons to perform common tasks, such as show or hide the navigation pane, browse to the next topic, and print the content.

### Topic pane

Displays the content of the topic selected in the navigation pane.



## Navigation Pane in WebWorks Help

By default, the navigation pane displays several tabs that provide core navigational features for your help:

### Contents tab

Displays the table of contents, including entries for each style that you assigned a TOC level value in Style Designer. When a user clicks on item in the Contents tab, the selected topic is displayed in the topic pane. This tab displays the table of contents as an expand/collapse tree view. Each book icon represents a table of content entry that has subentries.

### **Index tab**

Displays an alphabetical list of keywords associated with topics. To view index entries, select a letter to display the entries that start with that letter. When the user clicks on an index entry, the related topic is displayed in the topic pane. The writer defines the keywords as index entries in the source documents.

### **Search tab**

Provides a full-text search. When the user clicks **Go**, WebWorks Help lists the titles of the topics whose content contains the words specified in the **Search** field. This tab provides **Rank** and **Title** columns. Each listed topic has a relevancy ranking number, which reflects how well the topic matches the search criteria. The ranking is specified in the `wwh_files.xml` file. For more information about how to modify the ranking, see “Modifying the Search Ranking”.

### **Favorites tab**

Lists the topics that the current user added to his or her list of personal favorites. In WebWorks Help, users can add frequently accessed or important help topics to their personal list of favorites. When the user clicks on a topic on the Favorites tab, the help topic is displayed in the topic pane. The Favorites tab also provides a Remove button that allows users to delete any unwanted topics from their list.

## **Toolbar Pane in WebWorks Help**

By default, the toolbar pane displays a set of buttons that provide users with additional features. These buttons allow users to navigate through the help and access common functions, such as printing the displayed topic and emailing a link to a topic. The standard WebWorks Help toolbar includes the following buttons:

### **Show in Contents button**

Allows users to locate the topic they are viewing in the table of contents. When a user clicks this button, WebWorks Help highlights the entry in the Contents tab that corresponds to the currently displayed topic.

### **Show Navigation button**



Allows users to show the hidden navigation pane. When a context-sensitive help link displays a topic with the navigation pane hidden, this button is displayed in the toolbar. When a user clicks this button, WebWorks Help displays the navigation pane and highlights the entry in the Contents tab that corresponds to the currently displayed topic.

### **Previous button**

Allows users to navigate back to topics that precede the currently displayed topic in the help.

### **Next button**

Allows users to navigate forward to topics that follow the currently displayed topic in the help.

### **PDF button**

Displays a PDF file of the source document from which the currently displayed topic was generated.

### **Related Topics button**

Displays a list of topics that share a common or related theme with the currently displayed topic. The writer must define related topics in the source documents and the Stationery designer must enable related topics support for the help to display them. For more information, see "Defining Related Topics".

### **Print button**

Prints the currently displayed topic.

### **Email button**

Allows you to collect feedback from your users. This button opens a blank email message addressed to the email address you specify. The subject line of the email message identifies the displayed topic when the user clicked the Email button.

## **Topic Pane in WebWorks Help**

This pane displays the content of the help topics generated from your source documents. When a user selects a topic on the Contents, Search, Index, or Favorites tab, the topic pane displays the content of that topic.

## **Topic Only View in WebWorks Help**

You can display a topic-only view, which hides the navigation pane and displays only the toolbar pane and the topic pane. With this topic-only view, an application can open a smaller browser window that leaves more area on the screen for the application itself. Context-sensitive help often uses the topic-only view.

## WebWorks Help Output Files

When you generate the WebWorks Help output format, ePublisher creates all the files required to deliver and display WebWorks Help. The `index.html` file is the default entry file that defines the frameset used by the help. To open the help, users open this entry file. ePublisher creates the `.html`, `.css`, `.js`, and various image files included in the WebWorks Help output.

## Delivering WebWorks Help

When you generate output for your project, ePublisher creates the `Output` folder in your project folder, and then creates a folder in the `Output` folder for each generated target. Then, ePublisher creates a folder named for the project itself in the target folder. For example, when you generate output, ePublisher creates the following folder structure:

```
<Project Area>\<Project Name>\Output\<Target Name>\<Project Name>
```

In this folder structure, `<Project Area>` is the name of the ePublisher component you are using, such as `ePublisher Express`, `<Project Name>` is the name of your ePublisher project, and `<Target Name>` is the name of your ePublisher target, such as `WebWorks Help`. To deliver your WebWorks Help generated output, you need to deliver all the files and subfolders in the `<Target Name>\<Project Name>` folder.

The `<Target Name>\<Project Name>` folder contains the entry-point file, `index.html` by default, which establishes the help set appearance. When the user opens the entry-point file, the browser uses all the files in the `<Target Name>\<Project Name>` folder to display the help, including all the topic files, generated `.css` files, `.pdf` files, images, and WebWorks Help components.

## Searching WebWorks Help

Use the following guide to assist users in finding terms in a WebWorks Help 5.0 help system:

### Boolean

All search words and phrases have an implicit AND

### Word Search

For example, searching: `eggs bacon` returns all documents containing “eggs” and “bacon”

### **Phrase Search**

For example, searching `"Good Morning"` returns all documents containing “good” and “morning” where the two words are adjacent

### **Wildcard**

For example, searching `e*` returns all documents containing words that start with e.

To further customize the way the search results are displayed, See [Modifying the Search Ranking](#)

# Designing, Deploying, and Managing Stationery

Understanding Stationery  
Designing Stationery  
Deploying Stationery  
Managing and Updating Stationery

This section outlines the Stationery design, deployment, and management process. ePublisher Express projects use Stationery designed in ePublisher Designer to define the appearance and behavior of generated output. You can use Stationery to define a standard for one or more projects to use. You can also customize the design to meet your specific needs.

Writers use the Stationery when they create projects. When the Stationery designer changes or updates the Stationery a project uses, ePublisher prompts writers to synchronize their projects with the updated Stationery associated with their projects. This powerful feature allows you to quickly deploy and manage Stationery updates across your organization.

## Understanding Stationery

ePublisher projects use Stationery designed in ePublisher Designer by a Stationery designer to define the appearance and behavior of generated output. **Stationery** is a complete set of processing rules and styles that define all aspects of the output. Writers use the Stationery created by the Stationery designer when they create projects and generate output. Once the Stationery designer changes or updates the Stationery, ePublisher prompts writers when they open projects associated with that Stationery to synchronize their projects with the updated Stationery.

**Stationery design projects** are not based on Stationery. These projects are created in ePublisher Designer and are used to create and maintain Stationery. To modify or update Stationery, you need to update the Stationery design project and then save it as Stationery.

Stationery stores all the style and behavior settings. Stationery also captures the transformation process and isolates it from changes in future ePublisher releases. Since you can always use ePublisher Designer to create new Stationery, you can easily maintain the existing Stationery and move forward with a new ePublisher release. You can also limit the number of formats included in a Stationery to reduce complexity and potential confusion in your working environment. Each output format, such as HTML Help, WebWorks Help, and Dynamic HTML, requires certain files to generate output. Without these files, the formats do not have the components required to generate the correct output.

To create a new project, ePublisher Express users must have Stationery. For more information about Stationery and how it works, see “Stationery, Projects, and Overrides”.

## Stationery Components

Stationery includes the following components:

### Stationery .wxsp file

Contains the style definitions and target settings for the selected output formats and targets. This file stores the style definitions including paragraph, character, table, page, graphic, and marker styles. This file also stores the conditions, variable values, cross-reference definitions, and target settings for each target.

### Manifest file

Identifies the files in the `Files`, `Formats`, and `Targets` folders associated with the Stationery. Any time a Stationery designer adds, removes, or modifies a file in one of these folders and saves the Stationery, ePublisher updates the Stationery manifest file.

### Files folder

Contains all the files and folders you want to include in the project. ePublisher copies these files and folders to the output location and includes them in the output.

### Formats folder

Contains a complete copy of all the run-time files required to transform your source documents for specific output formats. In your Stationery design project, this folder contains the customized files for specific output formats needed to transform your source documents and create your output files. This folder includes a subfolder for each defined output format with customized files. These files override the default files used by ePublisher to generate the associated output format. New ePublisher releases may include changes to the default transformation files. By storing the customized files in a separate location, ePublisher can override the default files by using these customized files and it can also protect these customized files when you install a new ePublisher release.

### Targets folder

Contains the customized files for specific targets needed to transform your source documents and create your output files. This folder includes a subfolder for each defined target with customized files. These customized files

override the files stored in the Formats folder for the output format related to the target. For example, you can create a customized `Page.asp` page layout file for a specific target.

## Understanding Stationery Synchronization

ePublisher projects use Stationery designed in ePublisher Designer to define the appearance and behavior of generated output. **Stationery** is a complete set of processing rules and styles that define all aspects of the output. Writers use the Stationery created by the Stationery designer when they create projects and generate output.

Once the Stationery designer changes or updates the Stationery, ePublisher prompts writers when they open projects associated with that Stationery to synchronize their projects with the updated Stationery. ePublisher Express prompts you to synchronize your project with its Stationery under the following conditions:

- The project manifest file differs from the Stationery manifest file.
- The Stationery settings have been modified.

When writers synchronize their projects with the updated Stationery, all settings in the project are updated to match the Stationery. For more information, see “Synchronizing Projects with Stationery”.

## Designing Stationery

ePublisher projects use Stationery designed in ePublisher Designer to define the appearance and behavior of generated output. You can use Stationery to define a standard for one or more projects to use. You can also customize the design to meet your specific needs. Writers then use the Stationery when they create projects.

There are many considerations when designing and developing Stationery. A well-designed Stationery is less costly to maintain and manage into the future.

## Creating a Stationery Design Project

Stationery design projects are not based on Stationery. You create a Stationery design project in ePublisher Designer and use it to create and maintain Stationery. To create Stationery, you need to save a project as Stationery. To modify or update Stationery, you need to update the Stationery design project and then save it as Stationery.

The ePublisher workflow improves productivity and saves time for writers who need to manage online content. You create Stationery that defines a set of reusable settings to apply to any project. Without any additional work, writers can load these

settings into their projects, and update them automatically when a change to the Stationery is made.

**Note:** To create a new Stationery design project from an existing Stationery design project, you can manually copy the existing Stationery design project `.wep` file and associated files and folders.

## To create a Stationery design project

1. Open ePublisher Designer.
2. On the **File** menu, click **New Project**.
3. In the **Project name** field, type the name for your Stationery project.
4. In the **Format** field, select the default output format you want to include in your Stationery, and then click **Next**. You can add other output formats at a later time.
5. Click **Add** and select your standard sample source files to add them to the **Source Documents** list box. These files provide all the standards for each input format.
6. When all your sample source files are listed in the **Source Documents** list box, click **Finish**.
7. On the **Project** menu, click **Scan All Documents** to add all elements defined in your sample source documents to the project. This process ensures you can define how ePublisher processes each of these elements, such as styles and variables.
8. On the **File** menu, click **Save** to save your Stationery design project.

Next, you can add all the other output formats you want to include in your Stationery.

## Adding Output Formats to Your Stationery Design Project

You can include one or more output formats in your Stationery. To simplify your Stationery, define only the output formats you need. Writers can then use these output formats defined in the Stationery to create one or more targets in their projects.

Each output format creates a target in your Stationery design project. You can also define multiple targets with the same output format. For example, you can define multiple targets, one for each standard OEM partner you may have. Each of these targets may generate the same output format with settings, such as company name and address, customized for each partner.

The Stationery Designer properties and options are shared across all targets and output formats. Some settings, such as the target settings, variable values, conditions, and cross-reference formats, are defined per target. Some targets and output formats also offer additional features and customizations.



## **Adding a Target to Your Stationery Design Project**

When you create your Stationery design project, you select the default output format. ePublisher adds the target for that output format to your project. You can then add more targets and output formats to your project.

### To add a target to your Stationery design project

1. Open your Stationery design project.
2. On the **Project** menu, click **Manage Targets**.
3. Click **Add**.
4. In the **Format Type** field, select an output format for the target.
5. In the **Target Name** field, type a name for the target, and then click **OK**.

## Selecting an Active Target in Your Stationery Design Project

The **active target** is the target currently selected for your project. ePublisher uses the active target to identify what output to generate. If you modify settings, such as variable values, that are applied to an individual target and are not shared across targets in a project, ePublisher saves those changes for the active target. ePublisher applies the settings defined through the options on the **Target** menu to the active target.

### To select the active target

1. Open your Stationery design project.
2. On the **Project** menu, click the target you want in the **Active Target** menu option.

## Updating a Project to Include All Styles

ePublisher transforms your source documents into content that you can deliver to virtually any online format or platform. To make sure your content is properly displayed and easy to use for your audience, you may want to modify the appearance of the online content, as well as add features your audience wants and needs. However, you do not want to modify the original source documents multiple times to produce the different types of output you need, such as print, online help, and Web site content.

ePublisher enables you to maintain your source documents as you need to for print delivery. You can then use Style Designer to define how you want your content transformed for your other output formats. ePublisher helps you create the appearance you want and the functionality you need for your online output.

Your project defines how to transform your content based on the paragraph, character, table, graphic, page, and marker styles in your source documents. If you add new elements to your source documents, you need to scan your documents to include these new elements, such as new paragraph styles, in your project.

## To scan all your source documents

1. Open your Stationery design project.
2. On the **Project** menu, click **Scan All Documents**.

## Understanding Style Designer

Style Designer allows you to modify how various styles appear in your generated output. You can define how paragraphs, characters, tables, and images are displayed. You can also modify other aspects of your content, such as page layout and how page breaks are established. Style Designer defines the appearance of your online content, such as the color or font of a paragraph style, the style of a table border, the layout of a page, and the file format of your images in your generated output.

ePublisher uses the styles in your source documents to define how to transform and present your content online. ePublisher intelligently senses the styles in the source documents and presents a list of these styles in Style Designer. You can then define your generated output by specifying properties and options for each style. By using styles in your source documents, you have great control over your output. In Style Designer, the **Properties** tab controls the appearance of the selected style and the **Options** tab controls the behavior of the selected style in your generated output.

For example, if you are using a style called `Heading 1` in your source document, ePublisher lists this style in your project with the other styles used in your source documents. If you select `Heading 1` from the list of styles in Style Designer, you can then define all content that has that style to appear a specific way in your generated output. These modifications do not affect your source documents. ePublisher is essentially a filter that transforms your source documents into the output format you need.

You can also use a custom CSS file to modify the appearance of your content instead of using Style Designer for HTML-based output formats. For more information, see “Using a Custom CSS to Modify the Appearance of Content”.

## Modifying Output with CSS Properties and Attributes

Cascading style sheet (CSS) properties give you detailed control over the appearance of your generated output. Style Designer provides an intuitive interface to help you modify your output using CSS properties. In addition to the online help, additional knowledge of CSS properties and attributes can help you achieve the results you want. For more information about CSS principals, see W3 Schools at [www.w3schools.com](http://www.w3schools.com) and the World Wide Web Consortium (W3C) at [www.w3.org](http://www.w3.org).

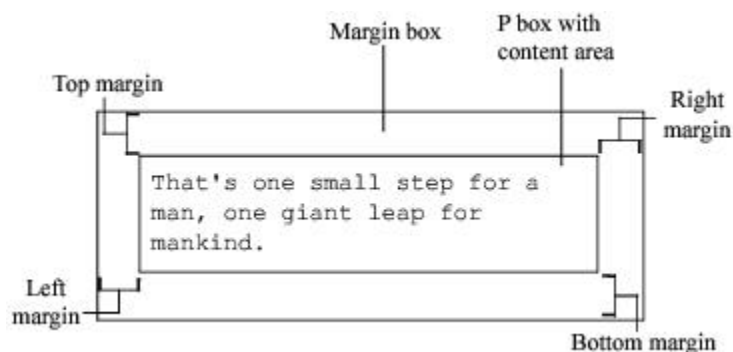
Because your online output is HTML- and CSS-based, the appearance of your output can be inconsistent between different browsers. This inconsistency is a limitation of HTML and CSS, since not all browsers implement the standards defined by the W3C in a consistent way. In addition, some output formats do not support all HTML and CSS features. In these cases, ePublisher disables or hides the incompatible options.

## Understanding the CSS Box Model

In accordance with the CSS2 specification, each element on a Web page generates its own invisible, rectangular box, sometimes referred to as the box model. This box delimits the space that the element occupies on the page and consists of a content area surrounded by optional blocks related to the border, margin, or padding around the content. For example, consider the following HTML snippet of a paragraph on a Web page:

```
<p>That's one small step for a man, one giant leap for mankind.</p>
```

Since this content is a paragraph, the text uses a `<p>` tag and is contained in a `P` box, or paragraph box. Assuming this paragraph has only a margin applied to it (and no border or padding), its box can be represented as shown in the following figure:



Other paragraph blocks, heading blocks, table blocks, and so forth may precede or follow this paragraph, and all blocks fit together inside a page block, which contains all other elements of the page. Boxes usually correspond to HTML `<h>`, `<p>`, or `<div>` tags and often contain inline elements within them. Inline formatting, such as a bold word within a sentence, creates inline invisible boxes within the paragraph text.

For Western languages, the boxes, and therefore the content within them, are arranged in a left-to-right, top-to-bottom flow, which is called normal flow. You can reverse normal flow for Eastern languages that flow from right to left. The normal flow includes all styles unless you move them outside of the normal flow using the **Float** or **Positioning** properties. With the **Float** or **Positioning** properties, you can create multi-column pages and other layouts that, in CSS1, required you to use tables. For more information about the CSS2 visual formatting model and normal flow, see the W3C documentation at [www.w3.org/TR/REC-CSS2/visuren.html](http://www.w3.org/TR/REC-CSS2/visuren.html).

# Inheriting Style Properties and Options

Many elements of your online design may have similar properties or options. For example, paragraphs, bulleted list items, and numbered list items may all use the same font and vertical spacing. You may also identify elements that should use settings from your source documents. For example, you may want tables to use the size defined in your source files. Style Designer allows you to specify a precise value for a property, or you can specify from where a property inherits its value. When using inheritance for properties, you can specify the source of the inheritance using the following values:

## Explicit

Ignores all inheritance values and uses the value you specify for this property.

## Do not emit

Excludes the property from the generated output for the selected style. This option can help you troubleshoot a design issue and determine which property or element is associated with the issue.

## Inherit from style

Inherits the value for the property from the parent style. You can organize styles in a hierarchy and then use inherited properties to reduce maintenance costs for future changes. For example, if you have a Heading 1 and a Heading 2 style, you could make Heading 2 a child of Heading 1. If you select **Inherit <style name or target> property** for a property of Heading 2, it inherits the value for that property from Heading 1. For more information, see “Organizing and Managing Styles”.

## Document paragraph style

Inherits the value for the property from the style definition in your source document. By choosing this option for a property, ePublisher uses the formatting from your source document for that property.

## Document style catalog

Inherits the value for the property from the style definition in the source documents. When you select this option for a property, ePublisher uses the source document definition and ignores all manual changes made to the style in the source document.

# Understanding Options in Style Designer

The **Options** tab allows you to define the behavior of the selected item in Style Designer. For example, you can set the options for a paragraph style to create an expand/collapse section or to split the content and start a new output file. The

available options depend on the selected item, such as a paragraph style or a marker style, and the active target, such as WebWorks Help or Eclipse Help.

**Note:** The options you specify for a parent style can be inherited by child styles if the child style does not explicitly set the option value.

## Organizing and Managing Styles

In Style Designer, you can organize your styles in a hierarchy and then use inherited properties to reduce design and maintenance time. You can create a hierarchy of similar styles, set the style property values once, and have those values inherited by child styles.

The **Prototype** style, which is essentially the parent style for all styles, allows you to quickly define properties for all styles. This style allows you to make global changes across all styles that inherit properties from the **Prototype** style. For example, you may have all styles inherit their font and vertical spacing from the **Prototype** style.

By default, all style properties and options of a parent style are inherited by its child styles. If you change the property or option values of a parent style, those changes are inherited by the child styles of that parent style. You can override specific properties or options for a child style to make those not inherited from its parent style. Once the value of a specific property or option is set at the child level, changes you make to the parent style for that property do not affect the child style.

## To organize styles in Style Designer

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. Click the type of styles you want to organize, such as **Paragraph Styles**.
4. ***If you want to make a style the child of another style***, drag and drop the child style on top of the style you want to make its parent style.
5. Click on the child style and review the properties for that style to make sure it inherits the property values you want from its parent style.

## Previewing the Output from a Source File

Once you specify or change settings in your Stationery design project, you should generate your output to review and verify your changes. You can also generate output for an individual source document. As you develop your Stationery, you will often modify a few settings, generate your output and review the results, and then continue the process until you have the final results you want.

To quickly review your changes, you can also display a preview of a source document. A **preview** allows you to quickly see the affects of the changes you made, without requiring you to generate output for the project. The preview provides a close approximation of how your generated output will look, but your generated output may not exactly match the preview.

**Note:** At intervals while you develop your Stationery, you may want to create a backup copy of your Stationery design project to serve as a snapshot. This backup gives you a version to return to and use if you make some changes that you no longer want. For more information, see “Saving a Snapshot (Backup Copy) of Your Project”.



## To display a preview of a source document

1. Open your Stationery design project.
2. In **Document Manager**, select the source document for which you want to generate a preview.
3. On the **Project** menu, click **Display Preview**.

ePublisher displays a preview of the source document on a preview tab labeled with the name of the source document you selected. The preview provides you with an idea of how modifications you made to project settings affect the appearance of your output. However, some output features are not displayed or active within the preview, such as popup windows, links, and conditions.

## Defining New Pages (Page Breaks)

By default, ePublisher transforms each source document into one output page. You can associate a page break with specific paragraph styles to split your content into multiple pages, where each page creates a new output file. By dividing your content, you can present information to your audience in smaller chunks, organize and focus your content, reduce redundant information through links, and reduce scrolling by your audience. To avoid empty topics when multiple heading styles occur in a row, you can also define page break handling based on whether the previous style created a new page. This flexibility enables you to deliver your content your way.

## To create new pages based on styles

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the style for which you want to create a page break.
5. On the **Options** tab, assign a value to the **Page break priority** option. For more information about this option, click **Help**.
6. Review the target settings to ensure the **Page break handling** setting will correctly process your priority level values by completing the following steps:
  - a. On the **Project** menu, select the **Active Target** you want to specify settings for.
  - b. On the **Target** menu, click **Target Settings**.
  - c. Set the **Page break handling** setting to the appropriate value. For more information about this setting, click **Help**.

## Defining TOCs and Mini-TOCs

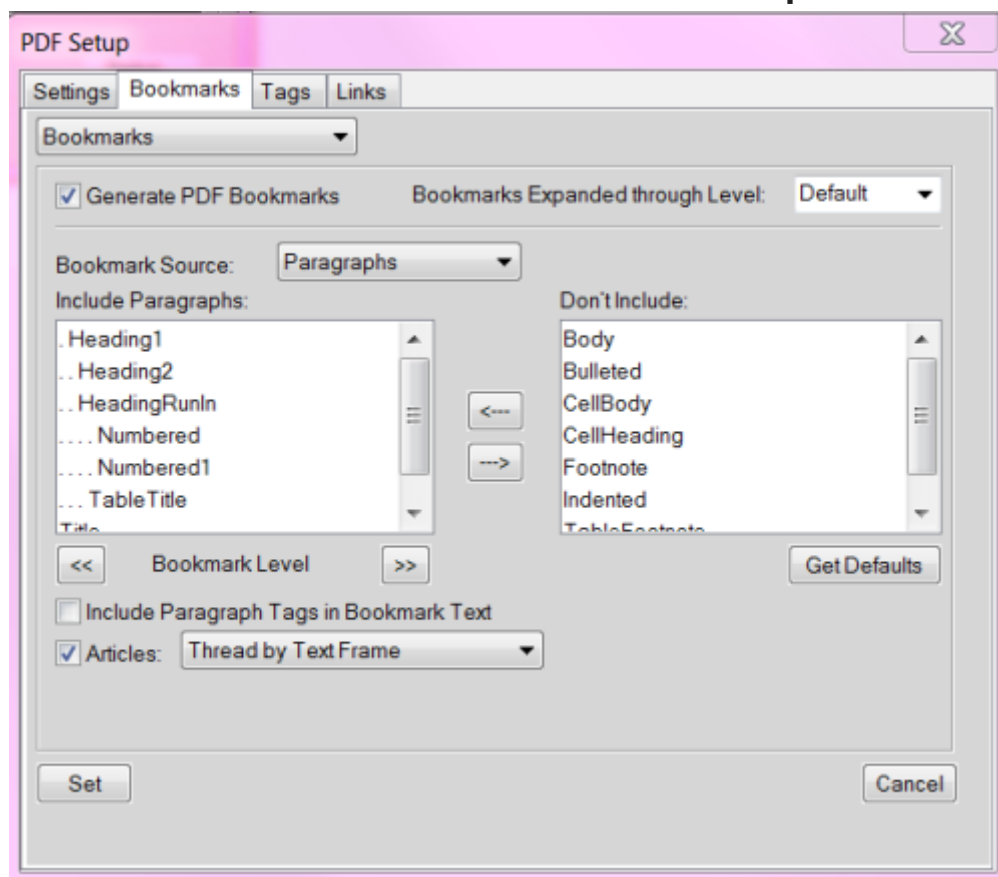
ePublisher does not use the table of contents (TOC) in your source document when it generates the online table of contents. This process allows ePublisher to correctly apply conditions and variables to the TOC, and to format the TOC as needed for the output format. ePublisher bases the TOC on paragraphs from your source document. Depending on the output format you select, you can specify whether to generate the TOC and what file name to assign to the TOC. For more information, see “Generating and Naming the Table of Contents File”. You can also modify the appearance of the table of contents for some output formats. For more information, see “Customizing the Navigation Pane in WebWorks Help” and “Modifying the Appearance of the Table of Contents in Dynamic HTML”.

## Defining the Table of Contents Structure (Levels)

By default, ePublisher attempts to automatically determine your source content’s TOC levels. You can manually adjust those levels if the settings do not meet your requirements.

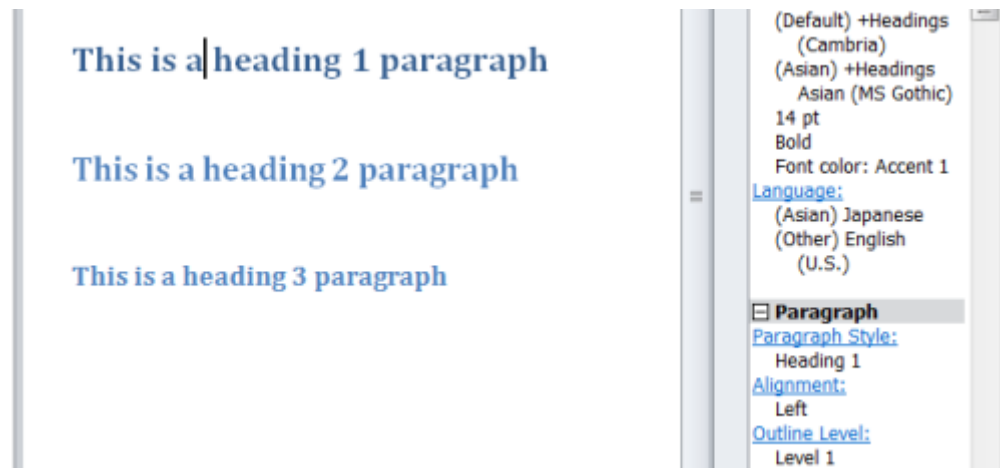
## To define the table of contents structure (levels)

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the style you want to include in your TOC.
5. On the **Options** tab, specify a value for the **Table of contents level** option. The default value, **Auto-Detect**, attempts to infer the necessary information based on your source content. Specify a value that indicates the level of the table of contents entry for the selected paragraph style. When working with Adobe FrameMaker, you can determine the levels by examining the **PDF Bookmarks** levels for paragraphs that appear in the table of contents. Here is an example of the **Bookmarks** tab from FrameMaker 11, to navigate here, you would select **File > Print....** Then click the **PDF Setup...** button.

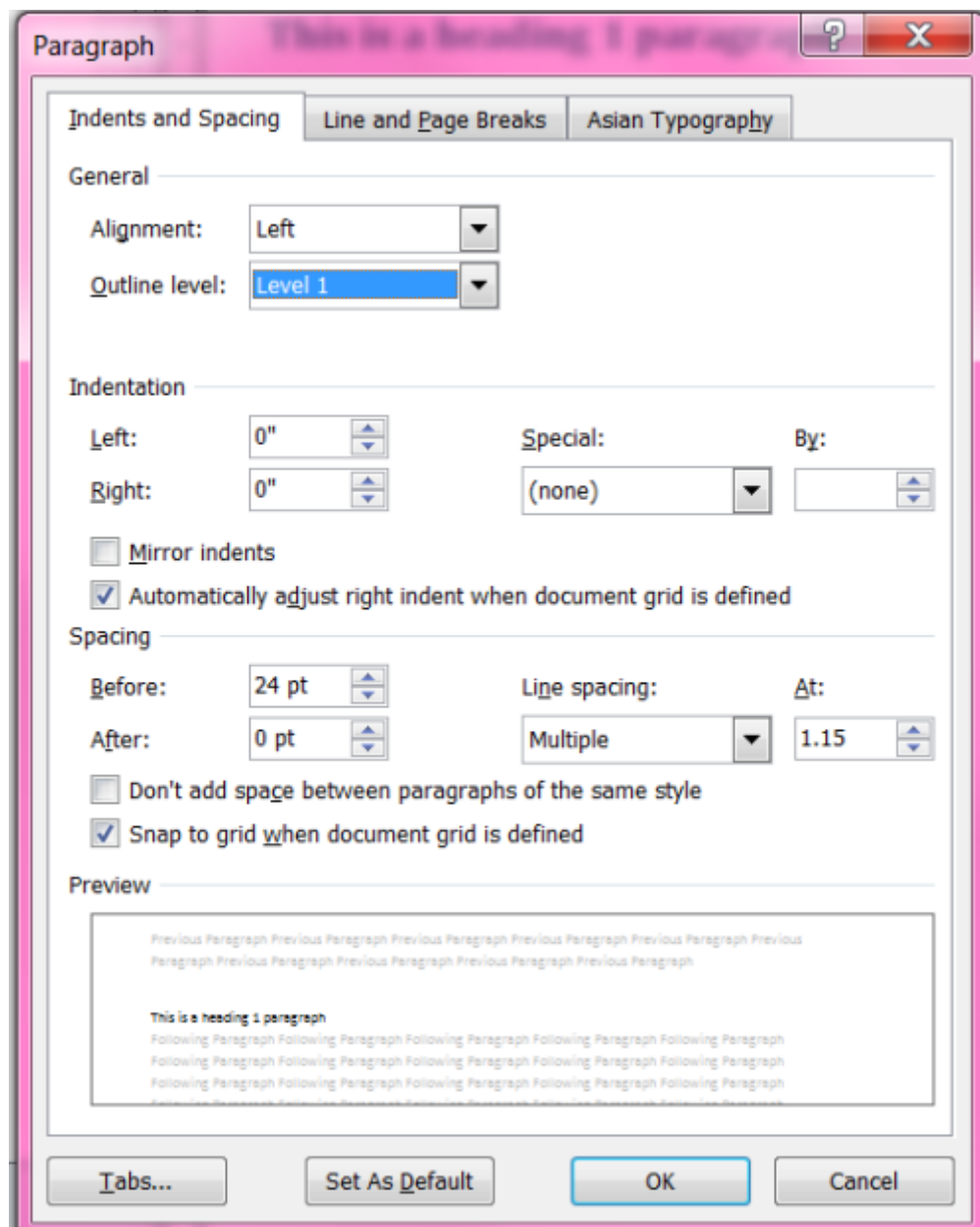


When working with Microsoft Word sources, the auto-detect values are based on the **Outline level** of each paragraph style. Below is a screenshot of the Word 2010

interface for each paragraph's outline level, to navigate here, you use the key combination **Shift + F1**.



Once you click the **Outline Level** hyperlink, you can select the desired level from the following window.



DITA XML auto-toc is taken from the map hierarchy. A good example would be from the sample input provided in the *Exp\_Design* project located in the ePublisher directory in the Documents library, or by the `[program files]\WebWorks\epublisher\[version]\Helpers\dita-ot\samples`

## Generating and Naming the Table of Contents File

By default, once you have defined your table of contents structure using Style Designer, ePublisher generates the table of contents for your output. The Dynamic HTML format provide additional capabilities for users to prevent generation of the table of contents or to change the file name of the generated table of contents.

In the **File Processing** section of the target settings, you can also specify whether to include the table of contents, front matter, and index from your source documents in your generated output.

## To specify additional table of contents target settings

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. *If you want to prevent ePublisher from generating the table of contents*, set **Generate table of contents** to **Disabled**.
4. *If you want to change the file name of the generated table of contents*, specify the file name you want in the **Table of Contents filename** field.

## Defining the Table of Contents from an Irregular Heading Hierarchy

Many authors follow a regular heading hierarchy in their source documents. A **regular heading hierarchy** is one in which no levels are skipped in the hierarchy, as shown in the following example:

```
First Heading 1
  First Heading 2
    First Heading 3
      Second Heading 3
        Second Heading 2
Another Heading 1
  Another Heading 2
    Yet Another Heading 2
```

**Note:** If you use a regular heading hierarchy, this section does not affect your output.

Some documentation methodologies and processes require authors to skip levels in the hierarchy, which produces irregular heading hierarchies like the one shown in the following example:

```
First Heading 1
  First Heading 4
    Second Heading 4
      First Heading 3
        Second Heading 1
          Another Heading 3
            Still Another Heading 3
Final Heading 1
```

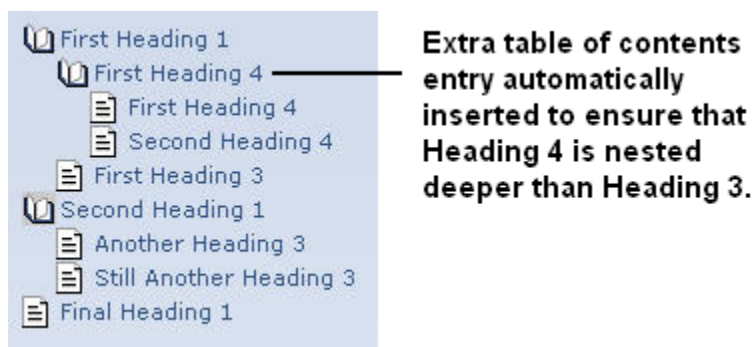
This heading hierarchy in a source document, produces an irregular table of contents hierarchy. The previous example is irregular because all **Heading 2** levels are skipped and several **Heading 3** levels are skipped in the hierarchy.

Since your generated table of contents is created from paragraphs in the source documents, irregular hierarchies can cause an aesthetically displeasing table of contents in your output. ePublisher provides several settings for how to transform irregular heading hierarchies, such as **Fully collapse**, **Don't collapse**, **Smart collapse**, and **Re-label**. By default, ePublisher uses **Smart collapse**.

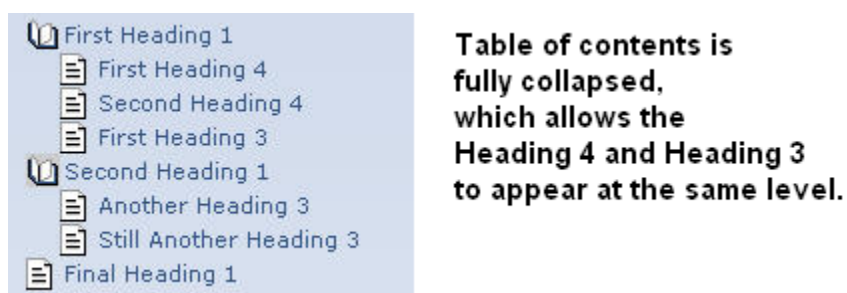


## To specify how ePublisher transforms irregular heading hierarchies

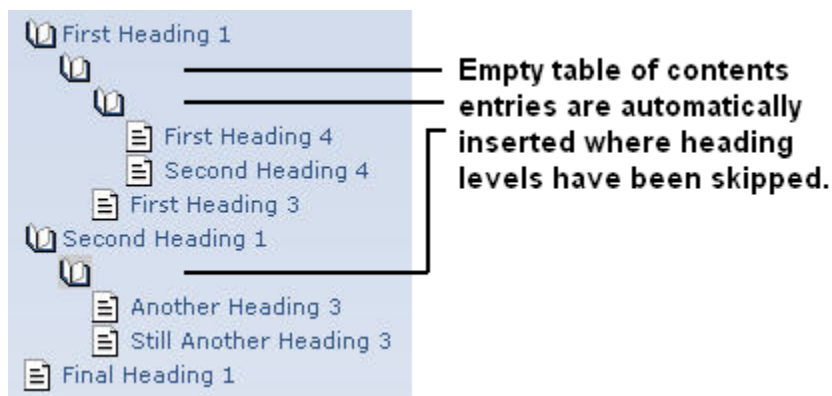
1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. **If you want to eliminate skipped levels from the table of contents except where needed to preserve the correct relative hierarchy**, set **Collapse table of contents** to **Smart collapse**. ePublisher removes skipped heading levels from the table of contents. However, if removing skipped levels results in, for example, a **Heading 3** and **Heading 4** displaying at the same level, ePublisher automatically inserts an entry into the table of contents to preserve the correct relative hierarchy. The irregular hierarchy in the previous example would be generated as follows.



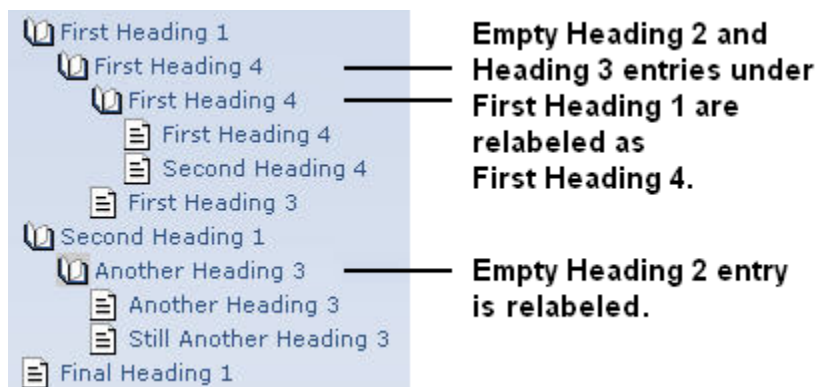
4. **If you want to eliminate all skipped levels from the table of contents**, set **Collapse table of contents** to **Fully collapse**. ePublisher removes all skipped levels, regardless of the level at which they appear in the source documents. The irregular hierarchy in the previous example would be generated as follows.



5. **If you want to duplicate the heading hierarchy of the source document with empty entries in the table of contents**, set **Collapse table of contents** to **Don't collapse**. ePublisher automatically inserts empty table of contents entries for the skipped levels. The irregular hierarchy in the previous example would be generated as follows.



6. *If you want to duplicate the heading hierarchy of the source document with labels in the table of contents*, set **Collapse table of contents** to **Re-label**. ePublisher automatically inserts labeled entries for the skipped levels. The heading text from the entry level appearing beneath the current entry is used as the label. The irregular hierarchy shown in the previous example would be generated as follows.



## Understanding the Table of Contents and Merge Settings

Not only can the table of contents be generated from defining your heading paragraph styles, the table of contents can also be generated using Merge Settings. Merge Settings allow you to combine multiple top-level groups into a single hierarchy. Not all output formats support merging. For more information see “Merging Top-level Groups (Multivolume Help)”.

**Note:** Since your Stationery design project is used to create Stationery, and Merge Settings are not stored in Stationery, you do not need to configure Merge Settings for your Stationery design project. You can configure the Merge Settings in your ePublisher Express projects, since these settings vary based on individual project needs.

## Defining the Icon for a Table of Contents Entry

For some output formats, ePublisher allows you to add a marker to a heading that defines the icon to use for the table of contents entry for that heading. For example, you can use a unique icon for new topics, or for specific types of information. ePublisher provides the TOCIconWWHelp, TOCIconHTMLHelp, TOCIconJavaHelp, and TOCIconOracleHelp markers.

## **Creating a Miniature Table of Contents**

A miniature table of contents (mini-TOC), also known as a partial table of contents, provides a list of the topics in the upcoming section. The topic titles are displayed as links beneath the current topic heading for easier navigation within the section. By default, ePublisher does not create a mini-TOC in topics within your output. You can configure ePublisher to generate mini-TOCs and you can define your mini-TOC levels in Style Designer. Before you begin, define the main table of contents for your projects. For more information, see “Defining the Table of Contents Structure (Levels)”. Your mini-TOC is derived from the table of contents level settings you define for the project.

## To create a mini-TOC

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the style you want to include a mini-TOC after.
5. On the **Options** tab, specify a value for the **Mini-TOC level** option. Specify a value that indicates the levels of the table of contents entries to include in the mini-TOC that follows the selected paragraph style. For more information about this option, click **Help**.

## Modifying the Appearance of Mini-TOC Entries

ePublisher stores CSS settings that control the appearance of mini-TOC entries in the `webworks.css` file. You can create an override file to modify these settings for specific levels of the mini TOC or for the entire mini TOC. For example, you can define a different font size and margin for each level in the mini TOC.

For more information about override files and locations, see “Stationery, Projects, and Overrides”.

## To modify the appearance of the mini TOC

1. **If you want to override the CSS settings for an output format**, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
  - b. Create the `Formats\formattype\Pages\css` folder in your project folder, where *formattype* is the name of the output format you want to override, such as `Microsoft HTML Help 1.x`.
2. **If you want to override the CSS settings for a target**, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Target Override Directory**.
  - b. Create the `Targets\targetname\Pages\css` folder in your project folder, where *targetname* is the name of the target you want to override.
3. Copy the `webworks.css` file from the following folder to the override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\formattype\Pages\css
```

4. Open the `webworks.css` file you copied to your project override folder.
5. **If you want to modify the appearance of the entire mini TOC**, find the code for `div.WebWorks_MiniTOC` and modify the values specified within the braces:
  - To modify the background color, specify the desired RGB color value for the `background-color` option.
  - To modify the border color, specify the desired RGB color value for the `border-color` option.
  - To modify the border width, specify the number of pixels you want for the `border-width` option.
  - To modify the type of border, specify the appropriate border style value for the `border-style` option.
  - To modify the spacing between the border and the text, specify the number of pixels you want for the `padding` option.

- To modify the font, specify the name of the font you want for the `font-family` option.

**6. *If you want to modify the appearance of a specific level of the mini TOC***, find the code for `div.WebWorks_MiniTOC_Levelx`, where x is the level number you want to modify. Then, specify the values within the braces to modify the font or margin:

- To modify the font of all mini-TOC entries for the specified level, specify the name of the font you want, such as `font-family: Arial;`.
- To modify the font size of all mini-TOC entries for the specified level, specify the size of the font you want, such as `font-size: 14pt;`.
- To modify the left margin indent of all mini-TOC entries for the specified level, specify the indent you want, such as `margin-left: 10px;`.

**7.** Save the `webworks.css` file.

**8.** Regenerate your project to review the changes.

For example, the following figure illustrates how you could customize your mini-TOC entries.

```
div.WebWorks_MiniTOC_Level1
{ font-size: 14pt;
  font-family: Arial;
  margin-left: 6px;
}
div.WebWorks_MiniTOC_Level2
{ font-size: 12pt;
  font-family: Arial;
  margin-left: 16px;
}
div.WebWorks_MiniTOC_Level3
{ font-size: 10pt;
  font-family: Arial;
  margin-left: 16px;
}
```

## Modifying the Appearance of Paragraphs

One of the most common modifications for online output is changing the appearance of text. Using Style Designer, you can select a paragraph style and then define the appearance of that style. This efficient method allows you to globally change all paragraphs with the same style at once, which increases your control and reduces maintenance costs.

# The Prototype Style for Paragraphs

The **Prototype** style is the parent to all other styles. When you set a property for the **Prototype** paragraph style, other paragraph styles inherit the value of that property. You can then override that value for specific styles as needed. The **Prototype** paragraph style allows you to quickly change a default property and apply that change to all paragraph styles within your Stationery project.

Depending on how you organized your styles in Style Designer, you may not have one style as the parent style on which all others are based. In Microsoft Word, usually all styles are based on **Normal**, but that may not always be the case. The **Prototype** style ensures that each of your styles has a parent style within your Stationery project.

## Setting the Background Color of a Paragraph

In terms of the CSS box model, the background for a paragraph refers to the background of the content and the padding areas. If you increase the padding for a paragraph style, the background color area for that style also increases. If you decrease the padding for a paragraph style, the background color area for that style also decreases.

### To set the background color of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Paragraph Styles**, select the style you want to modify.
4. On the **Properties** tab, click **Background**.
5. In the **Color** field, select a color from the drop-down menu, or type the RGB value of the color you want, such as `FFFFFF`.

## Setting the Border Style and Color of a Paragraph

Borders are lines that can be drawn around any or all of the four sides of a paragraph. In terms of the CSS box model, increasing the padding for a paragraph style increases the space between the paragraph and the border.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. The size and spacing of the dots in a dotted line may also be different between various browsers and operating systems.



### To set the border style and color of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Paragraph Styles**, select the style you want to modify.
4. On the **Properties** tab, click **Border**.
5. Click the tab for the side of the paragraph you want to display a border, and then specify the color, style, and width for that border. For more information about a property, click **Help**.

## Setting the Font for a Paragraph

Setting fonts for online output is an important step in making sure your content is properly displayed for your audience. Because many browsers and help systems use only the fonts available on the user's computer, you may not be able to use specific fonts, such as Times New Roman, as some computers may not have those fonts installed. You can specify a font family, such as sans-serif, to ensure a font of a similar type is used on each computer. You can also specify multiple fonts, separated by commas, to allow the browser to display the first available font.

### To set the font of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Paragraph Styles**, select the style you want to modify.
4. On the **Properties** tab, click **Font**.
5. Specify the family, size, style, and other properties you want to modify. For more information about a property, click **Help**.

## Setting the Width, Height, and Positioning of a Paragraph

In regards to the CSS box model, modifying the width and height of a paragraph adjusts the content box, which essentially defines the space that the paragraph uses. For example, if an existing paragraph stretches across the entire page, but you then adjust the width to 200 px, the paragraph is then limited to only 200 pixels wide.

## To set the width, height, and positioning of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Paragraph Styles**, select the style you want to modify.
4. On the **Properties** tab, click **HTML**.
5. Specify the appropriate values for the width, height, and positioning properties. For more information about a property, click **Help**.

## Adjusting the Space Around a Paragraph

You can adjust the white space around a paragraph by adjusting the margin and the padding. In terms of the CSS box model, modifying the padding property adjusts the space inside the border area. For example, if you create a border or background color for a paragraph and you increase the size of the padding, the border moves away from the text in the paragraph.

Modifying the margin properties adjusts the space outside the border area. For example, if you create a border or background color for a paragraph and you increase the size of the margins around the paragraph, the border remains the same distance from the text in the paragraph. However, the position of the paragraph changes since there is more white space between the modified paragraph and the other elements on the page. Review the following additional considerations:

- Set the right margin on paragraph styles to provide spacing between the text and the right edge of the window.
- For paragraphs used within bulleted and numbered lists, carefully adjust the left margin to correctly align with the text of the bulleted and numbered list items. Those list items may have different left margins to leave space for the bullet or number. For more information, see “Defining the Appearance of Bulleted Lists” and “Defining the Appearance of Numbered Lists”.
- When defining the left margin for a paragraph style used in a table, consider the size and weight of the font as well. For example, bold table headings need less pixels in their left margin than non-bold table text paragraphs so that both types of text align with each other.

### To set the margin and padding of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Paragraph Styles**, select the style you want to modify.
4. On the **Properties** tab, click **Margin**.
5. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** margin properties.
6. On the **Properties** tab, click **Padding**.
7. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** padding properties.

## Setting the Text Color and Other Characteristics of a Paragraph

In general, ePublisher generates output based on the properties of the content from the original source documents. You can modify the appearance of your online content without modifying your source documents. The text properties allow you to modify several important characteristics of the text:

- Color
- Kerning between letters and words
- Space between lines of a paragraph, also known as letting or line height
- Underlining, overlining, and strikethrough text
- Horizontal and vertical alignment of a paragraph
- Indentation of a paragraph

### To set the text characteristics of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Paragraph Styles**, select the style you want to modify.
4. On the **Properties** tab, click **Text**.
5. Specify the appropriate values for the text-related properties, such as **Color** and **Line height**. For more information about a property, click **Help**.

## Modifying Paragraphs for Bidirectional Languages

In some documents, such as those written with Arabic or Hebrew script, and in some mixed-language contexts, text within a single paragraph may appear with mixed directionality. For example, some characters are read from left to right, while others within the paragraph may be read from right to left. This phenomenon is called **bidirectionality**, or **bid** for short.

If a document contains right-to-left characters, and if the browser is able to display the language with the proper character set, the browser must apply the bidirectional algorithm. The proper character set means to not display arbitrary substitutes such as a question mark, a hex code, or a black box for some characters.

ePublisher allows you to make sure the browser correctly displays the content by offering the Unicode **Direction** and **Unicode Bidi** (bidirectional) properties.

### To set the Unicode properties of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Paragraph Styles**, select the style you want to modify.
4. On the **Properties** tab, click **Text**.
5. Under **Unicode**, set the appropriate values for the **Direction** and **Unicode Bidi** properties. For more information about these properties, click **Help**.

## Disabling Autonumbering in Output

In your source documents, you may have autonumbering enabled for print delivery purposes, such as printed manuals and PDFs. This function allows you to add autogenerated numbers to your chapters, headings, and figure captions. However, for some online delivery, you may not want to include the autonumbering in your online help.

## To disable autonumbering

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the style that has the autonumbering.
5. On the **Options** tab, set the **Keep paragraph numbering** option to **Disabled**.
6. Repeat steps 4-5 for every paragraph style for which you want to disable autonumbering.

## Defining the Appearance of Notes, Tips, Cautions, and Warnings

If you have the word `note`, `tip`, `caution`, or `warning` at the start of the paragraph, and the paragraph is defined with a hanging indent, increase the **Left** margin property in ePublisher to correctly align the word with the appropriate paragraph styles. ePublisher uses the **Left** margin property to define the hanging indent margin. It also uses the Text Indent property to place the first line of the paragraph to the left of the rest of the paragraph. For instructions on changing the hanging indent, see “Fixing Paragraph Indentation Including Hanging Indent”.

You may want to add an image to the left of each note, caution, tip, or warning in your generated output. ePublisher does not use images from Adobe FrameMaker reference pages. You can use the **Bullet** properties of a paragraph format to insert an image to the left of the paragraph. This process is similar to using an image for the bullet itself.

## Defining the Appearance of Bulleted Lists

By default, ePublisher generates output for bulleted lists based on the properties of the content from the original source documents. Some tools use the Symbol or Wingdings font for bullets. These fonts may not be available on all computers, which can cause the bullets in your output to not be displayed correctly. You can address this issue by using an image for your bullets, or you can create a character style with a font family assigned, such as sans-serif, and then apply that character style through ePublisher to the characters you choose to define for your bullets.

You may also need to increase the left margin of your bulleted list item styles to provide the correct spacing for the bullet and to allow the text to be properly aligned. In addition, you may need to adjust the left margins of both bulleted and numbered list items to get them to line up with each other. This alignment can

reduce the number of vertical columns on a page, which can help the user scan the information. This alignment can also let you use the same paragraph formats for paragraphs within either a bulleted or numbered list. For more information about additional margin adjustments, see “Fixing Paragraph Indentation Including Hanging Indent”.



## To create custom bullets

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the style you want to modify.
5. On the **Properties** tab, click **Bullet**.
6. ***If you want to create a custom bullet based on an image***, complete the following steps:
  - a. Make sure the image you want to use as your bullet image is stored in the **Files** folder in your project. You can copy the image file to this folder.
  - b. In the **Image** field under **Bullet Info**, select the image you want to use.
7. ***If you want to create a custom bullet based on a special character***, complete the following steps:
  - a. In the **Text** field, type the special character you want to use. You can type a special character using the Windows Alt key code.
  - b. In the **Separator** field, type the separator characters, such as a non-breaking space, that you would like to use between the bullet and the beginning of the first line of text.
  - c. In the **Character Style** field, select the defined character style you want to apply to the special character specified in the **Text** field. You can specify any character style that already exists in your project. The character style should use a font available on your users' computers. Avoid custom fonts, which can cause bullet characters to be displayed differently on various computers.

## Defining the Appearance of Numbered Lists

When you define the appearance of numbered list items, you need to define margins that leave the correct alignment for the numbers and the hanging indent. Since numbered items have a hanging indent, ePublisher uses a table in the generated output to create the appropriate format. Increase the **Left** margin property in ePublisher to correctly align the the hanging indented text. Then, set the **Flow Indent** text property for the paragraph to a negative value to leave enough space for double-digit numbered steps. For example, if your standard

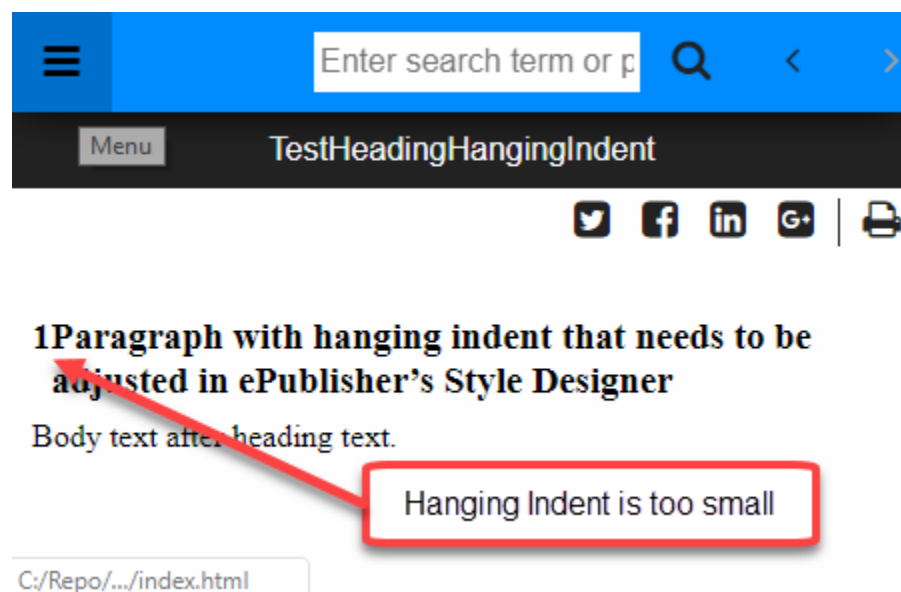
paragraph **Left** margin property is 30 pixels, you may want to set the **Left** margin property for your numbered list items to 70 pixels and set the **Flow Indent** text property to -30 pixels for your numbered list items. For instructions on changing the hanging indent, see “Fixing Paragraph Indentation Including Hanging Indent”.

You may also need to work with the left margins of both bulleted and numbered list items to get them to line up with each other. This alignment can reduce the number of vertical columns on a page, which can help the user scan the information. This alignment can also allow you to use the same paragraph formats for paragraphs within either a bulleted or numbered list.

## Fixing Paragraph Indentation Including Hanging Indent

It is very common to create paragraphs that use a hanging indent to offset the first line to the left of the rest of the paragraph. This can occur with headings, numbered lists, bulleted lists, and any other paragraph type.

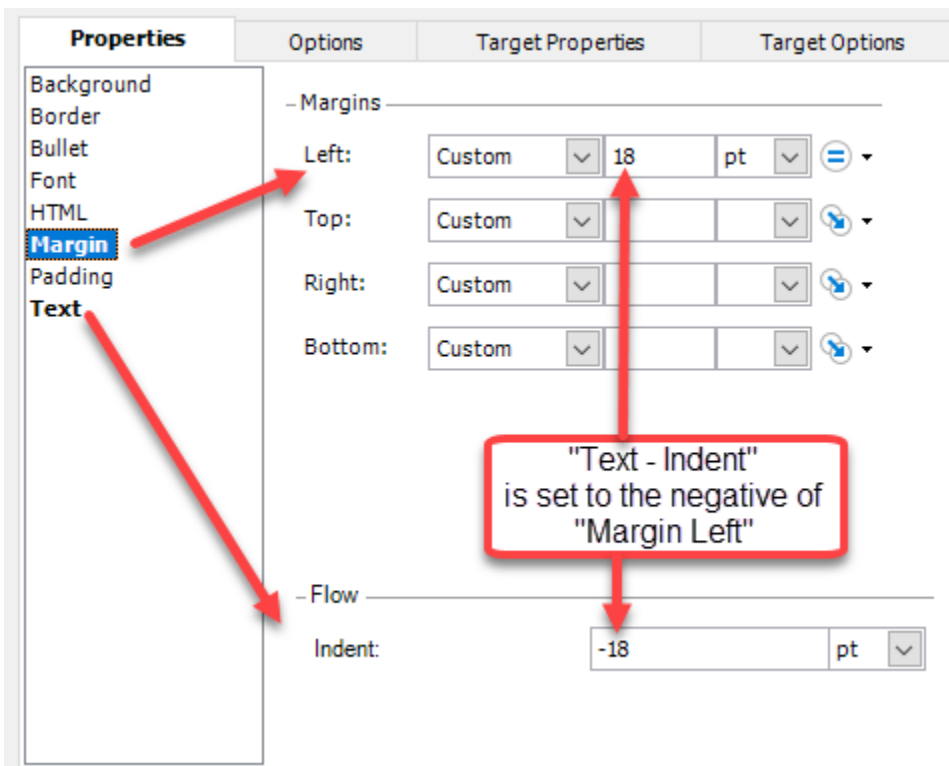
The following figure shows a heading paragraph that has a hanging indent that is not large enough in the generated output. Also, notice that there is not enough white space between the auto-number and the paragraph text.



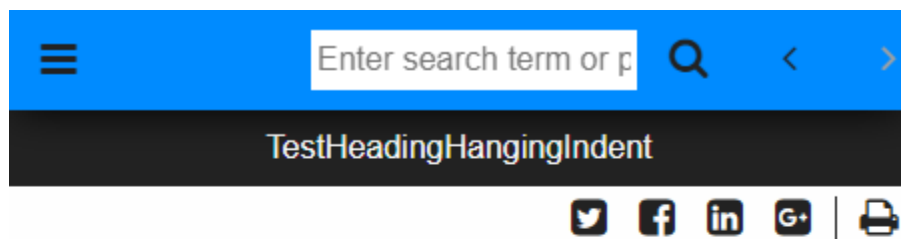
Using ePublisher's Paragraph Style Designer it is easy to increase the size of the hanging indent and thus increase the amount of white space between the auto-number and the paragraph text.

## Steps for changing the Hanging Indent

1. Open the Stationery Design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports the **Margin Left** and **Text Indent** properties.
4. In **Paragraph Styles**, select the style you want to modify.
5. On the **Properties** tab, click the name of the paragraph style to modify.
6. Select **Margin** and then for the **Left**, specify a value with units.
7. Select **Text** and then for the **Flow Indent**, specify the same value with units. However, the value must be the negative of that used for the **Margin Left**.



You may want to experiment with different values until you get something that looks good in the output. When generated it will look similar to the updated output shown below.



## 1 Paragraph with hanging indent that needs to be adjusted in ePubliher's Style Designer

Body text after heading text.

Hanging Indent corrected in output

## Modifying the Appearance of Characters

To modify the appearance of individual characters or words within a paragraph, you can use Style Designer to adjust the appearance of any character styles used in the source documents. This process allows you to optimize styles for print, using the styles in the source documents, and optimize the content for online presentation using the styles defined in ePubliher.

### The Prototype Style for Characters

The **Prototype** style is the parent to all other styles. When you set a property for the **Prototype** character style, other character styles inherit the value of that property. You can then override that value for specific styles as needed. The **Prototype** character style allows you to quickly change a default property and apply that change to all character styles within your Stationery project.

Depending on how you organized your styles in Style Designer, you may not have one style as the parent style on which all others are based. The **Prototype** style ensures that each of your styles has a parent style within your Stationery project.

### Setting the Background Color of a Character

In terms of the CSS box model, the background for a style refers to the background of the content and the padding areas. If you increase the padding for a style, the background color area for that style also increases.

### To set the background color of a character

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Character Styles**, select the character style you want to modify.
4. On the **Properties** tab, click **Background**.
5. In the **Color** field, select a color from the list, or specify the RGB value of a color, such as `FFFFFF`.

## Setting the Border Style and Color of Characters

Borders are lines that can be drawn around any or all of the four sides of a style. In terms of the CSS box model, increasing the padding for a style increases the space between the content and the border.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. The size and spacing of the dots in a dotted line may also be different between various browsers and operating systems.

### To set the border style and color of a character

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Character Styles**, select the character style you want to modify.
4. On the **Properties** tab, click **Border**.
5. Click the tab for the side of the character you want to display a border, and then specify the color, style, and width for that border. For more information about a property, click **Help**.

## Setting the Font for a Character

Setting fonts for online output is an important step in making sure your content is properly displayed for your audience. Because many browsers and help systems use only the fonts available on the user's computer, you may not be able to use specific fonts, such as Times New Roman, as some computers may not have those fonts installed. You can specify a font family, such as sans-serif, to ensure a font of a similar type is used on each computer. You can also specify multiple fonts, separated by commas, to allow the browser to display the first available font.

## To set the font of a character

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Character Styles**, select the character style you want to modify.
4. On the **Properties** tab, click **Font**.
5. Specify the family, size, style, and other properties you want to modify. For more information about a property, click **Help**.

## Adjusting the Space Around Characters

You can adjust the white space in all directions around characters by adjusting the margin and the padding. You can also adjust horizontal spacing by adjusting the kerning.

In terms of the CSS box model, modifying the padding property adjusts the space inside the border area. For example, if you create a border or background color for a character and you increase the size of the padding, the border moves away from the character.

Modifying the margin properties adjusts the space outside the border area. For example, if you create a border or background color for a character and you increase the size of the margins around the character, the border remains the same distance from the character. However, the position of the character changes since there is more white space between the modified character and other elements on the page.

### To set the margin, padding, and kerning of a character

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Character Styles**, select the character style you want to modify.
4. On the **Properties** tab, click **Margin**.
5. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** margin properties.
6. On the **Properties** tab, click **Padding**.
7. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** padding properties.
8. On the **Properties** tab, click **Text**.
9. In the **Letter spacing** field, specify a value and select the unit of measure to adjust the kerning. For more information about a property, click **Help**.

## Setting the Color and Other Characteristics of Characters

In general, ePublisher generates output based on the properties of the content from the original source documents. You can modify the appearance of your online content without modifying your source documents. The text properties allow you to modify several important characteristics of characters:

- Color
- Kerning between letters and words
- Underlining, overlining, and strikethrough text
- Vertical alignment to create subscripts and superscripts



### To set the text characteristics of a character

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Character Styles**, select the character style you want to modify.
4. On the **Properties** tab, click **Text**.
5. Specify the appropriate values on the text-related properties, such as **Color** and **Letter spacing**. For more information about a property, click **Help**.

## Modifying Characters for Bidirectional Languages

In some documents, such as those written with the Arabic or Hebrew script, and in some mixed-language contexts, text within a single paragraph may appear with mixed directionality. For example, some characters are read from left to right, while others within the paragraph may be read from right to left. This phenomenon is called **bidirectionality**, or **bid** for short.

If a document contains right-to-left characters, and if the browser can display the language with the proper character set, the browser must apply the bidirectional algorithm. If you prefer to control the handling of a particular phrase, you can apply a character style to that phrase and then define the character style with the **Direction** and **Unicode Bidi** properties.

## To set the Unicode direction of a character

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Character Styles**, select the character style you want to modify.
4. On the **Properties** tab, click **Text**.
5. Under **Unicode**, set the appropriate values for the **Direction** and **Unicode Bidi** properties. For more information about these properties, click **Help**.

## Defining the Appearance of Tables

Modifying the appearance of tables is different from modifying other elements in your content, such as paragraphs. The properties of tables are controlled by different layers. Some objects are in front of, or behind, other objects within the table. For example, the background property of the entire table is the first layer. The body, header, and footer properties reside in the next layer, which is on top of the background property. Paragraph properties are on top of that, followed by character properties, which are on the topmost layer.

With this model, you may have difficulty achieving the results you want when you try to adjust the appearance of a property for a table. You may need to experiment with the different property layers to fine-tune the appearance of tables. For example, if you try to create a transparent table by properly setting the body background property, but the table is not transparent, another layer may not be transparent. You need to make sure the background property is properly set for each layer in the table, including the table header, the paragraph, and the character styles.

Paragraph styles, such as CellHeading, CellBody, CellStep, and CellBullet give you additional control over formatting within tables in your generated output. When defining the left margin for a paragraph style used in a table, consider the size and weight of the font. For example, bold table headings need less pixels in their left margin than non-bold table text paragraphs so that both types of text align with each other.

## The Prototype Style for Tables

The **Prototype** style is the parent to all other styles. When you set a property for the **Prototype** table style, other table styles inherit the value of that property. You can then override that value for specific styles as needed. The **Prototype** table style allows you to quickly change a default property and apply that change to all table styles within your Stationery project.

## Setting the Background Color or Image of a Table

You can specify a color to use as the background of a table. You can also make the background transparent or specify an image to use as the background. In addition, you can specify alternative colors or images to alternate the appearance of rows or columns. Alternate shading of rows or columns is a useful layout to help minimize the number of lines and amount of information displayed, while organizing the data in a way that users can easily read and understand.

The background image for a table is behind other elements in a table. If you set a background color for the table, you cannot see the background image for the table. In addition, make sure the image is stored in the `Files` folder so it is part of the project.

## To set the background color or image of a table

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Table Styles**, select the table style you want to modify.
4. On the **Properties** tab, click **Body Background**.
5. **If you want to specify a color for the background of a table**, select a color in the **Color** field, or type the RGB value of the color you want, such as `FFFFFF`. To make the table transparent, click the **Web** tab in the **Color** field, and then click **Transparent**.
6. **If you want to specify an image for the background of a table**, complete the following steps:
  - a. Save the image file in the `Files` folder for the project. ePublisher copies files from the `Files` folder when you generate the project.
  - b. In the **Image** field under **Background Image**, select the image you want to use. ePublisher lists only the image files stored in the `Files` folder.
  - c. Specify the tiling, scrolling, and position properties to position the image as you want it.
7. **If you want to alternate the appearance of rows or columns in the table**, specify the appropriate values for the **Alternate Information** properties. For more information about a property, click **Help**.

## Setting the Border Style and Color of a Table

The border properties modify the appearance of the border that surrounds the outside of the table. However, some browsers display this information in different ways. For example, some browsers use the color and not the style of the border properties to define all other borders inside the table, unless that color has been previously defined.

For example, if you choose a red, dotted border for your table, the preview pane displays the outer edge of the table as a red, dotted border. All table cells inside the table have a red, solid border, unless the border properties for **Body** and **Header** (if applicable) are also defined. Not all browsers display the table the same way, so verify the results in multiple browsers when defining the table border and style.

### To set the border style and color of a table

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Table Styles**, select the table style you want to modify.
4. On the **Properties** tab, click **Border**.
5. Click the tab for the side of the table you want to display a border, and then specify the color, style, and width for that border. For more information about a property, click **Help**.

## Setting the Width and Height of a Table

You can define a fixed width and height for a table style. Make sure all the content of your tables will fit within the fixed height and width you specify. You can also use the table cell widths defined in your source documents.

## To set the height and width of a table

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Table Styles**, select the table style you want to modify.
4. ***If you want to define a fixed width and height for the table style,*** complete the following steps:
  - a. On the **Properties** tab, click **HTML**.
  - b. Specify the appropriate values for the **Width** and **Height** properties. For more information about a property, click **Help**.
5. ***If you want to use the cell widths defined in your source documents,*** on the **Options** tab, set **Use document cell widths** to **Enabled**.

## Setting the Vertical and Horizontal Alignment within a Table

You can specify the vertical and horizontal alignment of content within table cells. To define the vertical alignment, set the **Alignment** property for the table. To define the horizontal alignment, set the **Horizontal** alignment property for the paragraph style of the text in the table. These property values take effect unless a value is set at the paragraph style level, or a value is set for the table **Body**, **Header**, or **Footer** properties, if applicable.

## To set the alignment of content within a table

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Table Styles**, select the table style you want to modify.
4. **If you want to define the vertical alignment**, complete the following steps:
  - a. On the **Properties** tab, click **Table**.
  - b. Specify the appropriate value in the **Vertical** field under **Alignment**.
5. **If you want to define the horizontal alignment**, complete the following steps:
  - a. In **Paragraph Styles**, select the paragraph style you want to modify.
  - b. On the **Properties** tab, click **Text**.
  - c. Specify the appropriate value in the **Horizontal** field under **Alignment**.

## Adjusting the Space Within and Around a Table

After creating the external borders for your table, you can define the padding to specify the distance between the content and the borders within the table. This feature, which is enabled through CSS, is not supported by all browsers. View the output in several different browsers to verify your results.

You can also define the space around the table by adjusting the margin properties. Modifying the margin properties adjusts the space outside of the border area. For example, if you create a border for a table and you increase the size of the margins around the table, the border remains the same distance from the content of the table. However, the effective size of the table increases since there is more space between the table border and the other elements on the page.

## To adjust the padding and margin of a table

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Table Styles**, select the table style you want to modify.
4. On the **Properties** tab, click **Table**.
5. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** padding properties.
6. On the **Properties** tab, click **Margin**.
7. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** margin properties.

## Modifying Header, Footer, and Body Rows of a Table

In addition to modifying the appearance of an entire table, you can modify portions of a table, such as the header row, the footer row, and rows within the body of the table. To modify the appearance of a section of a table, you need to correctly define the parts of the table in your source documents. For example, in Microsoft Word, you need set the table property to define the header row. If you do not create a header row for a table in your source document, you cannot modify the header row appearance in your output. The first row of a table is not, by default, a header row.

To define the appearance of the header rows, specify the appropriate values for the **Header** and **Header background** properties for your table styles. These property values override the values specified for the **Table**, **Border**, and **Background** properties.

To define the appearance of the body rows, specify the appropriate values for the **Body** and **Body background** properties for your table styles. These property values override the values specified for the **Table**, **Border**, and **Background** properties.

To define the appearance of the footer rows, specify the appropriate values for the **Footer** and **Footer background** properties for your table styles. These property values override the values specified for the **Table**, **Border**, and **Background** properties.

Tables created in Microsoft Word do not identify footer rows. To control footer rows with Microsoft Word and ePublisher, set up your table in your source document to reflect the desired appearance, or use footer paragraph styles in the footer row of



the table. Then, ePublisher can use the table set up from your source document and you can use the footer paragraph styles to modify the appearance as needed.

Tables created in Adobe FrameMaker identify the footer rows of tables, which allows you to specify the **Footer** and **Footer background** properties to modify footer rows for online delivery.

## Modifying Cells of a Table

Modifying certain aspects of a table cell may also require you to modify the properties associated with the paragraphs that reside in the table cell. For example, if you make a table background color transparent, you may also need to modify the background color of the paragraphs in the table to make those cells transparent.

Cell spacing is used to create a transparent space between cells of a table. This capability allows you to create unique border structures, as the background color of a table can be seen through the space between table cells, which can have a different background color.

### To adjust the spacing between cells

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Table Styles**, select the table style you want to modify.
4. On the **Properties** tab, click **Table**.
5. In the **Cell spacing** field, specify an appropriate value.

## Defining the Appearance of Images

In general, ePublisher automatically transforms images from your source document to an optimized version for online distribution. However, if necessary, you can manually modify individual images or groups of images by applying a graphic style to the specific image or images you want to control. For example, you can define how images are resized, the maximum height and width for images, and the format for those images.

## Supported Image Formats

ePublisher supports several image formats and rasterizes all other formats:

### **.jpg files**

Provides a good format for photographs and images with many gradual variations in color. Although this format is compressed, the large number of colors can increase the size of the file when compared to the other supported image formats. The lossy data compression also reduces detail each time you save an image in this format. This format is not recommended for screen shots.

### **.gif files**

Provides a good format for screen shots, since the number of colors can be reduced and the need for gradual variations in color is minimal. This format supports transparency and animation. However, this format supports a reduced color palette of 256 RGB colors (8-bit).

### **.png files**

Provides a good multipurpose format for online presentation. This format supports up to 16 million RGB colors (24-bit) and uses lossless data compression.

### **.svg files**

Provides a powerful format that is not supported in all output formats. Some browsers do not support `.svg` image files. If you use `.svg` image files, you need to configure the `.svg` options to specify whether to rasterize these images.

You can use **passthrough** conditions and coding to include any type of file within your final output, such as Flash `.swf` files. For example, you can create a paragraph style in your source documents that is not included in the print version of your documentation. You can apply a condition to this paragraph that has the **passthrough** setting selected in ePublisher. Then, this paragraph can provide the exact coding to include in your output. Be sure to include any referenced files in your `Files` folder of your project so ePublisher copies those files to your output location.

## Image Quality and Processing

If ePublisher cannot use an original image in the output, or if ePublisher determines it needs to modify the image based on how it is included in the source document, ePublisher rasterizes the image using the options you define for your graphic styles in Style Designer. For example, you can define the dots per inch (DPI) and format for the final images. Rasterization of an image can cause the image to be blurry or distorted in the output.

For more information and specific considerations about your images, see “Image Formats and Considerations in FrameMaker” and “Image Styles and Considerations in Word”.

## The Prototype Style for Images

The **Prototype** style is the parent to all other styles. When you set a property for the **Prototype** graphic style, other styles inherit the value of that property. The **Prototype** graphic style allows you to quickly change a default property and apply that change to all graphic styles within your Stationery project.

## Defining Graphic Styles

All images are, by default, set to the **Prototype** style. If you need to modify any image properties, you can do so through the **Prototype** style. However, if you want to control a smaller set of images, even just one image, you need to assign a unique graphic style to those images.

### To define a graphic style

1. Create a marker named GraphicStyle in your FrameMaker template.
2. Open your Stationery design project.
3. On the **View** menu, click **Style Designer**.
4. In **Graphic Styles**, define a set of graphic styles. Writers can use a marker or field code to specify the graphic style to apply to each image.

## Setting the Border Style and Color of an Image

Borders are lines that can be drawn around any or all of the four sides of an image. In terms of the CSS box model, increasing the padding for a graphic style increases the space between the image and the border.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. The size and spacing of the dots in a dotted line may also be different between various browsers and operating systems.

## To set the border style and color of an image

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Graphic Styles**, select the graphic style you want to modify.
4. On the **Properties** tab, click **Border**.
5. Click the tab for the side of the image you want to display a border, and then specify the color, style, and width for that border. For more information about a property, click **Help**.

## Modifying the Width, Height, and Positioning of an Image

ePublisher automatically transforms images in your source document into Web-ready formats. However, the size of your print image may not be appropriate for online delivery. ePublisher provides several ways to modify the size of your images for online delivery without affecting the original images.

If you know the exact dimensions you want to assign to an image, you can use the height and width properties of a graphic style. However, you are defining the dimensions of all images that use that graphic style. Unless you want all your images to be the same size, such as 150 pixels high and 275 pixels wide, this option is not the most effective way to modify the size of your images. For most situations, it is more efficient to define the maximum height and width for an image as opposed to assigning a fixed height and width. For more information, see “Setting the Maximum Width and Height for Images” and “Modifying Image Size by Scale”.

In most cases, you probably want to leave your images positioned as they are in your source documents. To visually enhance the layout and presentation of your online images, ePublisher allows you to set the position of any image according to CSS rules.

## To set the width, height, and positioning of an image

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Graphic Styles**, select the graphic style you want to modify.
4. On the **Properties** tab, click **HTML**.
5. Specify the appropriate values for the width, height, and positioning properties.

## Adjusting the Space Around Images

You can adjust the white space around images by adjusting the margin and the padding. In terms of the CSS box model, modifying the padding property adjusts the space inside the border area. For example, if you create a border or background color for an image and you increase the size of the padding, the border moves away from the image.

Modifying the margin properties adjusts the space outside the border area. For example, if you create a border or background color for an image and you increase the size of the margins around the image, the border remains the same distance from the image. However, the position of the image changes since there is more white space between the modified image and other elements on the page.

### To set the margin and padding of an image

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Graphic Styles**, select the graphic style you want to modify.
4. On the **Properties** tab, click **Margin**.
5. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** margin properties.
6. On the **Properties** tab, click **Padding**.
7. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** padding properties.

## Using Thumbnails for Images

ePublisher allows you to automatically reduce the size of images in your online content to thumbnails, which are reduced versions of your images. Users can view the full-sized image by clicking on the thumbnail. If you set the width and height values for thumbnails, ePublisher automatically creates the thumbnails you need and links them to the full-sized images. If you do not set a width and height for thumbnails, ePublisher inserts the image itself in your output and does not use thumbnails.

## To use thumbnails for images

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, specify values, in pixels, for the **Maximum thumbnail width** and **Maximum thumbnail height** options.

## Setting the Maximum Width and Height for Images

Most images in your content vary in width and height. For your online content, you need to ensure that all your images fit within the display area. In most cases, you have a maximum width or height constraint. For example, although some of your images may only be 250 pixels wide, you want to make sure that none of your images are wider than 275 pixels.

ePublisher allows you to set maximum width and height values for the size of your images. By modifying the size of your images in this way, you ensure that the resized images retain their original aspect ratio.



## To set the maximum width and height of images

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, specify values, in pixels, for the **Maximum image width** and **Maximum image height** options.

## Modifying Image Size by Scale

If you are not concerned with actual width and height of an image, you can specify a scaling percentage to apply to all images with the selected graphic style. This option allows you to modify the size of all images associated with the graphic style in relation to their original sizes. Modifying images in this way retains the original aspect ratio of each image.

## To resize images based on a percentage

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, specify a percentage value for the **Scale %** option. For example, if you specify 50, each image with the selected graphic style will be reduced to half its original size.

## Modifying Image Resolution

If the resolution of the images in your source documents is set for print, such as a resolution of 300 dots per inch (dpi) or higher, you may want to reduce the image resolution for online delivery. High dpi images create larger files that require more time to download. In addition, most monitors display a resolution of 96 dpi, so higher resolutions do not increase the quality of the image displayed online.

Although transforming an image from 300 dpi to 96 dpi helps optimize your images for online delivery, the width and height of your images is also affected. Because a resolution of 300 dpi contains more dots per inch than a resolution of 96 dpi, when ePublisher transforms the image, the image will be roughly 68% smaller than the original image. For example, a 300 dpi image that is 100 x 100 pixels will be 32 x 32 pixels when transformed to a 96 dpi image.

To counter this effect, use the **Scale %** option in conjunction with the **Render DPI** option to control the size of your images. In the example of transforming a 300 dpi image to 96 dpi, set the **Scale %** option to **312**, which then generates an image that has roughly the same dimensions as the original source image. You can also use the **Scale %** and **Render DPI** options together with the **Maximum image height** and **Maximum image width** options to make sure your images are correctly sized for online delivery.

### To set the image resolution (dpi)

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, specify a value in dpi for the **Render DPI** option.

## Setting Color Bit Depth

The range of colors available through digital computer images is usually expressed in terms of bit depth. This expression refers to the number of bits of data carrying the color information. Common bit depth levels for images include 8-bit and 24-bit color. In general, more bits of data make more colors available. The **Color bit depth** option applies only to `.gif` and `.png` images. `.jpg` images always generate 32-bit images.

## To set the color-bit depth for .gif and .png images

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, select a value for the **Color bit depth** option.

## Choosing an Image File Format and Quality Level

By default, if ePublisher needs to rasterize an image, it transforms the image in your source documents, regardless of what file format it is, to a Web-ready `.jpg` image. In some instances, you may want to use other image formats for online delivery. For example, `.gif` images can produce similar quality images as `.jpg`, but the file size is smaller. `.gif` images can also support transparent colors. You can also create `.png` images, which combine some of the better qualities of both `.jpg` and `.gif`.

If you select **JPG** in the **Format** options for a graphic style, you can specify the quality of the images. The quality level impacts both the visual aspects of your images and the size of the generated files. Higher-quality images require larger files, which require more time to download and display. The **JPG Quality** option does not affect `.gif` or `.png` images.

## To choose the file format and quality for online images

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, select a value for the **Format** option.
6. In the **Output file extension** field, specify the proper file extension based on the value you selected for the **Format** option, such as **.jpg**, **.gif**, or **.png**.
7. ***If you selected JPG in the Format option***, in the **JPG Quality** field, specify the percentage value you want for your online images. A value of 100 creates the highest quality image that mimics your original image.

## Creating Grayscale Images

ePublisher allows you to transform your original color images into grayscale versions for online viewing. If you enable the **Grayscale** option, ePublisher removes the color saturation of the original images when those images are transformed with your online content. ePublisher displays the color versions of your images in the preview pane to generate the preview more quickly.

### To create grayscale images

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, set the **Grayscale** option to **Enabled**.

## Setting Transparency for .gif and .png Images

In some images, you may want to set a color to transparent. For example, if your source document has a white background, images with a white background appear as though they do not have a background. However, if your online content has a different color background, the background of these images appear as white areas. You can enable the transparency option in ePublisher to transform the white background into a transparent one. Only `.gif` and `.png` images support transparent colors.

**Note:** Some browser versions do not support transparent colors, especially in `.png` images.

## To set transparency for .gif and .png images

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Graphic Styles**, select the graphic style you want to modify.
5. On the **Options** tab, set the **Transparent** option to **Enabled**.

## Defining the Appearance of Pages

The way you present your online content greatly impacts the usability and readability of that content in the online environment. Information layout and design for print and online content are usually very different. ePublisher allows you to format your source documents to optimize for printed delivery, and then use ePublisher to deliver the content optimized for online presentation.

Unlike printed content, online documentation can include extra navigation features, follow different rules regarding topic length and page breaks, and include unique page layouts for specific types of information. While each output format varies in the types of files that are delivered, most output formats are based on HTML. Therefore, when you decide how to present your content online, consider the layout of your content on an HTML page.

In ePublisher, the overall appearance of each topic of your output is controlled by page styles. You can modify page style properties with Style Designer to define whether to include navigation browse buttons on each page, whether to include company logo and contact information, and other design elements for each page that uses a specific page style. To deliver your online content as individual chunks, or pages of information, define page breaks based on paragraph styles, such as headings. For more information, see "Defining New Pages (Page Breaks)".

## The Prototype Style for Pages

The **Prototype** style is the parent to all other styles. When you set a property for the **Prototype** page style, other page styles inherit the value of that property. You can then override that value for specific styles as needed. The **Prototype** page style allows you to quickly change a default property and apply that change to all page styles within your Stationery project.

## Displaying Company Logo and Information on a Page

Most HTML-based output formats support adding company information, such as company name, email address, and company logo, as part of a page. Once you specify your company information in the **Target Settings** for your project, you can select different locations in which to place the content on each page. If you specify a value for **Company webpage**, ePubliher links the specified company name to the specified Web page. If you specify **Company email address**, ePubliher creates a `mailto:` link to the specified address.

If you add a logo image to your project, ePubliher displays the logo next to your company contact information on the top or the bottom of your output pages. To include a logo, you must first store the image file in the `Files` folder of your project.



## To specify your company information and the location on the page

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. Specify the appropriate values for the **Company Information** settings, such as **Company name** and **Company phone number**. If you do not see these fields in the list of target settings, the output format for the active target does not support this feature. For more information about a setting, click **Help**.
4. Click **OK** to save the target settings.
5. On the **View** menu, click **Style Designer**.
6. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
7. In **Page Styles**, select the page style you want to modify.
8. On the **Options** tab, specify the appropriate values for the **Company info...** options. For more information about an option, click **Help**.

**Note:** Company information is not displayed in the Preview pane.

## Modifying the Appearance of the Company Information

You can modify the appearance of the company information by creating an override for the `Page.asp` file. This file controls the content coding for each page of your content. When you modify this file, you need to change the appropriate part of the file based on whether you want to modify the appearance of the company information at the top of the page or at the bottom of the page. You can also modify many other aspects of your content pages.

## To modify the appearance of the company information

1. Create an override file for the `Page.asp` file by copying it from the installation folder to your format or target override folder in your project, based on whether you want to modify the appearance of the company information for an output format or for a specific target. For more information, see "Format and Target Overrides".
2. Open the `Page.asp` file you copied to the `Formats\formatname\Pages` or `Targets\targetname\Pages` folder in your project.
3. Locate the code at the top or bottom of the `Page.asp` file, based on whether you want to change the appearance of the information at the top or the bottom of the page.

- To modify the top of the page, find the following line of code:

```
<table wwpage:condition="company-info-top" align="left"  
wwpage:attribute-align="company-info-top-alignment">
```

- To modify the bottom of the page, find the following line of code:

```
<table wwpage:condition="company-info-bottom" align="right"  
wwpage:attribute-align="company-info-bottom-alignment">
```

4. **If you want to modify the appearance of the company name**, create an override for `webworks.css` from the `Formats\[your format]\Pages\css` installation directory. Then, find `td.WebWorks_Company_Name_Top` in `Pages\css\webworks.css`, and modify the CSS attributes, such as the `font-size` or `font-family`, and required.
5. **If you want to modify the appearance of the company phone number**, find the following code immediately after the line of code you found at the top or bottom of the file:

```
<tr wwpage:condition="company-phone-exists">
```

```
<td class="WebWorks_Company_Phone_Bottom">
```

```
or <td class="WebWorks_Company_Phone_Top">
```

Look for `td.WebWorks_Company_Phone_Bottom` or `td.WebWorks_Company_Phone_Top` in `Pages\css\webworks.css` and change the `font-size` or `font-family`

6. **If you want to modify the appearance of the company fax number**, find the following code immediately after the line of code you found at the top or bottom of the file:

```
<tr wwpage:condition="company-fax-exists">
```

```
<td class="WebWorks_Company_Fax_Bottom">
```

or <td class="WebWorks\_Company\_Fax\_Top">

Look for `td.WebWorks_Company_Fax_Bottom` or `td.WebWorks_Company_Fax_Top` in `Pages\css\webworks.css` and change the `font-size` or `font-family`

- 7. If you want to modify the appearance of the company email address,** find the following code immediately after the line of code you found at the top or bottom of the file:

```
<tr wwpage:condition="company-email-exists">
```

```
<td class="WebWorks_Company_Email_Bottom">
```

or <td class="WebWorks\_Company\_Email\_Top">

Look for `td.WebWorks_Company_Email_Bottom` or `td.WebWorks_Company_Email_Top` in `Pages\css\webworks.css` and change the `font-size` or `font-family`

## Setting the Background Color or Image of a Page

You can specify a color to use as the background of a page. You can also specify an image to use as the background. This capability allows you to include a watermark, such as `DRAFT` for initial internal versions of your online content. The background image for a page is behind other elements on the page. If you set a background color for the page, you cannot see the background image for the page. Make sure the image is stored in the `Files` folder for the project so the image file is copied to your output folder. Also make sure the image does not make the text too difficult to read.

## To set the background color or image of a page

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Page Styles**, select the page style you want to modify.
4. On the **Properties** tab, click **Background**.
5. **If you want to specify a color for the background of a page**, select a color in the **Color** field, or type the RGB value of the color you want, such as `FFFFFF`.
6. **If you want to specify an image for the background of a page**, complete the following steps:
  - a. Save the image file in the `Files` folder for the project. ePublisher copies files from the `Files` folder when you generate the project.
  - b. In the **Image** field under **Background Image**, select the image you want to use. ePublisher lists only the image files stored in the `Files` folder.
  - c. Specify the tiling, scrolling, and position properties to position the image. For more information about a property, click **Help**.

## Setting the Border Style and Color of a Page

Borders are lines that can be drawn around any or all of the four sides of a page. In terms of the CSS box model, increasing the padding for a page style increases the space between the content within the page and the border of the page.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. The size and spacing of the dots in a dotted line may also be different between various browsers and operating systems. Some browsers may not display page borders at all. If you want a page border, view the generated output in multiple browsers to verify that your settings create the appearance you want.

### To set the border style and color for a page

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Page Styles**, select the page style you want to modify.
4. On the **Properties** tab, click **Border**.
5. Click the tab for the side of the page you want to display a border, and then specify the color, style, and width for that border. For more information about a property, click **Help**.

## Adjusting the Space Around a Page

You can adjust the white space around a page by adjusting the margin and the padding. In terms of the CSS box model, modifying the padding property adjusts the space inside the border area. For example, if you create a border or background color for a page and you increase the size of the padding, the border moves away from the content in the page.

Modifying the margin properties adjusts the space outside the border area. For example, if you create a border or background color for a page and you increase the size of the margins around the page, the border remains the same distance from the content in the page. However, the position of the content changes because the space outside the border increases.

### To set the margin and padding of a paragraph

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Page Styles**, select the page style you want to modify.
4. On the **Properties** tab, click **Margin**.
5. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** margin properties.
6. On the **Properties** tab, click **Padding**.
7. Specify a value and select the unit of measure for the **Left**, **Right**, **Top**, and **Bottom** padding properties.

## Using a Custom CSS to Modify the Appearance of Content

You can use a custom CSS file to modify the appearance of your content instead of using Style Designer for HTML-based output formats. By using style names in your source document that match the classes defined in your custom CSS file, such as **div.Heading1**, you can make sure your content uses your custom CSS file. You do not need to create any other association between styles in your source documents and the styles in Style Designer. Make sure you define each tag and class from your generated output in your custom CSS file.

### To use a custom CSS file with your project

1. Store your custom CSS file in the `Files` folder in your project.
2. Open your Stationery design project.
3. On the **View** menu, click **Style Designer**.
4. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
5. In **Page Styles**, select the page style you want to modify.
6. On the **Options** tab, select your custom CSS file in **Custom document css file**.

## Modifying the Location and Separators of Breadcrumbs

Breadcrumbs form a linked path to show users the location of the current topic in your online content. This clickable path steps you through the topics in the hierarchy above the current topic. You can display breadcrumbs at the top of the page, at the bottom of the page, or both. The breadcrumb trail at the top of the output page is enabled by default.

## To modify the location of the breadcrumb trail

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Page Styles**, select the page style you want to modify.
5. On the **Options** tab, select the appropriate value for the **Breadcrumbs shown at top of page** and **Breadcrumbs shown at bottom of page** options.
6. On the **Properties** tab, click **Navigation**.
7. Specify the appropriate values for the **Breadcrumbs Separator** and **Alignment** properties. For more information about a property, click **Help**.

## Modifying the Appearance of Breadcrumbs

Breadcrumbs form a linked path to show users the location of the current topic in your online content. If you enabled breadcrumbs for your output, you have several options to define the appearance of the breadcrumbs. For more information, see "Modifying the Location and Separators of Breadcrumbs".

For more information about override files and locations, see "Stationery, Projects, and Overrides".



## To modify the appearance of the breadcrumb trail

1. **If you want to override the CSS settings for an output format**, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
  - b. Create the `Formats\formattype\Pages\css` folder in your project folder, where *formattype* is the name of the output format you want to override, such as `Microsoft HTML Help 1.x`.
2. **If you want to override the CSS settings for a target**, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Target Override Directory**.
  - b. Create the `Targets\targetname\Pages\css` folder in your project folder, where *targetname* is the name of the target you want to override.
3. Copy the `webworks.css` file from the following folder to the override folder you created within your project folder:  
`Program Files\WebWorks\ePublisher Designer\Formats\formattype\Pages\css`
4. Open the `webworks.css` file you copied to your project override folder.
5. Find the code for `div.WebWorks_Breadcrumbs` and modify the values specified within the braces:
  - To modify the text color, specify the desired RGB color value for the `color` option.
  - To modify the font, specify the name of the font you want for the `font-family` option.
  - To modify the size of the font, specify the size you want for the `font-size` option.
6. Save the `webworks.css` file.
7. Regenerate your project to review the changes.

## Choosing the Location of Navigation Browse Buttons

If you have included navigation buttons in your output, you can specify whether to display the browse buttons at the top or the bottom of each page. You can also specify whether to align the button along the right or left side. Not all output formats support navigation browse buttons.

## To modify the location of navigation browse buttons





1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Page Styles**, select the page style you want to modify.
5. On the **Options** tab, select the appropriate value for the **Navigation links shown at top of page** and **Navigation links shown at bottom of page** options.
6. On the **Properties** tab, click **Navigation**.
7. Specify the appropriate values for the **Navigation Alignment** properties. For more information about a property, click **Help**.

## Modifying the Navigation Browse Buttons

You can use customized navigation browse buttons in your online content. For more information about customizing the WebWorks Help navigation buttons, see “Customizing the Toolbar in WebWorks Help”.

## To change the navigation button images

1. ***If you want to override the images for an output format***, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
  - b. Create the `Formats\formattype\Pages\Images` folder in your project folder, where *formattype* is the name of the output format you want to override, such as `Dynamic HTML`.
2. ***If you want to override the images for a target***, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Target Override Directory**.
  - b. Create the `Targets\targetname\Pages\Images` folder in your project folder, where *targetname* is the name of the target you want to override.
3. Paste the `.gif` files you want to use with names identical to those you want to replace in the `Images` folder you created. The following table lists the default button images and their file names and sizes.

Image	Name	Image Size	Description
	<code>prev.gif</code>	Width: 0.722 inch Height: 0.333 inch	Sends the user to the previous HTML document.
	<code>prevx.gif</code>	Width: 0.722 inch Height: 0.333 inch	Displayed when there is no previous HTML document available.
	<code>next.gif</code>	Width: 0.722 inch Height: 0.333 inch	Sends the user to the next HTML document.
	<code>nextx.gif</code>	Width: 0.722 inch Height: 0.333 inch	Displayed when there is no next HTML document available.

4. Regenerate your project to review the changes.

## Associating a Page with a Page Style

By default, output is associated with the **Prototype** page style in ePublisher. Therefore, you do not have to do anything to associate pages with the **Prototype** page style. However, if you want to have more than one page layout, you need to use additional page styles.

## To associate a page with a page style

1. Define page breaks based on paragraph styles, such as heading styles.
2. Define additional page styles in the Stationery.
3. ***If you want to associate a page with a page style other than the Prototype page style***, add a PageStyle marker or field code in your source document after the new page style, such as the heading that starts the page. The PageStyle marker or field code identifies what page style to use for that page.

Provide writers with the defined page style names they need to specify in their PageStyle markers or field codes.

## Defining the Appearance of Links

You can customize the appearance of links, also known as hyperlinks and hypertext links. You can define the color and text decoration, such as underline, for the currently selected link (active link), links you have previously visited, links you hover over, and the links you have not previously visited. These link settings are stored in the `webworks.css` file. You can create an override for the `webworks.css` file and then modify the tags that define the links.

**Note:** These changes do not affect output formats that do not use cascading style sheets.

In your source files, apply a character style to identify the text to include in each link. The character style can also affect the appearance of the link.

## To change the color of links

1. In your Stationery design project, go to **Advanced** menu bar item and either click **Manage Format Customizations** or **Manage Target Customizations**.
2. Navigate to `Pages\css` in this window.
3. Double click the `webworks.css` file and it will be bolded to indicate that a customization has been created.
4. Double click again, and the CSS file should be opened in a text editor:
5. Find the following code and modify the values specified within the braces:

```
a:link:active { color: #0000CC }  
a:link:hover { color: #CC0033 }  
a:link { color: #3366CC }  
a:visited { color: #9999CC }
```

- To modify the text color, specify the desired RGB color value for the `color` option of the appropriate link type and state.
  - To add text decoration, such as an underline, add the `text-decoration: underline;` option to the type of link you want to have an underline, and add the `text-decoration: none;` option to the other types of link.
  - To make the link text bold when you hover over the link, add the `font-weight: bold;` option to the `a:link:hover` definition.
6. Save the `webworks.css` file.
  7. Save and close your project.
  8. Reopen your project and regenerate to review the changes.

## Saving a Snapshot (Backup Copy) of Your Project

As you continue to make changes to your Stationery design project, you may want to revert to a previous state, before you implemented a specific feature. You can save a backup copy of your project and then use this backup copy at a later time, if needed.

As you develop your Stationery design project, periodically save a snapshot of your project. You can use this snapshot to revert to your Stationery design project at a specific point in time. This snapshot can also help you if your design project or

Stationery become corrupted in the future. For more information about the specific files and folders, see “Backing Up Your Stationery Design Project, Stationery, and Projects”.



### **To save a snapshot (backup copy) of your project**

1. Save your project and close ePublisher.
2. Copy your project folder and all its subfolders and files, and your source documents folder, maintaining the same structure, to another location. For example, create a folder with the date from today as the name. Then, copy your project folder and your source documents folder into the folder you created. This copy is your snapshot.

## **Defining Marker Types**

If you want to create your own markers, you can define the behavior of each marker you create. ePublisher provides many default markers to provide various built-in features and capabilities.

<b>Default Marker Name</b>	<b>Description</b>
AbbreviationTitle	Specifies alt text for an abbreviation, such as SS#. This text is displayed when a user hovers over the abbreviation in the output.
AcronymTitle	Specifies alt text for an acronym, which is displayed when a user hovers over the acronym in the output.
Citation	Specifies a citation as alt text for a quote, which is displayed when a user hovers over the citation in the output.
Context Plugin	Specifies context plug-ins for Eclipse help systems. Other Eclipse plug-ins can use the context plug-in IDs to call the Eclipse help system. For more information, see "Using Markers to Specify Context Plug-ins in Eclipse Help".
Description	Specifies a formal description for the generated HTML page that from the source content at this location of the source file. In WebWorks Reverb 2.0, it is used for the contents of the search result summary. In all HTML-based formats, specifies the description to include in the <code>&lt;META&gt;</code> tag for the topic. This <code>&lt;META&gt;</code> tag can improve the search experience on the Web.
DropDownEnd	Marks the end of an expand/collapse section. For more information, see "Defining Expand/Collapse Sections (Drop-Down Hotspots)".
Filename	Specifies the name of an output file for a page or an image. For more information, see "Defining File Names".
GraphicScale	Specifies a percentage to resize an image, such as 50 or 75 percent.
GraphicStyle	Specifies the name of a graphic style defined in the project to apply to an image. This marker is

Default Marker Name	Description
	an internal marker name that is not displayed in Stationery Designer. You cannot create a marker with a different name and assign it this functionality. For more information, see "Defining the Appearance of Images".
ImageAltText	Specifies alt text for an image. This text is added to the <code>alt</code> attribute of the <code>img</code> tag in the output.
ImageAreaAltText	Specifies alt text for clickable regions in an image map. This text is displayed when a user hovers over the region in the output.
ImageLongDescByRef	Specifies the path to the file that contains the long description for an image. This description is used when you create accessible content.
ImageLongDescNotReq	Specifies that a long description is not required for an image, which bypasses this accessibility check for that image.
ImageLongDescText	Specifies the long description for an image. This description is used when you create accessible content.
Keywords	Specifies the keywords to include in the <code>META</code> tag for the topic. This <code>META</code> tag improves searchability on the Web.
PageStyle	Specifies the name of a page style defined in the project to apply to a topic. This marker is an internal marker name that is not displayed in Stationery Designer. You cannot create a marker with a different name and assign it this functionality. For more information, see "Defining the Appearance of Pages".
PassThrough	Specifies that ePublisher place the contents of the marker directly into the generated output without processing the content in any way. For example, you could use a PassThrough marker if you wanted to embed HTML code within your generated output.

Default Marker Name	Description
Popup	Specifies the start of the content to include in a popup window. The content is displayed in a popup window when you hover over the link. When you click the link in some output formats, the topic where the popup content is stored, such as the glossary, is displayed. For more information, see “Defining Popup Windows”.
PopupEnd	Marks the end of the content to include in a popup window.
PopupOnly	Specifies the start of the content to include in a popup window. The content is displayed in a popup window when you hover over or click the link.
RubiComposite	No longer supported.
SeeAlsoKeyword	Specifies an internal identifier for a topic. SeeAlsoLink markers in other topics can list this identifier to create a link to this topic. For more information, see “Defining See Also Links”.
SeeAlsoLink	Identifies an internal identifier from another topic to include in the list of see also links in this topic.
SeeAlsoLinkDisplayType	Specifies whether to display the target topics on a popup menu or in a window. By default, the links are displayed in the Topics Found window. To display a popup menu, set the value to <code>menu</code> . This marker is supported only in HTML Help.
SeeAlsoLinkWindowType	Specifies the name of the window defined in the hhp file, such as TriPane or Main, that the topic opens in when the user clicks the link. This marker is supported only in HTML Help.
TableStyle	Specifies the name of a table style defined in the project to apply to a table in versions of Microsoft Word that did not support table styles. This marker is an internal marker name that is not displayed in Stationery

<b>Default Marker Name</b>	<b>Description</b>
	Designer. You cannot create a marker with a different name and assign it this functionality.
TableSummary	Specifies a summary for a table, which is used when you create accessible content.
TableSummaryNotReq	Specifies that a summary is not required for a table, which bypasses this accessibility check for that table.
TOCIconHTMLHelp	Identifies the image to use as the table of contents icon for this topic in the HTML Help output format.
TOCIconJavaHelp	Identifies the image to use as the table of contents icon for this topic in the Sun JavaHelp output format.
TOCIconOracleHelp	Identifies the image to use as the table of contents icon for this topic in the Oracle Help output format.
TOCIconWWHelp	Identifies the image to use as the table of contents icon for this topic in the WebWorks Help output format.
TopicAlias	Specifies an internal identifier for a topic that can be used to create a context-sensitive link to that topic. For more information, see "Defining Context-Sensitive Help Links".
TopicDescription	Specifies a topic description for a context-sensitive help topic in Eclipse help systems. For more information, see "Using Markers to Specify Topic Descriptions for Context-Sensitive Help Topics in Eclipse Help".
WhatIsThisID	Identifies a What's This help internal identifier for creating context-sensitive What's This field-level help in the HTML Help output format.
WindowType	Specifies the name of the window defined in the help project that the topic should be displayed in. In HTML

Default Marker Name	Description
	Help, the window names are defined in the hhp file. This marker is supported in HTML Help and Oracle Help.

## Defining File Names

By default, ePublisher automatically assigns file names to your generated output files for topics (pages) and images (graphics). ePublisher assigns output file names using a default naming rule. You can customize this naming convention using one of the following methods:

- Specifying a different topic (page) or image (graphic) naming pattern for ePublisher to use in the target settings for your output.
- Inserting Filename markers into source documents that specify the topic (page) or image (graphic) file name ePublisher should use for the file when generating output.

## Specifying File Names for Pages Using Page Naming Patterns

By default, ePublisher uses the following values when specifying file names for pages:

**\$D;.\$DN;.\$PN**

This specifies that ePublisher name the files using the following syntax when it generates page files for a target:

*SourceDocumentName.SourceDocumentNumber.TopicNumber*

The parts of the default naming rule are defined as follows:

### ***SourceDocumentName***

Identifies the name of the source document that the topic came from without the file extension.

### ***SourceDocumentNumber***

Identifies the number of the source document in the order it is included in its containing group in the project, such as 1 for the first source document in a group and 2 for the second source document in a group. This value starts at 1 for the first source document in each group in your project.

### ***TopicNumber***

Identifies the number of the topic (output page) generated from the source document, such as 1 for the first topic generated from a source document and 2 for the second topic generated from a source document. This value starts at 1 for the first topic in each source document.

For example, if you have a `MyFile.fm` source document that contains three topics that start with a paragraph style that has a page break priority set in Style Designer, ePublisher generates the following default output file names: `MyFile.1.1.html`, `MyFile.1.2.html`, and `MyFile.1.3.html`. You can change the default file extension for each page style.

You can use the following values to specify a page file naming pattern for a target:

**Note:** Each value you specify must begin with a dollar sign (\$) character and end with a semicolon (;) character. Inserting a period (.) character immediately before the value specifies that ePublisher use a period to separate the values when generating output.

<b>Value</b>	<b>Description</b>
<b>\$P;</b>	Includes the name of the project in the file name.
<b>\$T;</b>	Includes the name of the target in the file name.
<b>\$G;</b>	Includes the name of the group in Document Manager that contains the file name.
<b>\$C;</b>	Includes the project to project linking context value of the group in the file name. WebWorks Help and WebWorks Reverb use the context value when generating merged, or multivolume help that includes context-sensitive help. In WebWorks Help/Reverb, you need to include this context and the TopicAlias value in the help call to display the correct help topic. For more information, see “Merging Top-level Groups (Multivolume Help)” and “Opening Context-Sensitive Help in WebWorks Help using Standard URLs”.
<b>\$H;</b>	Includes the heading text or title of the topic in the file name.
<b>\$D;</b>	Includes the name of the source document that the topic came from in the file name.
<b>\$DN;</b>	Includes the source document number in the file name. The source document number is the number that identifies the position of the source document in the project.
<b>\$PN;</b>	Includes the page number in the file name. The page number is the number of the page that the topic is on in the source document.

You can also specify if you want ePublisher to convert spaces in file names to underscore (\_) characters when generating output. If you enable this setting, when you generate output, spaces in file names are replaced with the underscore character. For example, with this setting disabled, file names display as `Word1`



`Word2.html`. With this setting enabled, when you generate output file names display as `Word1_Word2.html`.

## To specify a page file naming patterns for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see "Working with Target Settings".
3. Under **Files**, specify the appropriate values for the page file naming pattern you want to use. For more information about file settings and values, click **Help**.

## Specifying File Names for Images Using Graphic Naming Patterns

ePublisher preserves the original file names for images imported by reference. If images are inserted directly into a source document, or if ePublisher cannot process the image by reference, then ePublisher assigns a file name using a graphic naming pattern.

By default ePublisher uses the following values when specifying file names for images:

**\$D;.\$DN;.\$PN;\$.GN**

This specifies that ePublisher name the files using the following syntax when it generates image files for a target:

*SourceDocumentName.SourceDocumentNumber.TopicNumber.ImageNumber*

The parts of the default naming rule are defined as follows:

### ***SourceDocumentName***

Identifies the name of the source document that the topic came from without the file extension.

### ***SourceDocumentNumber***

Identifies the number of the source document in the order it is included in its containing group in the project, such as 1 for the first source document in a group and 2 for the second source document in a group. This value starts at 1 for the first source document in each group in your project.

### ***TopicNumber***

Identifies the number of the topic (output page) generated from the source document, such as 1 for the first topic generated from a source document and

2 for the second topic generated from a source document. This value starts at 1 for the first topic in each source document.

### ***ImageNumber***

Identifies the number of the image in the topic generated from the source document, such as 1 for the first image generated in a topic and 2 for the second image generated in a topic. This value starts at 1 for the first image in each topic.

For example, if you have a source document named `MyFile.doc` that contains two images in the first topic that generates an output file, and three images in the second topic that generates an output file, and all the images are directly included in the source document, ePublisher generates the following default output image file names: `MyFile.1.1.1.jpg`, `MyFile.1.1.2.jpg`, `MyFile.1.2.1.jpg`, `MyFile.1.2.2.jpg`, and `MyFile.1.2.3.jpg`. You can change the default format and file extension for each graphic style.

You can use the following values to specify an image (graphic) file naming pattern for a target:

**Note:** Each value you specify must begin with a dollar sign (\$) character and end with a semicolon (;) character. Inserting a period (.) character immediately before the value specifies that ePublisher use a period to separate the values when generating output.

<b>Value</b>	<b>Description</b>
<b>\$P;</b>	Includes the name of the project in the file name.
<b>\$T;</b>	Includes the name of the target in the file name.
<b>\$G;</b>	Includes the name of the group in Document Manager that contains the file name.
<b>\$C;</b>	Includes the project to project linking context value of the group in the file name. WebWorks Help and WebWorks Reverb use the context value when generating merged, or multivolume help that includes context-sensitive help. In WebWorks Help/Reverb, you need to include this context and the TopicAlias value in the help call to display the correct help topic. For more information, see “Merging Top-level Groups (Multivolume Help)” and “Opening Context-Sensitive Help in WebWorks Help using Standard URLs”.
<b>\$H;</b>	Includes the heading text or title of the topic in the file name.
<b>\$D;</b>	Includes the name of the source document that the topic came from in the file name.
<b>\$DN;</b>	Includes the source document number in the file name. The source document number is the number that identifies the position of the source document in the project.
<b>\$PN;</b>	Includes the page number in the file name. The page number is the number of the page that the topic is on in the source document.
<b>\$GN;</b>	Includes the graphic number in the file name. The graphic number is the sequential number of the graphic

Value	Description
	in the generated output, and it is based on the position of the graphic in the generated output.

**You can also specify if you want ePublisher to convert spaces in file names to underscore (\_) characters when generating output. If you enable this setting, when you generate output, spaces in file names are replaced with the underscore character. For example, with this setting disabled, file names display as `Word1 Word2.jpg`. With this setting enabled, when you generate output file names display as `Word1_Word2.html`.**

## To specify an image (graphic) file naming pattern for a target

1. On the **Project** menu, select the target next to **Active Target** for which you want to specify settings.
2. On the **Target menu**, click **Target Settings**. You must have target modification permissions to modify target settings. For more information, see “Working with Target Settings”.
3. Under **Files**, specify the appropriate values for the graphic file naming pattern you want to use.

## Using Markers to Define File Names

You can use markers and field codes to specify the file name for a topic or image. Writers insert a Filename marker into their source documents and specify the file name they want to assign to the topic or image in the marker.

The default name for this marker type or field code is Filename. You can define your own marker style with a different name in your Stationery and assign the Filename marker type to it. Then, writers can use this marker type or field code to specify the output file name for each topic and included image.

**Note:** ePublisher uses page and graphic naming patterns to assign file names to all topics and images that do not include a Filename marker type or field code. For more information, see “Specifying File Names for Pages Using Page Naming Patterns” and “Specifying File Names for Images Using Graphic Naming Patterns”.

## To assign file name behavior to file name markers

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Marker Styles**, select the marker style you want to modify.
4. On the **Options** tab, set **Marker type** to **Filename**.

## Defining Context-Sensitive Help Links

Context-sensitive help links provide content based on the context of what the user is doing. In many cases, this help content is based on the window that is open and active. For example, the **Help** button on a window in a software product can open a specific help topic that provides important information about the window:

- What the window allows you to do
- Brief concepts needed to understand the window
- Guidance for how to use the window
- Descriptions about each field on the window, valid values, and related fields
- Links to related topics, such as concepts and tasks related to the window

The help topic can also be embedded in the window itself, such as an HTML pane that displays the content of the help topic. Providing this content when and where the user needs it, without requiring the user to search through the help, keeps the user productive and focused. This type of help also makes the product more intuitive by providing answers when and where needed.

There are several methods for creating context-sensitive help. In addition, output formats use different mechanisms to support context-sensitive help. You can reference a topic in the following ways:

### File name

Use a Filename marker to assign a file name to a topic. Each topic can have no more than one Filename marker by default. However, you can create a custom mapping mechanism using file names. Then, you can open the specific topic with that file name. However, if your file naming changes, you need to change the link to the topic. This file naming approach delivers context-sensitive help capabilities in output formats that do not provide a mapping mechanism.

### Internal identifier (topic alias)



Use a TopicAlias marker to define an internal identifier for each topic. The benefit of using an internal identifier is that it allows file names to change without impacting the links from the product. The writer inserts this marker in a topic and specifies a unique value for that topic. Then, the mapping mechanism of your output format determines how that internal identifier is supported. Some output formats, such as HTML Help, use a mapping file that defines these topic aliases. You can create more than one TopicAlias marker in a topic to allow multiple context-sensitive links to display the same topic.

To simplify the coding of your source documents, you can use the same marker to define both the file name and the topic alias for each topic file. In Style Designer, set the **Marker type** option for the marker you want to use to **Filename and topic alias**.

For more information about using markers to enable context-sensitive help links, see the following topics:

- “Defining Filename Markers for Context-Sensitive Help Links”
- “Defining Filename Markers for Context-Sensitive Help Links”

For more output format-specific information about using and customizing context-sensitive help, see the following topics:

- “Using Context-Sensitive Help in WebWorks Help”
- “Using Context-Sensitive Help in HTML Help”
- “Using Context-Sensitive Help in Oracle Help and Sun JavaHelp”
- “Using Context-Sensitive Help in WebWorks Reverb”
- “Using Context-Sensitive Help in WebWorks Reverb 2.0”

## Defining Filename Markers for Context-Sensitive Help Links

To enable context-sensitive help links using file names, you need to enable the Filename marker. By default, ePublisher sets the **Marker type** option for a marker named Filename to **Filename**. You can create a marker with a different name and set the **Marker type** option for that marker to **Filename**.

Then, writers can use this marker in the source documents to define a file name for each topic that will be opened by the application. File names must follow these guidelines:

- Must be unique

- Can only contain characters valid for file names

## To assign file name behavior to file name markers

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In **Marker Styles**, select the marker style you want to modify.
4. On the **Options** tab, set **Marker type** to **Filename**.

## Defining TopicAlias Markers for Context-Sensitive Help Links

To enable context-sensitive help links using topic IDs, you need to enable the TopicAlias marker. By default, ePublisher sets the **Marker type** option for a marker named TopicAlias to **Topic alias**. You can create a marker with a different name and set the **Marker type** option for that marker to **Topic alias**.

Then, writers can use this marker in the source documents to define a topic ID in each topic that will be opened by the application. Topic IDs must follow these guidelines:

- Must be unique
- Must begin with an alphabetical character
- May contain alphanumeric characters
- May not contain special characters or spaces, with the exception of underscores (  )

### **To assign topic alias behavior to topic alias markers**

- 1.** Open your Stationery design project.
- 2.** On the **View** menu, click **Style Designer**.
- 3.** In **Marker Styles**, select the marker style you want to modify.
- 4.** On the **Options** tab, set **Marker type** to **Topic alias**.

**To avoid duplicate topic Alias markers in Word, follow these steps:**

1. Always insert topic alias markers at the end of Word headings, never at the start or middle.
2. When you edit the headings, always display hidden text. Otherwise, you might inadvertently move a topic alias to the middle of the heading, causing problems.

## **Defining Expand/Collapse Sections (Drop-Down Hotspots)**

You can create sections of content that expand and collapse when you click a link or hot spot. This structure allows you to create items, such as bulleted lists, that are easy to scan, and then the users can expand individual items to get additional information. You can also use this structure to provide definitions.

When you identify the paragraph styles to start expand/collapse sections, you define whether those sections should initially be expanded (shown) or collapsed (hidden). ePublisher inserts an image indicating the state of the link. When a user clicks on a hotspot with the initial content collapsed, the content expands under the hotspot. If the user clicks a second time on the hotspot, the content is hidden again.

## **Using Styles and Markers for Expand/Collapse Sections**

To use expand/collapse sections, you need the paragraph styles you want to start these sections, and you need the DropDownEnd marker in your source documents. Decide which paragraph styles should provide the link and start an expand/collapse section. For example, you could make Heading 4, Procedure Title, and Bullet Expand paragraph styles start expand/collapse sections.

## To enable expand/collapse sections in your Stationery

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the paragraph style you want to define as the link for the start of an expand/collapse section.
5. On the **Options** tab, set **Dropdown** to **Start open** or **Start closed**, based on whether you want the expand/collapse section to be displayed or hidden by default.
6. On the **Options** tab, set **Dropdown toggle icon position** to **Left** or **Right**, based on whether you want the expand/collapse button to be on the left or right of your paragraphs.
7. Repeat steps 3-4 for each paragraph style you want to start expand/collapse sections.

When writers use these styles, they can identify the end of each expand/collapse section with the DropDownEnd marker. All other paragraph styles should have the Dropdown option set to Continue to be included in the expand/collapse sections as needed.

## Modifying Images for Expand/Collapse Sections

You can replace the default images ePublisher uses to indicate the state of an expanded or collapsed hotspot. For more information about override files and locations, see “Stationery, Projects, and Overrides”.

## To modify expand/collapse images

1. ***If you want to override the images for an output format***, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
  - b. Create the `Formats\formattype\Files\images` folder in your project folder, where *formattype* is the name of the output format to override, such as `Dynamic HTML`.
2. ***If you want to override the images for a target***, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Target Override Directory**.
  - b. Create the `Targets\targetname\Files\images` folder in your project folder, where *targetname* is the name of the target you want to override.
3. Paste the `.gif` files you want to use with names identical to those you want to replace in the `Images` folder you created. The following table lists the default images and their file names and sizes.

Image	Image Name	Image Size
	<code>expanded.gif</code>	Width: 8 pixels (0.111 inch) Height: 6 pixels (0.083 inch)
	<code>collapse.gif</code>	Width: 6 pixels (0.083 inch) Height: 8 pixels (0.111 inch)

**Note:** If you are generating with WebWorks Reverb output, you will modify the expand/collapse images in the skin.png file located in Pages\images in the Format or Target override directory for this output. For more information, See “Changing the Appearance of WebWorks Reverb”.

4. Regenerate your project to review the changes.

## Defining Popup Windows

You can use popup windows to display brief additional information about a word or phrase, such as a glossary definition for a term in a topic. A popup window displays a link in a topic, which indicates to users they can hover over or click on the link, which displays the additional content. Popup windows streamline the initial content and allow users to choose whether they want to view the additional content.

To create a popup window, writers first create a link in the original text to the content it should display. Writers create this link using the link features in their source document application. Then, the writers can implement the popup window using markers or paragraph styles.

You need to assign popup behavior options to your paragraph and marker styles in your Stationery.

## Using Marker Styles to Create Popup Windows

The default marker styles that define popup windows are Popup, PopupOnly, and PopupEnd. The writer inserts these markers into the source documents to specify what content to display through the popup window, and whether the popup content is displayed in only a popup window or in both a popup window and a clickable



link that displays the content in a different topic. Some output formats support displaying the content only in a popup window.

## **Using Paragraph Styles To Create Popup Windows**

You can define paragraph styles in your source documents to create popup windows. This approach avoids the need for markers, but you may need to create more styles in your source document templates. You can define both marker styles and paragraph styles to create different popup windows in your content.

## To define paragraph styles for popup windows

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the paragraph style you want to define as the first paragraph of a popup window.

This paragraph style is applied only to the first paragraph of content that should be displayed in a popup window. If a popup window may contain more than one paragraph of content, you need to create a second paragraph style and apply it to all paragraphs following the first paragraph that should be displayed in the popup window.

5. On the **Options** tab, select the appropriate value for the **Popup** option, such as **Define** or **Define with no output**. For more information about this option, click **Help**.
6. Select the paragraph style you want to define as one of the paragraphs of a popup window that follows the first paragraph of the popup window.
7. On the **Options** tab, select the appropriate value for the **Popup** option, such as **Append** or **Append with no output**. For more information about this option, click **Help**.

## Assigning a Page Style to Popup Windows

If you have popup windows in your content, the appearance of the popup windows matches the rest of your topic pages, including breadcrumbs and company information. If you want to assign a different page style to your popup windows, you need to create a new page style and assign it to your popup paragraph styles.

**Note:** If you are using marker styles to create popup windows, you cannot use page styles to control the appearance of your popup windows. This process applies only to popup windows created with paragraph styles.

## To assign a page style to popup windows

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. Create a new page style for popup windows by completing the following steps:
  - a. In **Page Styles**, click the **New Style** button. ePublisher adds a new page style called Untitled.
  - b. Enter the name for the new page style, such as `Popup Windows`.
  - c. On the **Properties** tab and the **Options** tab, select the options you want to assign to this page style.
5. In **Paragraph Styles**, select your popup paragraph style.
6. On the **Options** tab, set **Popup page style** to the page style you created for popup windows, such as `Popup Windows`.

## Defining Related Topics

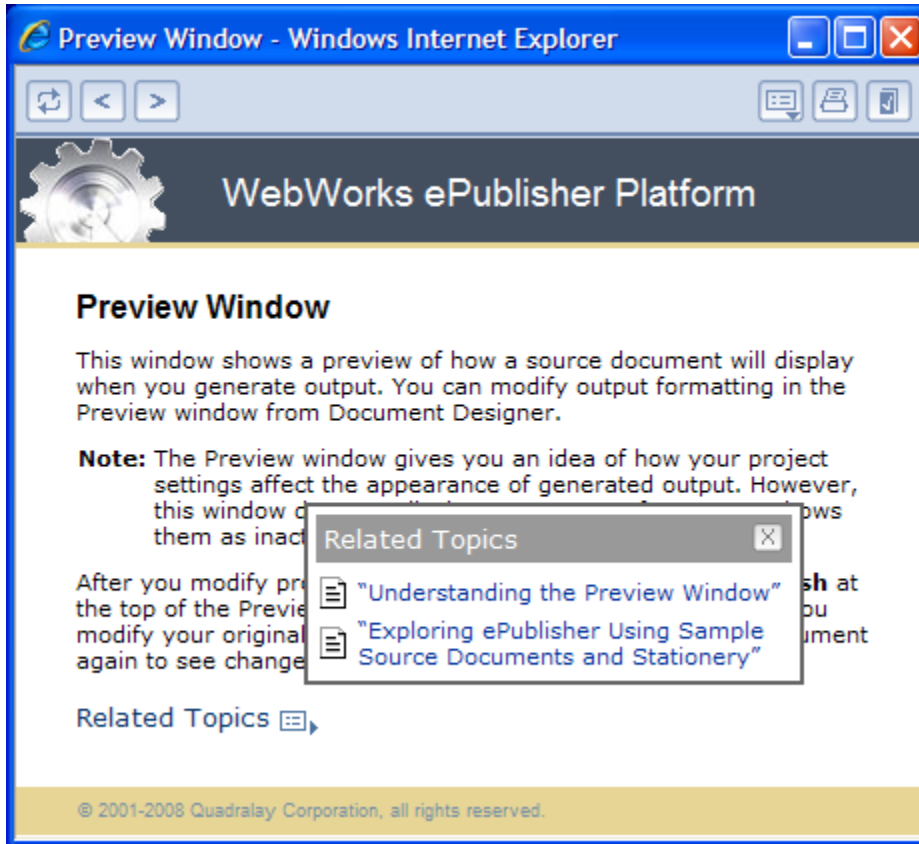
Related topics provide a list of other topics that may be of interest to the user of the current topic. For example, you could have a section called Creating Web Pages in your help. You may also have many other topics, such as HTML Tags and Cascading Style Sheets, that relate to creating Web pages. Identifying these related topics for users can help them find the information they need and identify additional topics to consider. However, providing these types of links as cross references within the content itself may not be the most efficient way to present the information.

Related topics and See Also links provide similar capabilities, but there are several important differences:

- Related topics can link to headings in a help set that do not start a new page.
- Related topics links are static and defined in the source documents as links. You must have all the source documents to create the link and generate the output.
- If a related topics list contains a broken link in the source document, that link is broken in the generated output. In a See Also link list, the broken link is not included in the output.

You can configure related topics to be displayed in the following ways:

- Included as a list in the topic itself.
- Displayed in a popup window when the user clicks a button, as shown in the following figure.



**Note:** If a related topic link is broken in the source document, in most cases that link is broken in the generated output. WebWorks Help and WebWorks Reverb provide an additional feature by removing broken links from related topics lists that are displayed in a popup window when a user clicks the Related Topics button.

## Using a Paragraph Style for Related Topics Lists

You can use a paragraph style to define a related topics list. Create a unique paragraph style to use specifically for the related topics list items. The writer should create a list of links in a topic in the source document, and apply the paragraph style to each list item.

## To define a paragraph style for a related topics list

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. In **Paragraph Styles**, select the paragraph style you want to define as the related topics list. This paragraph style is applied to all paragraphs in a related topics list.
5. On the **Options** tab, select the appropriate value for the **Related topic** option. For more information about this option, click **Help**.
  - To display the list of related topics in the body of the topic, select **Define**.
  - To display the list of related topics only when the **Related Topics** button is clicked, select **Define with no output**.

**Note:** The **Show related topics inline button** and the **Show related topics toolbar button** target settings specify whether to include related topics buttons and where to include them. If you select **Define** for your related topics paragraph style and you enable the **Show related topics inline button** setting, both the list of related topics and the related topics button itself are displayed in the topics with related topics.

## Defining See Also Links

See Also links are associative links (Alinks) that identify other topics that may be of interest to the user of the current topic. These links use internal identifiers to define the links and the list is built dynamically based on the topics available when the user displays the links. See Also links are important to use with larger help sets and merged help sets.

Related topics and See Also links provide similar capabilities, but there are several important differences:

- See Also links must link to styles that start a new topic, such as a heading.
- See Also links are dynamic and the lists of links are built at display time instead of during help generation.
- Since See Also link lists are dynamically built, they do not include links to topics that are not available when the user displays the links. If a related

topics list contains a broken link in the source document, that link is broken in the generated output for most output formats.

## Enabling See Also Functionality

If you want to create a See Also link as inline text without a button, create a unique character style and select the **See Also** option for that style. If you want to use a button to display the See Also links, create a unique paragraph style, select the **See Also** option for that style, and type the See Also text on that paragraph. The properties you select for the paragraph style in Style Designer affect the text of the **See Also** button. To modify the appearance of the button itself, see "Modifying the Appearance of the See Also Button". You also need to create a SeeAlsoLink and SeeAlsoKeyword marker.

HTML Help also supports the SeeAlsoLinkDisplayType and SeeAlsoLinkWindowType markers. These markers allow you to change how the See Also links are displayed in HTML Help. For example, you can display the links as a popup menu.

**Note:** If paragraph and marker styles are created in your source document after you create a project, scan the document in the project again for the changes to take effect.

## To enable See Also functionality in your Stationery

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. On the **Project** menu, click a target in the **Active Target** menu option that is an output format that supports this option.
4. ***If you want to use buttons for See Also links***, complete the following steps:
  - a. In **Paragraph Styles**, select the paragraph style to use for See Also links.
  - b. On the **Options** tab, set **See Also** to **Enabled**.
5. ***If you want to use inline text for See Also links***, complete the following steps:
  - a. In **Character Styles**, select the character style to use for See Also links.
  - b. On the **Options** tab, set **See Also** to **Enabled**.
6. In **Marker Styles**, select the SeeAlsoKeyword marker style.
7. On the **Options** tab, set **Marker type** to **See Also Keywords**.
8. In **Marker Styles**, select the SeeAlsoLink marker style.
9. On the **Options** tab, set **Marker type** to **See Also Link Keywords**.

## Modifying the Appearance of the See Also Button

The properties you select for the paragraph style in Style Designer affect the text of the **See Also** button. In some output formats, you can modify the color of the See Also button background and borders. You must separately modify each button border. You can also modify each of the See Also button colors.

**Note:** To change the color of the See Also button, you modify the `content.xsl` file. ***If you modify the content.xsl file***, you will be responsible for maintaining your customizations to the file as needed each time you update your Stationery to work with a new version of ePublisher.

For more information about override files and locations, see “Stationery, Projects, and Overrides”.

## To change the color of the See Also button

1. **If you want to override the processing for an output format**, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
  - b. Create the `Formats\formattype\Transforms` folder in your project folder, where *formattype* is the name of the output format you want to override, such as `WebWorks Help 5.0`.
2. **If you want to override the processing for a target**, complete the following steps:
  - a. In your Stationery design project, on the **View** menu, click **Target Override Directory**.
  - b. Create the `Targets\targetname\Transforms` folder in your project folder, where *targetname* is the name of the target you want to override.
3. Copy the `content.xml` file from the following folder to the override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\formattype\Transforms
```

4. Open the `content.xml` file you copied to your project override folder.
5. Find the following code section.

```
<html:table border="0" cellspacing="0" cellpadding="0"  
onclick="{ $VarCargo/wwalinks:ALink[1]/@onClick}" summary="">  
  <html:tr>  
    <html:td height="2" colspan="4" bgcolor="#FFFFFF"></html:td>  
    <html:td width="2" height="2" background="{ $Var_seertup}"></  
html:td>  
  </html:tr>  
  <html:tr>  
    <html:td width="2" height="2" bgcolor="#FFFFFF"></html:td>  
    <html:td height="2" colspan="3" bgcolor="#EEEEEE"></html:td>  
    <html:td width="2" height="2" background="{ $Var_seeright}"></  
html:td>  
  </html:tr>  
  ...
```

6. Modify the RGB color values in the `bgcolor` attributes within this table to adjust the colors of the margins that form parts of the **See Also** button.



7. Save the `content.xml` file.
8. Regenerate your project to review the changes.

## Define the Default Settings for Each Target

You can have one or more output formats in your Stationery. You can also define multiple targets in your Stationery. The Stationery Designer properties and options are shared across all targets and output formats. Some settings, such as the target settings, variable values, conditions, and cross-reference formats, are defined per target. Some targets and output formats also offer additional features and customizations.

For each target in your Stationery, define the following default settings in your Stationery. If a writer installs the support with ePublisher Express, that writer can modify these settings in each project based on the Stationery:

- Target settings, such as the company information and navigation on each page. To specify the target settings for a target, select the target, and then click **Target Settings** on the **Target** menu. For more information about a setting, click **Help**.
- Index settings. For more information, see “Defining the Default Index Settings”.
- Variable values. For more information, see “Defining the Default Processing of Variables”.
- Condition visibility. For more information, see “Defining the Default Processing of Conditions”.
- Cross Reference formats and rules. For more information, see “Defining the Default Processing of Cross References”.
- PDF integration. For more information, see “Defining Default PDF Generation Settings”.
- Output format-specific customizations and features. For more information, see “Customizing Stationery for Output Format-Specific Features”.
- Reporting options, such as accessibility reporting. For more information, see “Defining the Accessibility Report to Validate Content” and “Defining Other Reporting Options”.

## Defining the Default Index Settings

The index provides the user with a point-and-click resource for quickly navigating your online content. ePublisher generates the index by default for the available

formats, using the native indexing features of the source document tool used to create the printed index.

The groups and order of index entries in your online index are determined through index entries defined in the source document, the `locales.xml` file in the ePublisher installation folder, and the output format display environment. The `locales.xml` file also defines the text that identifies `See` and `See also` style entries in your index. For more information about customizing the index appearance, see “Modifying the Appearance of the Index in Dynamic HTML”.

Depending on your output format, you can specify the file name for the generated index whether to generate the index. With the power of many full-text search engines, you may choose not to include an index in your generated output.

**Note:** If you selected the WebWorks Help or WebWorks Reverb output format, you must generate an index with the given file name. These options are predefined and cannot be changed.

## To enable index generation in your online content

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. Set **Generate index** to **Enabled**.
4. *If you want to change your index file name and your output format supports it*, specify the new file name in **Index filename**.

## Defining the Default Processing of Variables

A variable provides a placeholder for standard terms and for names that may change. By defining variables in your source documents, you have global control over the values contained within those variables. For example, you can create a variable for the copyright date of your documentation. You can then use that variable as needed in your content. Each year when you need to update the copyright, you can change the variable value in a single location instead of using the search and replace method through your documents.

In your project, you can specify the value of any variable. When you change the value of a variable in your project, it changes the value only in your generated output. The variable value is not changed in your source documents. You can also use the value of the variable defined in your source documents. To use the value defined in a source document for a variable, select **Use Document Value** for that variable.

### To specify a variable value

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Variables**.
3. Find the variable you want to modify.
4. In the **Value** column, select its current definition and make the desired changes, or select **Use Document Value**.

## Defining the Default Processing of Conditions

In your source documents, you can define conditions and use them to show or hide parts of your content. ePublisher allows you to define the visibility for each condition in your project. The conditions that are available in your project come directly from your source documents. You can modify the value for any of your conditions, which affects how your conditional text is incorporated into your generated output. You can also select **passthrough** for a condition to insert the content directly into your output without being transformed and coded for your output format. This option allows you to directly add HTML coding or multimedia files to your output.

**Note:** You can also use PassThrough markers and the Pass Through paragraph style and character style options to insert content directionly into your output without being transformed and coded for your output.

### To modify the value of a condition

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Conditions**.
3. Find the condition you want to modify.
4. In the **Value** column, select the visibility option you want to use for that condition.

## Defining the Default Processing of Cross References

Cross-references help users navigate through your content. ePublisher automatically transforms cross-references to links in the generated output. However, you often want cross-references in your online content to use a different format than your printed content. For example, you usually include page numbers only in your printed content. ePublisher enables you to add, edit, and delete cross-reference formats for your online output.

**Note:** This option is available only for specific source document types and each defined cross reference rule applies only to the selected source document type.

## To define and manage cross-reference formats

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Cross Reference Rules**.
3. In **Document type**, select the source document type for which you want to define the cross-references.
4. **If you want to add a cross reference format**, complete the following steps:
  - a. Click **Add New Cross Reference**.
  - b. In **Name**, type the format or name of the cross reference format you want to define. For example, to remove the phrase `see page` followed by a page number from a Word source document, type: `see page {PAGEREF \h}`. You can find the correct syntax for the search pattern by inspecting your source documents for the values.
  - c. In **Replacement**, type the format or text you want to replace the format you specified in **Name**. To replace the format with nothing, leave the **Replacement** field blank.
  - d. Click **OK**.
5. **If you want to modify a defined cross-reference format**, complete the following steps:
  - a. Double-click an existing cross-reference format you want to modify.
  - b. In **Replacement**, type the format or text you want to replace the format you specified in **Name**. To replace the format with nothing, leave the **Replacement** field blank.
  - c. Click **OK**.
6. **If you want to change the order in which ePublisher processes the cross-reference formats**, select a cross reference rule in the list, and then use the arrow buttons to arrange the formats from top to bottom in the order you want ePublisher to process them:
7. **If you want to delete a defined cross-reference format**, select the cross reference rule in the list, and then click **Delete Cross Reference**:
8. Click **OK** to close the Cross Reference Rules window.

## Defining Default PDF Generation Settings

ePublisher provides PDF options that enable you to generate a PDF file for each source document or to generate a PDF file for each top-level group in a project. If you generate a PDF for each top-level group, ePublisher combines all the source documents within a single top-level group, and then generates a single PDF file for those documents.

You can deliver the PDF files separate from your help system, or in some output formats you can add a PDF button to your toolbar that displays the PDF files in the window. The PDF file displayed when the user clicks the PDF button depends on which PDF generation options you chose in your project. If you have selected to:

- Generate PDF files only for the top-level groups, the window displays the PDF file for the top-level group in which the currently viewed topic is located when the user clicks the PDF button.
- Generate PDF files only for each source document, the window displays the PDF file for the source document in which the currently viewed topic is located when the user clicks the PDF button.
- Generate PDF files for all source documents and for all top-level groups, the window displays the PDF file for the source document in which the currently viewed topic is located when the user clicks the PDF button.

## To specify the PDF file generation settings and enable the PDF toolbar button

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. *If you want to generate a PDF file for each source document*, set **Generate a PDF per document** to **Enabled**.
4. *If you want to generate a PDF file for each top-level group of source documents*, set **Generate a PDF per top level group** to **Enabled**.
5. Set **Show PDF button** to **Enabled**.
6. Click **OK**.

## Defining the Accessibility Report to Validate Content

Content that must be accessed by people with disabilities must conform to guidelines published by both the W3C and the United States government. These guidelines are intended to help authors produce accessible content. ePublisher helps you produce online content that conforms to the W3C Web Content Accessibility Guidelines 1.0 (WCAG), Section 508 of the U.S. Rehabilitation Act of 1998, and the Americans with Disabilities Act (ADA).

If you take certain steps in creating your source documents and setting up your project, your generated output is accessible through assistive technologies such as screen readers. When you generate your project, ePublisher can perform several checks to verify that you embedded information and conform to the accessibility standard in these areas:

- Alternate text for all images (ImageAltText marker)
- Alternate text for clickable regions in all image maps (ImageAreaAltText marker)
- Long descriptions for all images (ImageLongDescText and ImageLongDescByRef markers)
- Summaries for all tables (TableSummary marker)

**Note:** ePublisher does not check to ensure that you have provided expansion text for abbreviations or acronyms nor does it verify that you have included citation markers for quotations. You can use the AbbreviationTitle, AcronymTitle, and Citation markers to add this information to your content.



For more information about producing accessible content, and to check your content further for compliance, see the following Web sites:

- [www.w3c.org/TR/WCAG10-CORE-TECHS](http://www.w3c.org/TR/WCAG10-CORE-TECHS)
- [www.w3.org/WAI](http://www.w3.org/WAI)
- [www.w3.org/WAI/Policy/](http://www.w3.org/WAI/Policy/)

ePublisher does not perform accessibility validation by default. You must enable accessibility validation of the content. ePublisher validates that all images and image maps have alternate text, all images have long descriptions, and all tables have summaries. You can choose which accessibility validation checks you want ePublisher to run.

**Note:** If you disable any of the accessibility validation checks, you cannot consider your content to be accessible or Section 508 compliant.

## To define the accessibility report that validates online content for accessibility compliance

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. Set **Generate accessibility report** to **Enabled**.
4. *If you want to exclude the alternate text check for images*, disable **Validate accessibility image alternate tags**.
5. *If you want to exclude the alternate text check for image maps*, disable **Validate accessibility image map alternate tags**.
6. *If you want to exclude the long description check for images*, disable **Validate accessibility image long descriptions**.
7. *If you want to exclude the summary check for tables*, disable **Validate accessibility table summaries**.

When a project based on this Stationery is generated, the accessibility report is created with the information you selected to include in that report.

## Defining Other Reporting Options

In addition to the accessibility reporting features, ePublisher provides reports to help you identify and resolve potential transformation issues. You can define which conditions are informational messages, warnings, or errors. Errors force ePublisher AutoMap to return a non-zero return code and stops the output deployment process.

### To define the other report options

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. Review and define the settings for the following categories of report settings:
  - **Filenames Report**
  - **Links Report**
  - **Styles Report**
  - **Topics Report**
4. Click **OK**.

## Saving and Testing Stationery

Once you have defined your Stationery design project, you need to save and test the Stationery. This process allows you to adjust the Stationery as needed before further customizing your design and deploying the Stationery for use.

**Note:** Document-specific information, such as groups, documents, and changes made with Document Designer, are not saved in Stationery files.

When you save the Stationery design project as Stationery, ePublisher stores all the style information, settings, and definitions from the project in the Stationery. ePublisher also copies the user, output format, and format target override files, and saves them as part of your Stationery.

## To save your Stationery design project as Stationery and test it

1. Open your Stationery design project.
2. On the **File** menu, click **Save as Stationery**.
3. Specify the Stationery name, location to store the Stationery, and the targets to include in the Stationery. For more information about an option, click **Help**.
4. Click **OK**.
5. Close ePublisher Designer.
6. Open ePublisher Express.
7. On the **File** menu, click **New Project**.
8. Specify the project name and the location to store the project.
9. In **Stationery**, specify your Stationery file, and then click **Next**.
10. Add your standard sample source document to the project, and then click **Finish**. For more information about an option, click **Help**.
11. On the **Project** menu, click **Scan All Documents**.
12. On the **Project** menu, select the active target you want to test.
13. On the **Project** menu, click **Generate All**.
14. Review the generated output.
15. Repeat steps 12-14 for each target in your project.

Once you finish customizing your Stationery, store it in a central location where writers can use it. If you move the Stationery, ePublisher Express notifies the writers when they open their projects that it cannot find the Stationery associated with the project. The writers then need to update their projects to use the Stationery in its new location.

## Backing Up Your Stationery Design Project, Stationery, and Projects

You should save a backup copy of your Stationery design project, your Stationery, and any individual projects you have. These snapshots can help you revert your project to a specific stage at a point in time. This snapshot can also help you if your project or Stationery become corrupted or lost.

Element to Back Up	Description of Process and Considerations
Stationery design project	Copy all your sample files, the <code>.wep</code> file, and the <code>Files</code> , <code>Formats</code> , and <code>Targets</code> folders and subfolders. Maintain the same structure of these files and folders.
Stationery	You can create a <code>.zip</code> file of your Stationery to back it up. If needed, you can also recreate your Stationery from your Stationery design project. Copy the <code>.wxsp</code> file, the <code>.manifest</code> file, and the <code>Files</code> , <code>Formats</code> , and <code>Targets</code> folders and subfolders. Maintain the same structure of these files and folders.
Individual project	To recreate your generated output, you need the source documents, the Stationery you used, and your individual project file. Copy your source documents and the <code>.wrp</code> file. Maintain the same structure of your source files and the <code>.wrp</code> file.

## Deploying Stationery

Once you have saved, tested, and finished customizing your Stationery, you need to deploy it so writers can use the Stationery for their projects. Review the following considerations when deploying your Stationery for writers to use in their projects:

- Store your Stationery in a central location where writers have read access.
- When you save the Stationery design project as Stationery, ePublisher stores all the style information, settings, and definitions from the project in the Stationery. ePublisher also copies the user, output format, and format target override files, and saves them as part of your Stationery.
- If you move the Stationery, ePublisher Express notifies the writers when they open their projects that it cannot find the Stationery associated with the project. The writers then need to update their projects to use the Stationery in its new location.
- Make sure your source document templates and standards support your Stationery. Keep these files updated as a unit. If you add a style to your Word or FrameMaker template, also add it to your Stationery design project and update your Stationery.

- If the source documents for an existing project use only the styles in your standard templates, writers can use ePublisher Express to synchronize their project with your Stationery. This process allows you to update the project using the current Stationery for the project, or you can select a different Stationery to associate with the project.
- Schedule a training session with the team about each of the features they can use in their help, how to use styles and markers to define those features, and how to use ePublisher Express and their projects to achieve the results they need.
- Make sure your Stationery has the output formats and targets you need. Since all output formats in the Stationery share Style Designer properties and options, if you want different output formats to have different behaviors or appearance, you may need to create and maintain more than one Stationery.
- Determine whether writers should change target-specific settings, such as variables, conditions, cross reference rules, and target settings in their projects. When writers install ePublisher Express, they can specify whether to enable these menu options.

## To save your Stationery design project as Stationery

1. Open your Stationery design project.
2. On the **File** menu, click **Save as Stationery**.
3. Specify the Stationery name, location to store the Stationery, and the targets to include in the Stationery. For more information about an option, click **Help**.
4. Click **OK**.

## Managing and Updating Stationery

Once you deploy your Stationery, manage its use to make sure it continues to meet your needs. Your source document templates and standards can change over time. Make sure your source document templates support your Stationery and keep these files updated as a unit. If you add a style to your Word or FrameMaker template, also add it to your Stationery design project and update your Stationery.

You may also decide to add a feature to your output, such as expand/collapse sections or popup windows. To add a feature, you may need to make changes to both your source document templates and your Stationery. Then, you need to put together a deployment or roll-out plan to help writers decide when and how each project should use the features.

If you move or change the Stationery, ePublisher Express notifies the writers when they open their projects. The writers have the opportunity to synchronize their projects with the Stationery and bring their projects inline with your new standards. For more information, see "Synchronizing Projects with Stationery".

Be careful when you update your Stationery to make sure you have the files you need. Review the following considerations for properly maintaining your Stationery:

- Store your Stationery design project and supporting files in a version control system. This process allows you to monitor how it changes over time and ensures you can return to a previous version, if needed.
- Create a subfolder in your Stationery design project and store a sample of each source document type in that folder. These files help you test and verify the output as you modify your source document styles and your ePublisher styles and settings.
- Do not directly open and modify the Stationery files. To make sure your Stationery is properly updated, always open the Stationery design project for the Stationery, make your changes, and then save a new copy of the Stationery.

- Consider saving your updated project as Stationery to a new location and have several writers test some smaller projects with the updated Stationery before you update your Stationery for all projects.

When you save your Stationery, ePublisher creates the following folders that contain information about any customizations or overrides you created when you developed the Stationery:

- *StationeryName*\Formats\OutputFormat
- *StationeryName*\Formats\OutputFormat.base

where *StationeryName* is the name you specified for the Stationery, and *OutputFormat* is the type of output format you specified for a target in the Stationery. You can use these folders to help you identify any customizations or overrides you specified for your Stationery when updating your Stationery.

The *StationeryName*\Formats\OutputFormat folder contains any customizations or overrides you specified when designing the Stationery. ePublisher Express synchronizes with the files in the *OutputFormat* folder and uses the information about customizations and overrides contained in files in the *OutputFormat* folder to generate output.

**Note:** The Stationery may have one or more *OutputFormat* folders, based on the settings you specified in your Stationery.

The *StationeryName*\Formats\OutputFormat.base folder contains copies of all the files located in the \Program Files\WebWorks\ePublisher\2024.1\Formats\OutputFormat folder. These files define the default output format and transforms and are installed by default when you install ePublisher.

You can do a compare, or **diff**, between the files located in these folders to quickly see any customizations or overrides specified for the Stationery. You can use this information to help you reapply customizations and overrides as needed when designing a newer version of the Stationery in ePublisher Designer.



## To update your Stationery

1. Open your Stationery design project.
2. Make the desired changes.
3. On the **File** menu, click **Save as Stationery**.
4. Specify the Stationery name, location to store the Stationery, and the targets to include in the Stationery. To replace the existing Stationery, specify the same name and values as the existing Stationery. You do not have to replace the existing Stationery. You can also create a new Stationery with a different name, such as by adding a version number to the Stationery name.
5. Click **OK**.

# Target Settings Reference

[Accessibility Settings](#)  
[Accessibility Report Settings](#)  
[Analytics Settings](#)  
[Baggage Files Settings](#)  
[Baggage Files Report Settings](#)  
[Company Information Settings](#)  
[Conditions Report Settings](#)  
[Cover Settings \(eBook - ePUB 2.0\)](#)  
[Eclipse Settings](#)  
[ePUB Settings \(eBook - ePUB 2.0\)](#)  
[File Processing Settings](#)  
[Filenames Report Settings](#)  
[Files Settings](#)  
[Footer Settings](#)  
[Header Settings](#)  
[HTML Help Settings](#)  
[Images Report Settings](#)  
[Index Settings](#)  
[JavaHelp Settings](#)  
[Links Settings](#)  
[Links Report Settings](#)  
[Locale Settings](#)  
[Menu Settings](#)  
[Oracle Help Settings](#)  
[Page Settings](#)  
[PDF Settings](#)  
[Result Options Settings \(PDF - XSL-FO\)](#)  
[Search Settings](#)  
[Social Settings](#)  
[Styles Settings \(PDF - XSL-FO\)](#)  
[Styles Report Settings](#)  
[Table of Contents Settings](#)  
[Title Page Settings \(PDF - XSL-FO\)](#)  
[Toolbar Settings](#)  
[Topics Report Settings](#)  
[WebWorks Help Settings](#)  
[WebWorks Reverb Settings](#)  
[WebWorks Reverb 2.0 Settings](#)

In ePublisher, each output format has a set of configurable target settings that allow features of that format to be enabled, disabled, or specified from a fixed set of values or even user provided values. Some settings are specific to one or more output formats. The following are all of the possible target settings that an output format may have.

Use this section to find and learn more about target settings available in your format. The **Target Settings** window for a particular output format will list only settings that are available for that format.

## Accessibility Settings

These settings are not supported in PDF output formats. The settings in this category are defined as follows:

### **Add link to skip navigation content**

Specifies whether you want screen readers to skip over top-level navigation content on a page, such as breadcrumbs, previous and next buttons, and company information. If you enable this setting, when a screen reader reads the content on the page, the screen reader skips over the top-level navigation content and begins reading the page where the page content begins. This setting is supported in Dynamic HTML, Eclipse Help, Microsoft HTML Help, Oracle Help, and XML + XSL, output formats.

### **Include visible [D] links to images with long descriptions**

Specifies whether you want to include a link labeled [D] beside any image for which you provide a long description. When users click the [D] link, the link takes the user to a long description of the text. Because the [D] link is a standard hyperlink, it can be processed correctly by browsers and screen readers. This setting is supported in Microsoft HTML Help, WebWorks Help, Oracle Help, Dynamic HTML, and XML+XSL output formats.

## **Accessibility Report Settings**

The settings in this category are defined as follows:

### **Generate accessibility report**

Specifies whether to generate Accessibility reports.

### **Image links without alternative text**

Specifies the notification to receive if any image maps do not have alternative text.

### **Images without alternative text**

Specifies the notification to receive if an image does not have alternate text.

### **Images without long descriptions**

Specifies the notification to receive if an image does not have a long description.

### **Tables without summaries**

Specifies the notification to receive if a table does not have a summary.

# Analytics Settings

These settings apply only to the WebWorks Reverb output format. This allows for better tracking of how users are using your help system via Google Analytics. For more information on how Google Analytics is used please refer to the following webpage: <http://www.google.com/support/googleanalytics/>.

## Google Default URL

If a default URL is specified, analytics events will only be captured when the online help files are hosted from this specified URL. Specifying this setting is useful to prevent analytic events from being logged during testing from the file system or a test web server.

## Google Tracking ID

Enables the use of Google Analytics to measure page activity. Requires that your end-users have a connection to the Internet so that Google's tracking urchin can be loaded within their browser when a page is viewed or an event is captured. If this value is empty, then no analytics tracking will be enabled.

## Page "Was This Helpful?" Buttons

The **Page Was This Helpful Buttons** provide your end-users with the ability to provide feedback using analytics events to anonymously record their experience with the current page being viewed. The name of the page will be recorded along with the end-user's feedback about the page's usefulness. A Google Tracking ID is required for these buttons to be in operation.

## Search "Was This Helpful?" Buttons

The **Search Was This Helpful Buttons** provide your end-users with the ability to provide feedback using analytics events to anonymously record their experience with the current search results from a given search query. The search term(s) will be recorded along with the end-user's feedback about the search results' usefulness. A Google Tracking ID is required for these buttons to be in operation.

# Baggage Files Settings

These settings are only supported in the WebWorks Reverb output format. The settings in this category are defined as follows:

## Baggage files info list

Specifies the path (absolute or relative to the project) to the baggage files info list containing information related to the baggage files, like title and summary, as well as determining when not to index a baggage file.

### **Copy baggage file dependents**

Specifies whether you want to include in your output all the dependents of your HTML baggage files, such as images, CSS files, JS files, videos and audios.

### **Index baggage files**

Specifies whether you want to index baggage files linked to by your source documents.

### **Index external links**

Specifies whether you want to index external links and include those words in the corresponding index file.

## **Baggage Files Report Settings**

The settings in this category are defined as follows:

### **Baggage files without summary**

Specifies the notification to receive when baggage files without a summary are found.

### **Baggage files without title**

Specifies the notification to receive when baggage files without a title are found.

### **Generate baggage files report**

Specify whether to generate baggage files report.

## **Company Information Settings**

The settings in this category are defined as follows:

### **Company copyright**

Specifies company copyright text.

### **Company email address**

Specifies the email address to display for the company in your output. By default, ePublisher displays this email address as a link using the `mailto` command.

### **Company fax number**

Specifies the fax number to display for the company in your output.

### **Company logo image**

Specifies the image to display for the company logo in your output. Store the image file in the `Files` folder within your project. If you want to use a company logo image in your generated output, press F12 to open the `Files` folder and verify that the folder contains the company logo image you want to use in your generated output.

### **Company name**

Specifies the company name to display in your output. If you specify a company URL in the **Company web page** setting, ePublisher displays the company name as a link to the specified URL.

### **Company phone number**

Specifies the company phone number to display in your output.

### **Company web page**

Specifies the company web page URL address. Specify the complete URL, including the prefix, such as <http://www.webworks.com>. If you specify this value, ePublisher displays the company name specified in the **Company name** setting as a link to the specified URL.

## **Conditions Report Settings**

The settings in this category are defined as follows:

### **Generate conditions report**

Specify whether to generate Conditions report.

### **Unknown conditions**

Specifies the notification to receive if a condition is encountered that was not defined in the project being generated.

## **Cover Settings (eBook - ePUB 2.0)**

These settings apply only to the eBook - ePUB 2.0 output format. The settings in this category are defined as follows:

### Cover page file name

Specifies the name of the file that ePublisher generates for the eBook cover page.

### Cover page image

Specifies the image to display on your cover page in your output. Store the image file in the `Files` folder within your project. If you want to use an image on your cover page in the generated output, press F12 to open the `Files` folder and verify that the folder contains the cover page image you want to use in your generated output.

### Cover page style

Specifies the ePublisher page style to use for generating the cover page.

## Eclipse Settings

These settings apply only to the Eclipse Help output format. The settings in this category are defined as follows:

### Eclipse Help ID prefix

Specifies the prefix for ePublisher to use when creating the Eclipse Help `.jar` file. By default, this value is set to `com.webworks.eclipsehelp`, and by default when ePublisher generates Eclipse Help, ePublisher names the Eclipse Help `.jar` file `com.webworks.eclipsehelp.ProjectName`, where *ProjectName* is the name of your ePublisher project.

### Eclipse Help vendor

Specifies the setting to use for the Vendor parameter in the Eclipse Help Content Plugin manifest file. By default, this value is set to **WEBWORKS**.

### JDK location

Specifies the location of the Java Development Kit (JDK) used by ePublisher when generating Eclipse Help system. By default, ePublisher auto-detects and uses the most recent version of the JDK on the computer where you generate output. Use this field if you have a specific version of the JDK that you want ePublisher to use when generating Eclipse Help for the specified target.

### Manifest bundle version

Specifies the version identifier for the manifest. Defaults to 1.0.0, but can be changed to between updates of your Eclipse Help deliveries.

## ePUB Settings (eBook - ePUB 2.0)

These settings apply only to the eBook - ePUB 2.0 output format. The settings in this category are defined as follows:

### Author name

Specifies author name metadata value of the generated eBook.

### Author name (file as)

Specifies an alternate value for the author name that is used when sorting multiple eBooks displayed together.

## File Processing Settings

File processing settings define how ePublisher processes front matter, index, and table of contents files. If you use ePublisher to generate output from Microsoft Word source documents, these settings are related to the RD field behavior settings you configure for the project on the Input Configurations tab on the Project Settings window. ePublisher does not use these settings when generating output from DITA-XML source documents.

The settings in this category are defined as follows:

### File processing behavior for front matter

Specifies whether you want to generate output for front matter included in your source documents. Front matter is all of the content before the table of contents. For example, in Adobe FrameMaker this setting allows you to use an Adobe FrameMaker `.book` file to generate output without generating output for the front matter included in the Adobe FrameMaker `.book` file.

### File processing behavior for index files

Specifies whether you want to generate output for index files included in your source documents. For example, in Adobe FrameMaker this setting allows you to use an Adobe FrameMaker `.book` file to generate output without generating output for a generated index (IX) `.fm` file included in the Adobe FrameMaker `.book` file.

### File processing behavior for TOC files



Specifies whether you want to generate output for table of contents files included in your source documents. For example, in Adobe FrameMaker this setting allows you to use an Adobe FrameMaker `.book` file to generate output without generating output for a generated table of contents (TOC) `.fm` file included in the Adobe FrameMaker `.book` file.

### **Insert Mark of the Web (MOTW)**

Enables the MOTW in the HTML content, used to get rid of the "blocked content message in Internet Explorer.

### **Pretty Print**

Enables the generation of HTML that is more easily human readable by inserting appropriate white space and end-of-line characters. By default this setting is disabled to produce smaller file sizes. Changing this setting should have no noticeable affect on the appearance of your content when rendered within an HTML browser.

## **Filename Report Settings**

The settings in this category are defined as follows:

### **File named by marker**

Specifies the notification to receive if ePublisher created a file using a Filename marker.

### **Filename marker collision with existing filenames**

Specifies the notification to receive if ePublisher detects duplicate Filename markers in the source documents used by your project to generate output.

### **Filename marker ignored**

Specifies the notification to receive if ePublisher ignored a file name in your source document.

### **Generate filenames report**

Specifies whether to generate Filenames reports.

## **Files Settings**

The settings in this category are defined as follows:

### **Convert name to**

Specifies the file naming case you want ePublisher to use when generating output files.

By default, ePublisher uses the **Normal** value when specifying file naming case.

The values for this setting are defined as follows:

Value	Description
<b>Normal</b>	Keeps the naming with the created case.
<b>Lower case</b>	Converts the names to lower case.
<b>Upper case</b>	Converts the names to UPPER CASE.
<b>Camel case</b>	Converts the names to camel Case.
<b>Pascal case</b>	Converts the names to Pascal Case.

For example, with this setting set to **Normal**, file names display as `Great publishing tool.html`. With this setting set to **Camel case**, when you generate output file names display as `great Publishing Tool.html`. With this setting set to **Pascal case**, when you generate output file names display as `Great Publishing Tool.html`. With this setting set to **Upper case**, when you generate output file names display as `GREAT PUBLISHING TOOL.html`. With this setting set to **Lower case**, when you generate output file names display as `great publishing tool.html`.

## Graphic naming pattern

Specifies the file naming pattern you want ePublisher to use when generating output files for images.

ePublisher preserves the original file names for images imported by reference. If images are inserted directly into a source document, or if ePublisher cannot process the image by reference, then ePublisher assigns a file name using a graphic naming pattern.

By default, ePublisher uses the following values when specifying the image naming pattern:

**\$D;.\$DN;.\$PN;\$.GN**

This specifies that ePublisher name the files using the following syntax when it generates output:

*SourceDocumentName.SourceDocumentNumber.TopicNumber.ImageNumber*

where *SourceDocumentName* is the name of the source document, *SourceDocumentNumber* is the number that identifies the position of the source document in the project, *TopicNumber* is the number of the topic (output page) in the source document, and *ImageNumber* is the number of the image in the source document.

The values for this setting are defined as follows:

**Note:** Each value you specify must begin with a dollar sign (\$) character and end with a semicolon (;) character. Inserting a period (.) character immediately before the value specifies that ePublisher use a period to separate the values when generating output.

<b>Value</b>	<b>Description</b>
<b>\$P;</b>	Includes the name of the project in the file name.
<b>\$T;</b>	Includes the name of the target in the file name.
<b>\$G;</b>	Includes the name of the group in Document Manager that contains the file name.
<b>\$C;</b>	Includes the project to project linking context value of the group in the file name. WebWorks Help uses the context value when generating merged, or multivolume help that includes context-sensitive help. In WebWorks Help, you need to include this context and the TopicAlias value in the help call to display the correct help topic. For more information, see "Merging Top-level Groups (Multivolume Help)" and "Opening Context-Sensitive Help in WebWorks Help using Standard URLs".
<b>\$H;</b>	Includes the heading text or title of the topic in the file name.
<b>\$D;</b>	Includes the name of the source document that the topic came from in the file name.
<b>\$DN;</b>	Includes the source document number in the file name. The source document number is the number that identifies the position of the source document in the project.
<b>\$PN;</b>	Includes the page number in the file name. The page number is the number of the page that the topic is on in the source document.
<b>\$GN;</b>	Includes the graphic number in the file name. The graphic number is the sequential number of the graphic in the generated output and it is based on the position of the graphic in the generated output. For example, if you have five images in the generated output and you

Value	Description
	use this setting, the page with you first image has the number 1 in the file name, and the page with your fifth image has the number 5 in the file name.

## Page break handling

Specifies how ePublisher processes the **Page break priority** value the Stationery designer specifies in Style Designer for each paragraph style. For example, if you set the **Page break handling** format setting to **Always**, all paragraph styles with the **Page break priority** set greater than 0 will create a new page. The default value is **Combine**. The values for this setting are defined as follows:

Value	Description
<b>Never</b>	Ignores all <b>Page break priority</b> values specified in the Stationery.
<b>Always</b>	Creates a new page for all paragraph styles with a <b>Page break priority</b> value greater than 0.
<b>Combine</b>	Creates a new page for all paragraph styles with a <b>Page break priority</b> value greater than 0 unless the previous paragraph created a new page and the <b>Page break priority</b> value for the previous paragraph is a number less than the <b>Page break priority</b> value for this paragraph.
<b>If not previous</b>	Creates a new page for all paragraph styles with a <b>Page break priority</b> value greater than 0 unless the previous paragraph created a new page.

## Page naming pattern

Specifies the file naming pattern you want ePublisher to use when generating output pages for topics. By default, ePublisher uses the following values when specifying the page naming pattern:

**\$D;.\$DN;.\$PN**

This specifies that ePublisher name the files using the following syntax when it generates output:

*SourceDocumentName.SourceDocumentNumber.TopicNumber*

where *SourceDocumentName* is the name of the source document, *SourceDocumentNumber* is the number that identifies the position of the source document in the project, and *TopicNumber* is the number of the topic (output page) in the source document.

The values for this setting are defined as follows:

**Note:** Each value you specify must begin with a dollar sign (\$) character and end with a semicolon (;) character. Inserting a period (.) character immediately before the value specifies that ePublisher use a period to separate the values when generating output.





<b>Value</b>	<b>Description</b>
<b>\$P;</b>	Includes the name of the project in the file name.
<b>\$T;</b>	Includes the name of the target in the file name.
<b>\$G;</b>	Includes the name of the group in Document Manager that contains the file name.
<b>\$C;</b>	Includes the project to project linking context value of the group in the file name. WebWorks Help uses the context value when generating merged, or multivolume help that includes context-sensitive help. In WebWorks Help, you need to include this context and the TopicAlias value in the help call to display the correct help topic. For more information, see "Merging Top-level Groups (Multivolume Help)" and "Opening Context-Sensitive Help in WebWorks Help using Standard URLs".
<b>\$H;</b>	Includes the heading text or title of the topic in the file name.
<b>\$D;</b>	Includes the name of the source document that the topic came from in the file name.
<b>\$DN;</b>	Includes the source document number in the file name. The source document number is the number that identifies the position of the source document in the project.
<b>\$PN;</b>	Includes the page number in the file name. The page number is the number of the page that the topic is on in the source document.

## Replace spaces

Specifies if you want ePublisher to replace spaces in file names with the string defined in the setting **Replace spaces with** when generating output. For example, with this setting **Disabled**, file names with spaces

would generate to `Word1 Word2.html`. With this setting **Enabled**, and the **Replace spaces with** setting set to value `_`, the file name would generate to `Word1_Word2.html`.

## Replace spaces in group names

Specifies if you want ePublisher to replace spaces in group derived file paths with the value defined in the setting **Replace spaces with**. For example, with this setting **Disabled**, group derived file names with spaces would generate to `My Group Name.html`. With this setting **Enabled**, and the **Replace spaces with** setting set to value `_`, the file name would generate to `My_Group_Name.html`.

## Replace spaces with

Specifies the value ePublisher will use to replace spaces in generated file names. There are predetermined values you can use, but you really can use any string allowed by the file system for a file name. For example, with this setting set to **No space**, and if the file name was originally `Word1 Word2.html`, then the new name would generate to `Word1Word2.html`.

By default, ePublisher uses the underscore (`_`) value for this setting.

The predetermined values for this setting are defined as follows:

Value	Description
—	Replaces spaces with the underscore character (default).
-	Replaces spaces with the hyphen character.
No space	Replaces spaces with the empty character (removes spaces).
Ignore	Ignores this setting.

**Note:** These values will only be used if **Replace spaces** and/or **Replace spaces in group names** are **Enabled**.

### Table naming pattern

Specifies the file naming pattern you want ePublisher to use when generating output pages that contain tables in MoinMoin. By default, ePublisher uses the following values when specifying the file naming pattern for MoinMoin pages that contain tables:

**\$D;.\$DN;.\$PN;.TN**

This specifies that ePublisher name the files that contain tables using the following syntax when it generates output:

*SourceDocumentName.SourceDocumentNumber.TopicNumber.TableNumber*

where *SourceDocumentName* is the name of the source document, *SourceDocumentNumber* is the number that identifies the position of the source document in the project, *TopicNumber* is the number of the topic (output page) in the source document, and *TableNumber* is the number of the table in the source document.

**Note:** Inserting a period (.) character immediately before the value specifies that ePublisher use a period to separate the values when generating output.

The values for this setting are defined as follows:

**Note:** Separate values with a semicolon (;) character.

<b>Value</b>	<b>Description</b>
<b>\$P;</b>	Includes the name of the project in the file name.
<b>\$T;</b>	Includes the name of the target in the file name.
<b>\$G;</b>	Includes the name of the group in Document Manager that contains the file name.
<b>\$C;</b>	Includes the project to project linking context value of the group in the file name. WebWorks Help uses the context value when generating merged, or multivolume help that includes context-sensitive help. In WebWorks Help, you need to include this context and the TopicAlias value in the help call to display the correct help topic. For more information, see "Merging Top-level Groups (Multivolume Help)" and "Opening Context-Sensitive Help in WebWorks Help using Standard URLs".
<b>\$H;</b>	Includes the heading text or title of the topic in the file name.
<b>\$D;</b>	Includes the name of the source document that the topic came from in the file name.
<b>\$DN;</b>	Includes the source document number in the file name. The source document number is the number that identifies the position of the source document in the project.
<b>\$PN;</b>	Includes the page number in the file name. The page number is the number of the page that the topic is on in the source document.
<b>\$TN;</b>	Includes the table number in the file name. The table number is the sequential number of the table in the generated output and it is based on the position of the table in the generated output. For example, if you have five tables in the generated output and you use this

Value	Description
	setting, the page with your first table has the number 1 in the file name, and the page with your fifth table has the number 5 in the file name.

## Footer Settings

Allows a footer to be displayed across the bottom of the help system. Typically this area is reserved for company information and logos. Some footer settings are provided that work with the **Company Information** settings and are dependent on them being set.

The settings in this category are defined as follows:

### Footer location

Specifies footer location. Choices range from displaying the footer across the entire bottom of the help system to just along the bottom of the content page.

### Footer logo

Specifies a footer logo to be displayed. This logo gets its value from the **Company name** or **Company logo image**.

### Footer logo link address

Specifies footer logo link address. Link address can inherit from the home page (splash page or first page in help set) or the **Company webpage**. In order for this setting to be displayed in the output, the target setting: **Linked footer logo** must be enabled.

### Footer logo override

Specifies which image to use as the footer logo. This setting will override the **Footer logo** setting. Choose from the image files stored in the `Files` folder within your project.

### Generate footer

Specifies whether to generate footer.

### Linked footer logo

Enables the **Footer logo link address** to be displayed.

# Header Settings

Allows a header to be displayed across the top of the web page above the Reverb toolbar. Typically this area is reserved for company information and logos. Some header settings are provided that work with the **Company Information** settings and are dependent on them being set.

The settings in this category are defined as follows:

## Generate header

Specifies whether to generate header.

## Header logo

Specifies a header logo to be displayed. The logo gets its value from the **Company name** or **Company logo image** settings.

## Header logo link address

Specifies header logo link address. Link address can inherit from the home page (splash page or first page in help set) or the **Company webpage**. In order for this setting to be displayed in the output, the target setting: **Linked header logo** must be enabled.

## Header logo override

Specifies which image to use as the header logo. This setting will override the **Header logo** setting. Choose from the image files stored in the `Files` folder within your project.

## Linked header logo

Enables the **Header logo link address** to be displayed

# HTML Help Settings

These settings apply only to the Microsoft HTML Help output format. The settings in this category are defined as follows:

## HTML Help binary index

Specifies whether you want to generate a binary index for Microsoft HTML Help output.

## HTML Help binary TOC

Specifies whether you want to generate a binary table of contents for Microsoft HTML Help output.

### **HTML Help custom map file**

Specifies the location of a custom Microsoft HTML Help map file if you want to use a custom HTML Help map file to generate Microsoft HTML Help output.

## **Images Report Settings**

The settings in this category are defined as follows:

### **Generate images report**

Specifies whether to generate images reports

### **Images in table cells**

Specifies the notification to receive if an image appears in a table cell.

### **Missing by-reference source files**

Specifies the notification to receive if an image is referenced but not found in the location where it is referenced.

## **Index Settings**

These settings are not supported in the PDF output format. The settings in this category are defined as follows:

### **Generate index**

Specifies whether you want to generate an index when you generate output.

## **JavaHelp Settings**

These settings apply only to the Sun JavaHelp output format. The settings in this category are defined as follows:

### **Category closed image**

Specifies the category closed image you want to use in the table of contents when generating Sun JavaHelp. This setting applies only to Sun JavaHelp 2.0. By default, Sun JavaHelp uses a closed folder image for the category closed image in the table of contents for a Sun JavaHelp system.

### **Category opened image**



Specifies the category opened image you want to use when generating Sun JavaHelp. This setting applies only to Sun JavaHelp 2.0. By default, Sun JavaHelp uses an open folder image for the category opened image in the table of contents for a Sun JavaHelp system.

### **Enable favorites tab**

Specifies whether you want to display the Favorites tab when generating Sun JavaHelp. This setting applies only to Sun JavaHelp 2.0.

### **Enable glossary tab**

Specifies whether you want to display the Glossary tab when generating Sun JavaHelp. This setting applies only to Sun JavaHelp 2.0.

### **Enable search tab**

Specifies whether you want to display the Search tab in your generated Sun JavaHelp.

### **JavaHelp location**

Specifies the location of Sun JavaHelp SDK you want ePublisher to use when generating Sun JavaHelp output.

### **JDK location**

Specifies the location of the Java Development Kit (JDK) you want ePublisher to use when generating Sun JavaHelp output.

## **Links Settings**

These settings apply only to the WebWorks Reverb (1 & 2), Microsoft HTML Help, WebWorks Help, Dynamic HTML, and XML\_XSL output formats. The settings in this category are defined as follows:

### **Baggage File Target**

Specifies how links to baggage files display in your output. A **baggage file** is any file that your source document references but is not contained in the ePublisher project, such as a `.jpeg`, `.avi`, `.swf`, `.gif`, or `.png` file or a Microsoft Word, Adobe FrameMaker, or XML file. The values for this setting are defined as follows:

Value	Description
<b>external_window</b>	Specifies the name of the window in which to open the page by typing in the name of the window. This creates a new browser window, unless one already exists with the same name. If a browser window already exists with the same name, it opens the page in that window. <b>external_window</b> is the default.
<b>_blank</b>	Specifies to always open the page in a new browser window and leave the current window in its current state.
<b>_self</b>	Specifies to open the page in the same browser window as the link tag.
<b>_parent</b>	Specifies to open the page in the immediate parent window of the link tag. This option is only useful for very specialized cases and is quite uncommon.
<b>_top</b>	Specifies to open the page in the full body of the same browser window as the link tag. Specify this value to break out of a frame all the way to the top of a window.
<b>None</b>	Specifies to open the page within the same browser window. However, unlike <b>_self</b> , the settings of the browser may control this action.

### External URL Target

Specifies how links to external URLs display in your output. The values for this setting are defined as follows:

Value	Description
<b>external_window</b>	Specifies the name of the window in which to open the page by typing in the name of the window. This creates a new browser window, unless one already exists with the same name. If a browser window already exists with the same name, it opens the page in that window. <b>external_window</b> is the default.
<b>_blank</b>	Specifies to always open the page in a new browser window and leave the current window in its current state.
<b>_self</b>	Specifies to open the page in the same browser window as the link tag.
<b>_parent</b>	Specifies to open the page in the immediate parent window of the link tag. This option is only useful for very specialized cases and is quite uncommon.
<b>_top</b>	Specifies to open the page in the full body of the same browser window as the link tag. Specify this value to break out of a frame all the way to the top of a window.
<b>None</b>	Specifies to open the page within the same browser window. However, unlike <b>_self</b> , the settings of the browser may control this action.

### Preserve Unknown File Links

If enabled, links to files that are not part of the generation will be preserved in the generated output. Normally, it is not desired to have unknown file links preserved because this can lead to file not found errors in the end-user experience. If this setting is enabled, make sure that the link has a target when it is finally deployed.

## Links Report Settings

These settings are not supported for the PDF output format. The settings in this category are defined as follows:

## Baggage file

Specifies the notification to receive if ePublisher does not successfully bring over baggage files to the baggage files area of your output folder. A **baggage file** is any file that your source document references but is not contained in the ePublisher project, such as a `.jpeg`, `.avi`, `.swf`, `.gif`, or `.png` file or Microsoft Word, Adobe FrameMaker, or XML file.

## External URLs

Specifies the notification to receive when ePublisher detects a hyperlink in your source document that refers to an external web site or email address.

## Generate links report

Specifies whether to generate Links reports.

## Unresolved link to destination in other documents

Specifies the notification to receive when ePublisher finds a document, such as a Microsoft Word, Adobe FrameMaker, or XML document, referenced by a link in your source document, but ePublisher cannot find the anchor or location within the document that your link references.

## Unresolved link to missing document

Specifies the notification to receive when ePublisher cannot resolve a link between Microsoft Word, Adobe FrameMaker, or XML source documents because ePublisher cannot find the source documents in the ePublisher project or in any location on the local computer.

## Unresolved link to missing file

Specifies the notification to receive when ePublisher cannot resolve a link between a source document and a file, such as a `.jpeg`, `.gif`, `.avi`, `.swf`, or `.pdf` file.

## Unresolved link within document

Specifies the notification to receive when ePublisher cannot find a location referenced by a link in a source document. Typically, this occurs when ePublisher cannot find the anchor or cross-reference that the link references.

## Unsupported baggage files

Specifies the notification to receive when the output format specified for the target does not support baggage files. A **baggage file** is any file that your source document references but is not contained in the ePublisher project,

such as a `.jpeg`, `.avi`, `.swf`, `.gif`, or `.png` file or Microsoft Word, Adobe FrameMaker, or XML file.

### Unsupported external URLs

Specifies the notification to receive when the output format specified for the target does not support links to external locations outside of the help system.

### Unsupported group to group links

Specifies the notification to receive when the output format specified for the target does not support group to group linking. **Group to group linking** is when you reference a location within another group in your ePublisher project.

## Locale Settings

The settings in this category are defined as follows:

### Encoding

Specifies the character encoding method used to convert bytes into characters in your output. This setting is not supported in PDF.

**Note:** In some formats this setting is not available. In these cases, the encoding is fixed to `UTF-8`.

### Locale

Specifies the language in which your output displays. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>en</b>	English
<b>fr</b>	French
<b>de</b>	German
<b>ja</b>	Japanese
<b>es</b>	Spanish
<b>it</b>	Italian
<b>ko</b>	Korean
<b>pt</b>	Portugese
<b>ru</b>	Russian
<b>sv</b>	Sweden
<b>zh</b>	Simplified Chinese for Mainland China
<b>zh_TW</b>	Traditional Chinese for Taiwan

## Menu Settings

Controls certain behaviors for the menu used to display the TOC and/or Index functions of a help output format.

The settings in this category are defined as follows:

### Generate menu

Specifies whether you want a menu to be generated in your output.

### Menu initial state

Specifies the initial state of the menu as either **Opened** or **Closed** when the help is first opened by the end-user.

### Minimum page width for docked menu

If the device viewing region displaying the help is larger than the value of this setting, then the **Menu** will be positioned (docked) next to the content page. If the viewing region is smaller than this setting value, the **Menu** will be displayed on top of the content page, potentially obscuring some of the content.

## Oracle Help Settings

These settings apply only to the `Oracle Help` output format. The settings in this category are defined as follows:

### Enable search tab

Specifies whether you want to display the Search tab in your generated Oracle Help output.

### Helpsettitle

Specifies the name of the Oracle Help system.

### JDK location

Specifies the location of the Java Development Kit (JDK) used by the Oracle Help system.

### Oracle Help Location

Specifies the location of the Oracle Help system.

## Page Settings

These settings control behavior specific to the page where content is displayed. The settings in this category are defined as follows:

### Document last modified date setting

Displays the date the source document used to generate this specific page was last modified. If enabled, this document's last modified date will be displayed at the bottom of each page generated from this document.

### Dropdown expand/collapse toggle button setting

Enables/disables a button that toggles all paragraph dropdown buttons on the currently displayed page.

### **Reverb 2.0 page style setting**

Specifies page style setting for Reverb 2.0 output. By default, the page will be generated using the **Default** (or **[Prototype]** if there is no **Default** style) page style in ePublisher.

### **Splash page style setting**

Specifies splash page style setting for Reverb 2.0 output. By default, the splash page will be generated using the **Default** (or **[Prototype]** if there is no **Default** style) page style in ePublisher.

## **PDF Settings**

These settings apply only to the PDF output format. The settings in this category are defined as follows:

### **Acrobat Distiller - PDF job settings**

Specifies the Acrobat Distiller PDF job settings you want ePublisher to use when generating PDF output. The Acrobat Distiller PDF job settings you specify must be configured on the computer where ePublisher generates output.

### **Acrobat Distiller - Use for document extensions**

Specifies the document extensions you want to use when generating PDF output using Acrobat Distiller.

### **Copy PDFs from Target**

Allows users to name the PDF target instead of generating a separate one for the help target. This is especially helpful if the user wants to specify a PDF based on PDF-XSL FO to be linked from the help system.

### **Force download on click**

Specifies the download behavior of the PDF button. Normally selecting the button causes the PDF to load in the browser. Enabling this feature forces the PDF to download to the user's file system instead.

### **Generate a combined PDF per top level group**

Specifies whether you want to generate a combined PDF for each top level group in Document Manager.



## **Generate a PDF per document**

Specifies whether you want to generate a PDF for each source document included in Document Manager.

## **PDF job settings**

Specifies the PDF job settings you want to use when generating output. When you create PDFs using Adobe FrameMaker, you can manually set job options to specify the compression for ePublisher to use when creating PDFs. Adobe FrameMaker provides predefined sets of job options that you can use to control the quality of the PDFs. This setting applies only when you generate output from Adobe FrameMaker source documents. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Screen</b>	Specifies the appropriate settings if you plan to post PDFS on a web site. This value is designed for low-resolution, on-screen display. Because of the low-resolution display quality, the screen setting allows content to download and display faster when users access and view PDF on web sites
<b>eBook</b>	Specifies the appropriate settings if users view your PDFs on desktops computers, laptop computers, and eBook reading tools.
<b>Print</b>	Specifies the appropriate settings if your PDFs will be printed. This value produces high-resolution and high image quality PDFs appropriate for printing. This value is similar to the pre press setting, but it produces a smaller file size and fewer image details.
<b>Pre Press</b>	Specifies the appropriate settings if your PDFs will be printed using a high-resolution printing method. This value produces the most detailed image quality. However, because of its high resolution, the PDFs produced using this value have the largest file sizes when compared to other PDF job settings.

## **PDF Target Window**

Specifies how links to PDF files will be opened. The values for this setting are defined as follows:

Value	Description
<b>ww_connect_pdf</b>	Specifies the name of the window in which to open the page by typing in the name of the window. This creates a new browser window, unless one already exists with the same name. If a browser window already exists with the same name, it opens the page in that window. <b>ww_connect_pdf</b> is the default.
<b>_blank</b>	Specifies to always open the page in a new browser window and leave the current window in its current state.
<b>_self</b>	Specifies to open the page in the same browser window as the link tag.
<b>_parent</b>	Specifies to open the page in the immediate parent window of the link tag. This option is only useful for very specialized cases and is quite uncommon.
<b>_top</b>	Specifies to open the page in the full body of the same browser window as the link tag. Specify this value to break out of a frame all the way to the top of a window.

### Save As PDF - PDF job settings

Specifies the Acrobat Distiller PDF job settings you want FrameMaker to use when generating PDF output under the direction of ePublisher. If this setting is empty, then ePublisher will not use FrameMaker's "Save as PDF" method. The Acrobat Distiller PDF job settings you specify must be configured on the computer where ePublisher generates output.

This option is only necessary when using FrameMaker and you have embedded videos in your content that you want to be published in the PDF output.

**Note:** If your output generation requires page rotations, then you should not use this target setting.

### Save As PDF - Use for document extensions

Specify the FrameMaker file extensions to apply the “Save As PDF” operation to.

## Result Options Settings (PDF - XSL-FO)

These settings apply only to the PDF - XSL-FO output format. The settings in this category are defined as follows:

### **Generate bookmarks**

Specifies whether or not to generate acrobat bookmarks based on the table of contents settings.

### **Generate group result**

Specifies whether or not a PDF will be generated for an entire group of documents or ditamaps within an ePublisher project.

### **Generate per document result**

Specifies whether or not a PDF will be generated for each document or ditamap within an ePublisher project.

### **Generate project result**

Specifies whether or not a PDF will be generated that includes all the documents or ditamaps of all the groups in an ePublisher project.

## Search Settings

This setting controls behavior for end-user search in the generated output. The settings in this category are defined as follows:

### **Minimum character amount to execute search**

Specifies the minimum amount of characters needed to execute a search. When users type the minimum amount of characters specified in this setting, an initial list of search results will begin to be displayed in the search results area. Reducing the value of this setting will make the progressive search begin to display sooner with suggested results. However, it will also make the help download file size larger and potentially slower for your end-users the first time they access the online help. After the first access of any page in the online help, the end-user’s browser will start downloading the search files and cache them for use when the user does perform a search. The cached search files may have the effect of making the download size not an issue depending on end-user behavior.

## **Progressive search**

This setting allows search result suggestions to begin being displayed before a user completes their search term input. Specifies whether progressive search is enabled/disabled.

## **Search result count**

Allows end-users to see how many search results there are for a given search term input. Specifies whether search result count is enabled/disabled.

## **Search scope filtering**

This setting creates a dropdown menu displaying all the groups and sub-groups defined in your ePublisher project within the **Document Manager**. This menu is available to your end-users next to the search input in the toolbar. End-users can then select which group(s) they want to perform a search within. Search results will not be displayed for groups that are not selected. Specifies whether search scope filtering is enabled/disabled.

# **Social Settings**

These settings are specific to the WebWorks Reverb Format for integration with social media. Please refer to the specific site for further instructions once integrated

## **Disqus - Allow non-public networks**

Enables users to post Disqus comments behind a firewall or non-public website.

## **Disqus Identifier**

Enables support for user comments using the Disqus comment web service

## **FaceBook Like**

Enables users to report "I like this!" in Facebook

## **LinkedIn Share**

Enable users to share individual pages within the LinkedIn community.

## **Tweet This!**

Enable Twitter "Tweet This!" support

# **Styles Settings (PDF - XSL-FO)**

These settings apply only to the PDF - XSL-FO output format. The settings in this category are defined as follows:

### **Content page style**

Specifies the ePublisher page style to use for generating all non-TOC and non-Index pages.

### **Index page style**

Specifies the ePublisher page style to use for generating the Index pages.

### **Index style prefix**

Specifies the prefix that will be used for the generated index style names such as: **IXGroup**, **IX1**, **IX2**, **IX3**. These styles correspond to paragraph styles that can be created in the ePublisher Style Designer.

### **Index title style**

Specifies the ePublisher paragraph style to use for the title of the generated index pages.

### **MiniTOC container style**

Specifies the ePublisher paragraph style to use for generating mini-TOC sections.

### **MiniTOC style**

Specifies the ePublisher paragraph style to use for generating mini-TOC paragraphs.

### **Related Topics Title Style**

Specifies the ePublisher paragraph style to use for the title of the generated related topics sections.

### **TOC page style**

Specifies the ePublisher page style to use for generating the TOC pages.

### **TOC style prefix**

Specifies the prefix that will be used for the generated TOC style names such as: **TOC1**, **TOC2**, **TOC3**, **TOC4**. These styles correspond to paragraph styles that can be created in the ePublisher Style Designer.

### **TOC title style**

Specifies the ePublisher paragraph style to use for the title of the generated TOC pages.

## Styles Report Settings

The settings in this category are defined as follows:

### Generate styles report

Specifies whether to generate the Styles report.

### Non-standard styles

Specifies the notification ePublisher generates if ePublisher detects styles in your source document that are undefined in the Stationery used by your project.

### Style overrides

Specifies the notification ePublisher generates if ePublisher detects style overrides in your source documents. A **style override** is any modification made to the original style definition defined in the Stationery used by your project.

## Table of Contents Settings

The settings in this category are defined as follows:

### Collapse table of contents

Specifies how you would like skipped heading levels handled in the table of contents for your output. For more information, see "Defining the Table of Contents from an Irregular Heading Hierarchy". The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
Don't collapse	Inserts empty table of content entries for the skipped levels.
Re-label	Inserts labeled entries for the skipped table of contents levels. When you specify this value, ePublisher displays the heading text from the table of contents entry below the current entry as the table of contents label.
Fully collapse	Removes all skipped heading levels and table of contents entries and places all table of contents headings at the same level, regardless of the table of contents level specified in the Stationery.
Smart collapse	Removes empty table of contents entries and moves the heading that follows an empty table of content entry up a level to replace the skipped table of contents level.

### **Generate table of contents**

Specifies whether ePublisher generates a table of contents.

### **Table of Contents filename**

Specifies the file name ePublisher should use when ePublisher generates a table of contents file in certain output formats.

## **Title Page Settings (PDF - XSL-FO)**

These settings apply only to the PDF - XSL-FO output format. The settings in this category are defined as follows:

### **Generate Publish Date on Title Page**

Specifies whether or not to generate a publication date on the title page.

### **Generate Title Page**

Specifies whether or not to generate a title page.



**Title Page Style**

Specifies the ePubublisher page style to use for generating the title page.

**Title Page title**

Specifies how the text of the title page title will be calculated. The following choices are available with the current installation of ePubublisher:

<b>Value</b>	<b>Description</b>
\$Title;	This value is set via the ePublisher merge setting title of the given group or project. Merge setting titles are configured under the Merge Settings dialog. In the case of a PDF for a single document or ditamap, then this value is the top-level heading.
\$PublishDate;	This value is set to the output generation date using the string formatting specified in <code>locales.xml</code> .
None	No title will be generated.

**Note: Project Variables** and **Style Variables** can also be used in this setting.

### **Title Page Title style**

Specifies the ePublisher paragraph style to use for generating the title page's title.

### **Title Page Sub-title**

Specifies how the text of the title page sub-title will be calculated. The following choices are available with the current installation of ePublisher:

Value	Description
\$Title;	This value is set via the ePublisher merge setting title of the given group or project. Merge setting titles are configured under the Merge Settings dialog. In the case of a PDF for a single document or ditamap, then this value is the top-level heading.
\$PublishDate;	This value is set to the output generation date using the string formatting specified in <code>locales.xml</code> .
None	No sub-title will be generated.

**Note:** **Project Variables** and **Style Variables** can also be used in this setting.

### Title Page Sub-title style

Specifies the ePublisher paragraph style to use for generating the title page's sub-title.

## Toolbar Settings

The settings in this category are defined as follows:

### Google Translate Button

Enables support for the Google Translate web service. Requires your output to be deployed on a public-facing web server.

### Home setting

Specifies whether you want to display the linked home button in your toolbar.

### Linked toolbar logo

Specifies whether you want your toolbar logo to be linked to the home (splash or first page) page or the **Company webpage**.

### Toolbar logo

Specifies whether you want to display a logo in your toolbar. Toolbar logo can inherit from **Company name** or **Company logo image**.

### Toolbar logo link address

Specifies the logo link address.

### Toolbar logo override

Specifies which image to use as the toolbar logo. Store the image file in the `Files` folder within your project.

## Topics Report Settings

These settings are not supported for the Dynamic HTML, XML+XSL, PDF - XSL-FO, and PDF output formats. The settings in this category are defined as follows:

### Generate topics report

Specifies whether to generate the Topics report.

### Topic links

Specifies the notification ePublisher generates if ePublisher detects that ePublisher did not create a context-sensitive help topic file for each context-sensitive help topic ID you specified using the **TopicAlias** marker.

### Duplicate topic aliases

Specifies whether or not there are two or more **TopicAlias** markers with the same marker text

## WebWorks Help Settings

These settings apply only to the WebWorks Help output format. These settings are defined as follows:

### Automatically synchronize the TOC

Specifies whether you want to allow users to locate the topic they are viewing in the table of contents. When you enable this setting, when a user clicks the "Show in Contents" button located to the left of the navigation button, WebWorks Help highlights the entry in the Contents tab that corresponds to the currently displayed topic.

### Cookie expiration in days

Specifies the number of days before cookies expire.

### Cookie id

Specifies the cookie ID.

### **Show bookmark toolbar button**

Specifies whether you want to display the bookmark toolbar button in your output.

### **Show favorites tab**

Specifies whether you want to display the Favorites tab in your output.

### **Show first document instead of splash page**

Specifies whether you want to show the first topic in your WebWorks help system instead of the splash page in your output.

### **Show PDF button**

Specifies whether you want to display the PDF button in your output. If you enable this setting, WebWorks Help displays a PDF file of the source document from which the currently displayed topic was generated.

### **Show previous and next tool bar buttons**

Specifies whether you want to display the previous and next tool bar buttons in your output. The previous button allows users to navigate back to topics that precede the currently displayed topic in the help. The Next button allows users to navigate forward to topics that follow the currently displayed topic in the help.

### **Show print toolbar button**

Specifies whether you want to display the print toolbar button in your output.

### **Show related topics inline button**

Specifies whether you want to display related topics links as a list in the topic itself as well as in a popup window when the user clicks a related topics button.

### **Show related topics toolbar button**

Specifies whether you want to display the related topics toolbar button in your output.

### **Show search tab**

Specifies whether you want to display the Search tab in your output.

## Theme

Specifies the theme you want to use in your output.

## Top level filename

Specifies the top-level entry-point file for the generated WebWorks Help output.

## Use browser cookies

Specifies whether you want to use browser cookies.

## WebWorks Help accessibility

Specifies whether you want to enable WebWorks Help accessibility, or if you want to ask users if they want to enable WebWorks Help accessibility on first use.

# WebWorks Reverb Settings

## Display large images in lightbox

When a thumbnail is used for an image, the full size version of the image can be viewed by clicking the image which is then displayed in a lightbox. When disabled the image displays in a separate file.

## Enabled Print Icon

Enables/disables the print icon in generated pages

## Entry Filename

Specifies the name of the Reverb entry-point file name (default is "index.html")

## Feedback Email

Defines the feedback email address for use in generated pages

## Feedback Email Message

Defines the contents of both the subject line and body section of the email message sent when **Feedback Email** address is specified. Use `$Location` to include the current page's URL in the message. Please note that this message is single-line only.

## Home

Provides a button in the toolbar that links to the entry point of the Reverb help set. If the splash page is enabled then it will link to the splash page, otherwise it will link to the first page of actual content.

### **Minimum height for non-scrolling toolbar**

Specifies the minimum browser window height required for Reverb to fix the toolbar at the top of the window instead of scrolling out of view.

### **Minimum page width for sidebar**

Specifies the minimum browser window width required for Reverb to display the TOC/Index sidebar panel next to the content panel. If the width of the window is less than this value, then the TOC/Index will be displayed in place of the content panel and only when the TOC/Index button has been selected.

### **Reverb Page Style**

Specifies the page style to use when creating the outer-level HTML file that contains all the Reverb panels (i.e. TOC, index, search, content).

### **Search Implementation**

Specifies the type of search implementation or engine to use. Currently, there are 2 choices available:

<b>Value</b>	<b>Description and Deployment Requirements</b>
Client-side Search	<p>Provides a stand-alone search capability using javascript that is executed on the end-user's browser client. Provides basic search capabilities.</p> <p>No additional requirements.</p>

### **Sidebar Width**

Specifies the width of the TOC or Index sidebar panel.

### **Skin**

Specifies an alternate skin plugin to use when generating Reverb Help. Skin plugins allow for a wide range of look-and-feel differences for Reverb Help. The following choices are available with the current installation of ePublisher Designer:



<b>Skin Types</b>	<b>Color Schemes</b>
Classic Classic - Compact	
Corporate Corporate - Compact	Blue Grey Red
Metro Metro - Compact	Blue Dark Green Grey Light Green Orange Purple Red Tiled
Social Social - Compact	Blue Green Grey Light Blue Red

## **Splash Page Style**

Specifies the page style to use when processing the splash page

### **Use first document as splash page**

Determines initial page displayed. If set to Enabled then no splash page will be created and instead the first page will be used.

### **WebWorks Help API Compatibility**

Enables users to use the same help API as used for WebWorks Help 4 and 5 with Reverb Help.

## **WebWorks Reverb 2.0 Settings**

### **Browser Tab Icon (favicon)**

Allows you to select an icon file from the **User Files** location in the project. This file will then be used as the favicon which displays in the browser tab.

### **Browser Tab Title**

Specifies the value used in the title of the internet browser tab. When **Page Title** is used, the title of the currently open page in the Reverb 2.0 output will populate the browser tab. When Merge Title is used, the Merge Title value in the Merge Settings window will populate the browser tab.

### **Display large images in lightbox**

Enable/disable the lightbox setting. Lightbox displays the full size of images that have been made into thumbnails.

### **Enabled Print Icon**

Enables/disables the print icon in generated pages

### **Entry Filename**

Specifies the name of the Reverb entry-point file name (default is "index.html")

### **Feedback Email**

Defines the feedback email address for use in generated pages

### **Feedback Email Message**

Defines the contents of the body section of the email message sent when **Feedback Email** address is specified. Use `$Location` to include the current page's URL in the message. Please note that this message is single-line only.

### **Skin**

Specifies an alternate skin plugin to use when not using the default skin: Neo. It is recommended that you leave this field empty, thus using the default skin: Neo. Neo has been optimized to fully take advantage of Reverb 2.0 design. You can still customize the look of the skin using SASS. However, if you prefer, you can select from the alternate skin plugins that have been provided as a means for mimicking an existing Reverb 1 design. The following choices are available with the current installation of ePublisher Designer:

<b>Skin Types</b>	<b>Description</b>
Classic	This skin features a larger toolbar, and gradients across the layout to produce a traditional help look and feel. This skin is great for all around acceptability and use across any platform and website.
Corporate	Modeled after high profile technology websites. The Corporate skin has a polished look and would suit the needs of a high-profile technology company's help set.
Metro	The Metro skin represents the latest ideas in computer interface design and comes straight from the Metro Design Language, now the heart of interfaces used in all Microsoft desktop, smartphone, and game box operating platforms.
Neo	The default skin for WebWorks Reverb 2.0. Neo is the latest design for Reverb output and was designed with current web aesthetics in mind. It features a simplistic layout that looks great across many devices. This skin is very versatile and well suited for most purposes.
Social	The Social skin has been designed to provide a user experience most similar to that of social networking websites. It provides a familiar interface that is intuitive and casual enough to maximize your end-user participation.

### **Use first document as splash page**

Determines initial page to be displayed. If set to Enabled, the first page will be used as the splash page.

### **WebWorks Help API Compatibility**

Enables users to use the WebWorks Help API.

# Style Designer Reference

- Advanced Properties
- Aural Properties
- Background Properties
- Body Properties
- Body Background Properties (Tables)
- Border Properties
- Bullet Properties
- Font Properties
- Footer Properties (Tables)
- Footer Background Properties (Tables)
- Header Properties (Tables)
- Header Background Properties (Tables)
- HTML (Layout) Properties
- Margin Properties
- Markdown++ Properties
- Master Page Properties (Pages)
- Navigation Properties (Pages)
- Padding Properties
- Pagination Properties
- Table Properties (Tables)
- Text Properties
- Markdown++ Options
- Paragraph Styles Options
- Character Styles Options
- Table Styles Options
- Page Styles Options
- Page Styles Options (PDF - XSL-FO)
- Graphic Styles Options
- Marker Styles Options

The Style Designer allows you to specify the appearance and functionality of online content, including paragraphs, characters, tables, page layouts, images, table of contents levels, popups, and related topics. ePublisher Designer builds Stationery based on the settings you specify as properties and/or options.

## Advanced Properties

Available only in the *PDF - XSL-FO* format.

These properties are for use with output formats that require styling and layout characteristics above and beyond what is provided by CSS. Links all point to [this website](#)

### Absolute Position

<http://www.w3.org/TR/xsl11/#absolute-position>

### Alignment Adjust

<http://www.w3.org/TR/xsl11/#alignment-adjust>

**Alignment Baseline**

<http://www.w3.org/TR/xsl11/#alignment-baseline>

**Baseline Shift**

<http://www.w3.org/TR/xsl11/#baseline-shift>

**Block Progression Dimension**

<http://www.w3.org/TR/xsl11/#block-progression-dimension>

**Clear**

<http://www.w3.org/TR/xsl11/#clear>

**Display Align**

<http://www.w3.org/TR/xsl11/#display-align>

**Dominant Baseline**

<http://www.w3.org/TR/xsl11/#dominant-baseline>

**End Indent**

<http://www.w3.org/TR/xsl11/#end-indent>

**Float**

<http://www.w3.org/TR/xsl11/#float>

**Font Selection Strategy**

<http://www.w3.org/TR/xsl11/#font-selection-strategy>

**Font Size Adjust**

<http://www.w3.org/TR/xsl11/#font-size-adjust>

**Font Stretch**

<http://www.w3.org/TR/xsl11/#font-stretch>

**Glyph Orientation Horizontal**

<http://www.w3.org/TR/xsl11/#glyph-orientation-horizontal>

**Glyph Orientation Vertical**

<http://www.w3.org/TR/xsl11/#glyph-orientation-vertical>

### **Inline Progression Dimension**

<http://www.w3.org/TR/xsl11/#inline-progression-dimension>

### **Intrusion Displace**

<http://www.w3.org/TR/xsl11/#intrusion-displace>

### **Last Line End Indent**

<http://www.w3.org/TR/xsl11/#last-line-end-indent>

### **Leader Alignment**

<http://www.w3.org/TR/xsl11/#leader-alignment>

### **Leader Length**

<http://www.w3.org/TR/xsl11/#leader-alignment>

### **Leader Pattern**

<http://www.w3.org/TR/xsl11/#leader-pattern>

### **Leader Pattern Width**

<http://www.w3.org/TR/xsl11/#leader-pattern-width>

### **Line Stacking Strategy**

<http://www.w3.org/TR/xsl11/#line-stacking-strategy>

### **Linefeed Treatment**

<http://www.w3.org/TR/xsl11/#linefeed-treatment>

### **Maximum Height**

<http://www.w3.org/TR/xsl11/#max-height>

### **Maximum Width**

<http://www.w3.org/TR/xsl11/#max-width>

### **Provisional Distance Between Stars**

<http://www.w3.org/TR/xsl11/#provisional-distance-between-starts>

**Provisional Label Separation**

<http://www.w3.org/TR/xsl11/#provisional-label-separation>

**Reference Orientation**

<http://www.w3.org/TR/xsl11/#reference-orientation>

**Relative Align**

<http://www.w3.org/TR/xsl11/#relative-align>

**Relative Position**

<http://www.w3.org/TR/xsl11/#relative-position>

**Role**

<http://www.w3.org/TR/xsl11/#role>

**Rule Style**

<http://www.w3.org/TR/xsl11/#rule-style>

**Rule Thickness**

<http://www.w3.org/TR/xsl11/#rule-thickness>

**Score Spaces**

<http://www.w3.org/TR/xsl11/#score-spaces>

**Source Document**

<http://www.w3.org/TR/xsl11/#source-document>

**Space After**

<http://www.w3.org/TR/xsl11/#space-after>

**Space Before**

<http://www.w3.org/TR/xsl11/#space-before>

**Space End**

<http://www.w3.org/TR/xsl11/#space-end>

**Space Start**



<http://www.w3.org/TR/xsl11/#space-start>

### **Span**

<http://www.w3.org/TR/xsl11/#span>

### **Start Indent**

<http://www.w3.org/TR/xsl11/#start-indent>

### **Suppress at Line Break**

<http://www.w3.org/TR/xsl11/#suppress-at-line-break>

### **Text Align**

<http://www.w3.org/TR/xsl11/#text-align>

### **Text Align Last**

<http://www.w3.org/TR/xsl11/#text-align-last>

### **Text Altitude**

<http://www.w3.org/TR/xsl11/#text-altitude>

### **Text Depth**

<http://www.w3.org/TR/xsl11/#text-depth>

### **Text Shadow Blur Radius**

<http://www.w3.org/TR/xsl11/#text-shadow-blur-radius>

### **Text Shadow Color**

<http://www.w3.org/TR/xsl11/#text-shadow-color>

### **Text Shadow Vertical Offset**

<http://www.w3.org/TR/xsl11/#text-shadow-vertical-offset>

### **Treat as Word Space**

<http://www.w3.org/TR/xsl11/#treat-as-word-space>

### **White Space Collapse**

<http://www.w3.org/TR/xsl11/#white-space-collapse>

## **White Space Treatment**

<http://www.w3.org/TR/xsl11/#white-space-treatment>

## **Wrap Option**

<http://www.w3.org/TR/xsl11/#wrap-option>

# **Aural Properties**

Available only in the *PDF - XSL-FO* format.

These properties are for use with output formats that require aural property characteristics.

## **Azimuth**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-azimuth>

## **Cue After**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-cue-after>

## **Cue Before**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-cue-before>

## **Elevation**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-elevation>

## **Pause After**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-pause-after>

## **Pause Before**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-pause-before>

## **Pitch**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-pitch>

## **Pitch Range**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-pitch-range>

## **Play During**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-play-during>

### **Richness**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-richness>

### **Speak**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-speak>

### **Speak Header**

<http://www.w3.org/TR/REC-CSS2/tables.html#propdef-speak-header>

### **Speak Numeral**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-speak-numeral>

### **Speak Punctuation**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-speak-numeral>

### **Speech Rate**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-speech-rate>

### **Stress**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-stress>

### **Voice Family**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-voice-family>

### **Volume**

<http://www.w3.org/TR/REC-CSS2/aural.html#propdef-volume>

## **Background Properties**

In terms of the CSS box model, the background for a style refers to the background of the content and the padding areas. If you increase the padding for a style, the background color area for that style also increases.

### **Color**

Specifies the color for the background. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB

values can range from the lowest RGB value of 0 (hex #00) to the highest value of 255 (hex #FF). For example, the RGB hexadecimal notation for black is 000000 and white is FFFFFFFF.

## Image

Specifies the background image. This field lists only the image files located in the Files folder of your project. If this field does not list the background image you want, press F12 and verify the image is in the Files folder for your project. Once you copy the image into the Files folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

## Tiling

Specifies whether the background image repeats. The table must be larger than the background image to see the image repeated in the background. The background image size also determines if an image repeat shows when set horizontally or vertically. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>No repeat</b>	Indicates the background image does not repeat.
<b>Repeat</b>	Indicates the background image repeats.
<b>Repeat horizontally</b>	Indicates the background image repeats horizontally.
<b>Repeat vertically</b>	Indicates the background image repeats vertically.

## Scrolling

Specifies whether the background image remains in a fixed location as the user scrolls down the page. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Scroll</b>	Scrolls the background image as the user scrolls through the page contents.
<b>Fixed</b>	Keeps the background image in a fixed location as the user scrolls through the page contents.

## **Horizontal**

Specifies the starting horizontal position of a background image. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-width.
<b>Left</b>	The left side of the background image aligns with the leftmost side of the page.
<b>Center</b>	The center of the background image aligns with the center of the page.
<b>Right</b>	The right side of the background image aligns with the right of the page.

## **Vertical**

Specifies the starting vertical position of a background image. The values for this setting are defined as follows:

Value	Description
<b>Custom</b>	The background image aligns based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-height.
<b>Top</b>	The top of the background image aligns with the top of the tallest element on the line
<b>Center</b>	The center of the background image aligns with the middle of the parent element bottom.
<b>Bottom</b>	The bottom of the background image aligns with the lowest element on the line.

## Body Properties

**Body rows** are the rows of a table that contain the data. Values you define for table body rows establish the default values for the table, unless you indicate separate header and footer row values.

In Microsoft Word and Adobe FrameMaker, you can define a header row for your table. In FrameMaker, you can also define a footer row. Body rows are typically all rows in a table not already defined as a header or footer rows. However, unless you explicitly specify a row as a header row, ePublisher considers all rows except the footer rows as body rows.

Within body rows, ePublisher lets you specify the table padding. **Padding** refers to the space between the body border and the body content. You can change the left, top, right, and bottom padding values independently or set all properties to have the same padding. Padding values must be positive numbers.

### Left



Specifies the left padding for table body rows. Set a custom value by specifying a number and selecting the unit of measure. Select **Auto** for ePublisher to automatically set table row padding.

## Top

Specifies the top padding for table body rows. Set a custom value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). Select **Auto** for ePublisher to automatically set table row padding.

## Right

Specifies the right padding for table body rows. Set a custom value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). Select **Auto** for ePublisher to automatically set table row padding.

## Bottom

Specifies the bottom padding for table body rows. Set a custom value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). Select **Auto** for ePublisher to automatically set table row padding.

## Vertical

Specifies the vertical alignment for the content in table body rows. Vertical alignment of content is very important for readability of online information.

If the cells across a table row have differing numbers of content lines, align the content to either the top or the bottom of the table row so that all of the row cells across the page align. Typically, cells within the body of a table area are top-aligned and the header cells for the table are bottom-aligned.

## Color

Specifies the border color for table body rows. Select a color from the list or type the RGB value of the border color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

To select color values for all body row borders, specify the color on the **All** tab. To set unique color values for each individual body row border, click the **Left**, **Right**, **Top** or **Bottom** tab and separately set each border value.

This field sets the border color for body rows in a table, not the entire table border. To set the border color for an entire table, select **Border** from the **Table Styles** properties.

## Style

Specifies a border style in table body rows, such as a dotted, dashed, or solid line.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. Various browsers and operating systems often display the size and spacing of the dots in a dotted line differently.

## Width

Specifies the width of a border in table body rows, such as a thick or thin border. Choose a value or set a custom width by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

# Body Background Properties (Tables)

Body Background properties let you specify a background color for table body rows, set the background as transparent, or specify an image to use in the background of body rows. Indicate alternative colors or background images to alternate the appearance of rows or columns in a table.

To understand how ePublisher modifies table properties, visualize the body rows of a table as sitting on a layer above the table background. When you modify the background color of the body rows, you actually apply an opaque color over the color of the entire table background color. To get the desired results, you may need to alter other body row elements, such as paragraphs. For example, if you cannot make the body row transparent, make sure you correctly set the background property for all the different **Table Styles** properties.

## Color

Specifies the background color of table body rows. Select a color from the list or type the RGB value of the background color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0`

(hex #00) to the highest value of 255 (hex #FF). For example, the RGB hexadecimal notation for black is 000000 and white is FFFFFFFF.

### Image (Background Image Area)

Specifies the background image for table body rows. This field lists only the image files located in the Files folder of your project. If this field does not list the background image you want, press F12 and verify the image is in the Files folder for your project. Once you copy the image into the Files folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

### Tiling (Background Image Area)

Specifies whether the background image repeats in table body rows. The body rows must be larger than the background image to see the image repeated in the background of the body rows. The background image size also determines if an image repeat shows when set horizontally or vertically. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>No repeat</b>	Specifies that the background image does not repeat.
<b>Repeat</b>	Specifies that the background image repeats.
<b>Repeat horizontally</b>	Specifies that the background image repeats horizontally.
<b>Repeat vertically</b>	Specifies that the background image repeats vertically.

### **Scrolling (Background Image Area)**

Specifies whether the specified image remains in one fixed location as the user scrolls through content. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Scroll</b>	Scrolls the background image as the user scrolls through content.
<b>Fixed</b>	Keeps the background image in a fixed location as the user scrolls through content.

### **Horizontal (Background Image Area)**

Specifies the horizontal position of the background image for table body rows.  
The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns in the table row based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-width.
<b>Left</b>	The left side of the background image aligns with the leftmost side of the table row.
<b>Center</b>	The center of the background image aligns with the center of the table row.
<b>Right</b>	The right side of the background image aligns with the right of the table row.

### **Vertical (Background Image Area)**

Specifies the vertical position of the background image of table body rows.  
The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns on the table row based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-height.
<b>Top</b>	The top of the background image aligns with the top of the tallest element on the table row.
<b>Center</b>	The center of the background image aligns with the middle of the parent element bottom on the table row.
<b>Bottom</b>	The bottom of the background image aligns with the lowest element on the table row.

## Context

Specifies whether to use alternate colors or images for alternating rows or columns of a table.

Alternate shading of rows or columns is a useful layout to help minimize the number of lines and amount of information displayed, while organizing the data in a way that users can easily read and understand.

## Default period

Specifies the number of rows or columns that use the default color in the body row. You must first set the default body row color in the **Color** field of the **Body Background** properties for the table style. Specify a minimum value of **1**. If you specify a value of **0**, ePublisher does not use alternate rows or columns and displays entire body row in the default color.

## Alternate period

Specifies how many consecutive rows or columns use the alternate color or image. Specify a value greater than 0. For example, if you set **Alternate period** to **2**, after the first two rows of a table, the next two rows would use the alternate color or image. Then, there would be two plain rows, followed by two more using the alternate color or image, and so on.

### Alternate Color

Specifies the color to use as the alternate color. Select a color from the list or type the RGB value for the alternative color, such as `000000`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

### Image (Alternate Info, Background Image Area)

Specifies the alternate background image for table body rows. This field lists only the image files located in the `Files` folder of your project. If this field does not list the background image you want, press `F12` and verify the image is in the `Files` folder for your project. Once you copy the image into the `Files` folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

### Tiling (Alternate Info, Background Image Area)

Specifies whether to repeat the alternate background image in table body rows. The table must be larger than the alternate background image to see the image repeated in alternate table body rows. The alternate background image size also determines if an image repeat shows when set horizontally or vertically. The values for this setting are defined as follows:



<b>Value</b>	<b>Description</b>
<b>No repeat</b>	Specifies that the background image does not repeat.
<b>Repeat</b>	Specifies that the background image repeats.
<b>Repeat horizontally</b>	Specifies that the background image repeats horizontally.
<b>Repeat vertically</b>	Specifies that the background image repeats vertically.

### **Scrolling (Alternate Info, Background Image Area)**

Specifies whether the specified alternate background image remains in one fixed location as the user scrolls through content. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Scroll</b>	Scrolls the background image as the user scrolls through content.
<b>Fixed</b>	Keeps the background image in a fixed location as the user scrolls through content.

### **Horizontal (Alternate Info, Background Image Area)**

Specifies the horizontal position for the alternate background image in table body rows. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The alternate background image aligns in the table row based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-width.
<b>Left</b>	The left side of the alternate background image aligns with the leftmost side of the table row.
<b>Center</b>	The center of the alternate background image aligns with the center of the table row.
<b>Right</b>	The right side of the alternate background image aligns with the right of the table row.

### **Vertical (Alternate Info, Background Image Area)**

Specifies the vertical position for the alternate background image in table body rows. The values for this setting are defined as follows:

Value	Description
<b>Custom</b>	The alternate background image aligns on the table row based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-height.
<b>Top</b>	The top of the alternate background image aligns with the top of the tallest element on the table row.
<b>Center</b>	The center of the alternate background image aligns with the middle of the parent element on the table row.
<b>Bottom</b>	The bottom of the alternate background image aligns with the lowest element on the table row.

## Border Properties

**Borders** are lines that you can draw around any or all of the four sides of a style. In terms of the CSS box model, increasing the padding for a style increases the space between the content and the border.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. Various browsers and operating systems often display the size and spacing of the dots in a dotted line differently.

### Color

Specifies the border color. Select a color from the list or type the RGB value for the border, such as `000000`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

This field sets the border color for an entire table, not borders around body rows. To set the border color for table body rows, select **Body** in the **Table Styles** properties and specify the border settings for body rows.

## Style

Specifies a border style, such as a dotted, dashed, or solid line.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. Various browsers and operating systems often display the size and spacing of the dots in a dotted line differently.

## Width

Specifies a border width, such as a thick or thin border. Choose a value or set a custom width by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

# Bullet Properties

By default, ePublisher generates output for bulleted lists based on the properties in the original source content. Some source applications use the Symbol or Wingdings font for bullets. These fonts may not be available on all computers, which can cause the bullets in your output to display incorrectly. To avoid this issue, use an image for bullets. Also, you can create a character style with a font family assigned, such as sans-serif, and then apply that character style through ePublisher to the characters you choose to define for bullets.

**Note:** When inserting text for either the **Text** or **Separator** properties described below, make sure to use unicode characters. You can copy-and-paste unicode characters from another program such as the Windows accessory program called **Character Map**, or you can use a Windows ALT key sequence.

## **Steps for inserting special characters using the Windows ALT key sequence**

- 1.** Position the cursor where you want the special character to appear
- 2.** Make sure that the numeric keypad `Num Lock` is on
- 3.** Press and hold the `Alt` key and then press the keys on the numeric keypad that represent the keystroke value of the character you want to input
- 4.** After you finish typing, release the `Alt` key

Character Name	ALT Key Sequence
Round Bullet	0149
En Dash	0150
Em Dash	0160

## Text

Specifies a special character for bullets. Enter the special character using the Windows ALT key code or by copying and pasting a unicode character from another program. Common bullet characters are round bullet, en dash, and em dash.

## Separator

Specifies the type of separator to use between the bullet and the beginning of the first text line, such as a space or a colon.

## Image

Specifies the image to use for bullets. This field lists only the image files located in the **Files** folder of your project. If you do not see the background image you want in this field, press **F12** and verify the image is in the **Files** folder for your project. Once you copy the image into the **Files** folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

## Character Style

Specifies a character style for your bullet to ensure the bullet character uses the correct font. Use any character style available in your project. To associate an existing character style with a custom bullet, select the character style in this field.

# Font Properties

## Family

Specifies a font family. Click the **Ellipsis** button to open the Font Family Picker window where you can select from a list of installed fonts, specify a generic font family, or select a custom font family.

Consider specifying a generic font family, such as sans-serif, rather than a specific font. Many browsers and help systems use only fonts installed on the user's computer. If you indicate a specific font, a user who does not have that font installed does not see the help output correctly. By selecting a generic font family, the user does not have to have the exact font specified, only a font from that family.

If you specify multiple fonts, separated by commas, the user's browser displays the first available font in the list. For example, if you specify `Verdana, Arial, Helvetica, sans-serif` and the user does not have the Verdana font installed, the browser displays in the Arial font for the help output instead.

## Size

Specifies the size of the font. Select a value from the list or select **Custom**, and then specify a number and unit of measure for the custom font size. The **smaller** value sets the font to a smaller size than the parent element. The **larger** value sets the font to a larger size than the parent element.

## Style

Specifies the font style, such as **italic**.

## Variant

Specifies the font variation, such as **small caps**.

## Weight

Specifies how thick or thin ePublisher displays text characters, such as **bold**. For a more precise weight, select from the numeric values, which range from **100** to **900**. For example, a weight of **400** is the same as normal and **700** is the same as bold.

# Footer Properties (Tables)



Tables created in Adobe FrameMaker identify the footer rows of tables, so you can quickly specify the **Footer** and **Footer background** properties in the **Table Styles** properties in Style Designer to modify footer rows for online delivery.

Tables created in Microsoft Word do not overtly identify footer rows. To control footer rows with Microsoft Word and ePublisher, set up the table in your source document to reflect the desired appearance, or use footer paragraph styles in the footer row of the table. Then, ePublisher uses the footer paragraph styles in your source document and you can use the footer **Table Styles** properties in Style Designer to modify the appearance as needed.

You can also specify footer padding properties for tables. **Padding** refers to the space between the table footer border and the table footer content. You can change the left, top, right, and bottom padding values independently or set all properties to have the same padding. Padding values must be positive numbers.

### **Left**

Specifies the left padding for a table footer, or the space between the left table border and where the content begins in the left of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

### **Top**

Specifies the top padding for a table footer, or the space between the top table border and where the content aligns at the top of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

### **Right**

Specifies the right padding for a table footer, or the space between the right table border and where the content ends at the right of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

### **Bottom**

Specifies the bottom padding for a table footer, or the space between the bottom table border and where the content ends at the bottom of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include:

percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

## Vertical

Specifies the vertical alignment of the content in a table footer. The values for this setting are defined as: **Top**, **Middle**, and **Bottom**.

## Color (Borders, Table Styles)

Specifies the border color of a table footer. Select a color from the list or type the RGB value for the border. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of 0 (hex #00) to the highest value of 255 (hex #FF). For example, the RGB hexadecimal notation for black is 000000 and white is FFFFFFFF.

## Style

Specifies a border style for a table footer, such as a dotted, dashed, or solid line.

Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. Various browsers and operating systems often display the size and spacing of the dots in a dotted line differently.

## Width

Specify a width for the border of a table footer, such as a thick or thin border. Choose a value or set a custom width by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

# Footer Background Properties (Tables)

To help you visualize how you can modify properties, think of the footer rows of a table as sitting on a layer above the background of a table. By modifying the color of the footer rows, you are applying an opaque color over the color of the background color. You may need to modify other elements, such as paragraphs, that reside within the footer row to get the desired results. For example, if you cannot make the footer row transparent, make sure the background property is properly set for all the different layers in the table.

Tables created in Adobe FrameMaker identify the footer rows of tables, so you can quickly specify the **Footer** and **Footer background** properties in the **Table Styles** properties in Style Designer to modify footer rows for online delivery.

Tables created in Microsoft Word do not overtly identify footer rows. To control footer rows with Microsoft Word and ePublisher, set up the table in your source document to reflect the desired appearance, or use footer paragraph styles in the footer row of the table. Then, ePublisher uses the footer paragraph styles in your source document and you can use the footer **Table Styles** properties in Style Designer to modify the appearance as needed.

## Color

Specifies the background color for the table footer row. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

## Image (Background Image Area)

Specifies the background image for table footer rows. This field lists only the image files located in the `Files` folder of your project. If this field does not list the background image you want, press `F12` and verify the image is in the `Files` folder for your project. Once you copy the image into the `Files` folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

## Tiling (Background Image Area)

Specifies whether the background image repeats in the table footer row. The table must be larger than the background image to see the image repeated in the footer. The background image size also determines if an image repeat shows when set horizontally or vertically. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>No repeat</b>	Specifies that the background image does not repeat.
<b>Repeat</b>	Specifies that the background image repeats.
<b>Repeat horizontally</b>	Specifies that the background image repeats horizontally.
<b>Repeat vertically</b>	Specifies that the background image repeats vertically.

### **Scrolling (Background Image Area)**

Specifies whether the specified image remains in one fixed location as the user scrolls through content. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Scroll</b>	Scrolls the background image as the user scrolls through content.
<b>Fixed</b>	Keeps the background image in a fixed location as the user scrolls through content.

### **Horizontal (Background Image Area)**

Specifies the horizontal position for the background image in your table footer row. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns in the table footer based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-width.
<b>Left</b>	The left side of the background image aligns with the leftmost side of the table footer.
<b>Center</b>	The center of the background image aligns with the center of the table footer.
<b>Right</b>	The right side of the background image aligns with the right of the table footer.

### **Vertical (Background Image Area)**

Specifies the vertical position of the background image in your table footer.  
The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns in the table footer based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-height.
<b>Top</b>	The top of the background image aligns with the top of the tallest element in the table footer.
<b>Center</b>	The center of the background image aligns with the middle of the parent element in the table footer.
<b>Bottom</b>	The bottom of the background image aligns with the lowest element on the table footer.

## Context

Specifies whether to use alternate rows or columns in a table. Alternate shading of rows or columns is a useful layout to help minimize the number of lines and amount of information displayed, while organizing the data in a way that users can easily read and understand.

## Default period

Specifies the number of rows or columns that use the default color in the table footer row. You must first set the default table footer row color in the **Color** field of the **Footer background** properties. Specify a minimum value of **1**. If you specify a value of **0**, ePublisher does not use alternate rows or columns and displays entire body row in the default color.

## Alternate period

Specifies how many consecutive table footer rows or columns use the alternate color. Set this value to a value greater than **0** (zero).

## Alternate Color

Specifies the alternate color for the table footer row. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

## Image (Alternate Info/Background Image Area)

Specifies the alternate background image for table footer rows. This field lists only the image files located in the `Files` folder of your project. If this field does not list the background image you want, press `F12` and verify the image is in the `Files` folder for your project. Once you copy the image into the `Files` folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

## Tiling (Alternate Info/Background Image Area)

Specifies whether to repeat the alternate background image in the table footer rows. The table must be larger than the background image to see the image repeated in the footer. The background image size also determines if an image repeat shows when set horizontally or vertically. The values for this setting are defined as follows:



<b>Value</b>	<b>Description</b>
<b>No repeat</b>	Specifies that the background image does not repeat.
<b>Repeat</b>	Specifies that the background image repeats.
<b>Repeat horizontally</b>	Specifies that the background image repeats horizontally.
<b>Repeat vertically</b>	Specifies that the background image repeats vertically.

### **Scrolling (Alternate Info/Background Image Area)**

Specifies whether the specified image remains in one fixed location as the user scrolls through content. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Scroll</b>	Scrolls the background image as the user scrolls through content.
<b>Fixed</b>	Keeps the background image in a fixed location as the user scrolls through content.

### **Horizontal (Alternate Info/Background Image Area)**

Specifies the horizontal position for the alternate background image in table footer rows. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The alternate background image aligns in the table footer based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-width.
<b>Left</b>	The left side of the alternate background image aligns with the leftmost side of the table footer.
<b>Center</b>	The center of the alternate background image aligns with the center of the table footer.
<b>Right</b>	The right side of the alternate background image aligns with the right of the table footer.

### **Vertical (Alternate Info/Background Image Area)**

Specifies the vertical position of the alternate background image in table footer rows. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns in the table footer based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-height.
<b>Top</b>	The top of the background image aligns with the top of the tallest element in the table footer.
<b>Center</b>	The center of the background image aligns with the middle of the parent element in the table footer.
<b>Bottom</b>	The bottom of the background image aligns with the lowest element on the table footer.

## Header Properties (Tables)

Header rows are rows that contain information that identify the content of a particular column. If the table spans several pages of a print layout, the header row usually repeats itself at the beginning of each new page.

In Microsoft Word and Adobe FrameMaker, you can define a header row for your table. In FrameMaker, you can also define a footer row. Body rows are typically all rows in a table not already defined as a header or footer rows. However, unless you explicitly specify a row as a header row, ePublisher considers all rows except the footer rows as body rows.

You can also specify footer padding properties for tables. **Padding** refers to the space between the table header border and the table header content. You can change the left, top, right, and bottom padding values independently or set all properties to have the same padding. Padding values must be positive numbers.

### Left

Specifies the left padding for a table header, or the space between the left table border and where the content begins in the left of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

## Top

Specifies the top padding for a table header, or the space between the top table border and where the content aligns at the top of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

## Right

Specifies the right padding for a table footer, or the space between the right table border and where the content ends at the right of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

## Bottom

Specifies the bottom padding for a table header, or the space between the bottom table border and where the content ends at the bottom of the table cell. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

## Vertical

Specifies the vertical alignment of the content in a table footer. The values for this setting are defined as: **Top**, **Middle**, and **Bottom**.

## Color

Specifies the border color for a table header. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

## Style

Specifies a border style for a table header, such as a dotted, dashed, or solid line. Not all browsers display border styles the same way. For example, some browsers may not differentiate dotted lines from solid lines. Various browsers and operating systems often display the size and spacing of the dots in a dotted line differently.

## Width

Specifies a border width, such as a thick or thin border. Choose a value or set a custom width by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

# Header Background Properties (Tables)

To help you visualize how you can modify properties, think of the header rows of a table as sitting on a layer above the background of a table. By modifying the color of the header rows, you are applying an opaque color over the color of the background header row to get the desired results. For example, if you cannot make the header row transparent, make sure the background property is properly set for all the different layers in the table.

## Color

Specifies the background color for a table header. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

## Image (Background Image Area)

Specifies the background image for table header rows. This field lists only the image files located in the `Files` folder of your project. If this field does not list the background image you want, press `F12` and verify the image is in the `Files` folder for your project. Once you copy the image into the `Files` folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you

can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

### **Tiling (Background Image Area)**

Specifies whether the background image repeats in table header rows. The table must be larger than the background image to see the image repeated in the header. The background image size also determines if an image repeat shows when set horizontally or vertically. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>No repeat</b>	Specifies that the background image does not repeat.
<b>Repeat</b>	Specifies that the background image repeats.
<b>Repeat horizontally</b>	Specifies that the background image repeats horizontally.
<b>Repeat vertically</b>	Specifies that the background image repeats vertically.

### **Scrolling (Background Image Area)**

Specifies whether the specified image remains in one fixed location as the user scrolls through content. The values for this setting are defined as follows:



<b>Value</b>	<b>Description</b>
<b>Scroll</b>	Scrolls the background image as the user scrolls through content.
<b>Fixed</b>	Keeps the background image in a fixed location as the user scrolls through content.

### **Horizontal (Background Image Area)**

Specifies the horizontal position for the background image in your table header row. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns in the table header based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-width.
<b>Left</b>	The left side of the background image aligns with the leftmost side of the table header.
<b>Center</b>	The center of the background image aligns with the center of the table header.
<b>Right</b>	The right side of the background image aligns with the right of the table header.

### **Vertical (Background Image Area)**

Specifies the vertical position of the background image in your table header row. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The background image aligns in the table header based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-height.
<b>Top</b>	The top of the background image aligns with the top of the tallest element in the table header.
<b>Center</b>	The center of the background image aligns with the middle of the parent element in the table header.
<b>Bottom</b>	The bottom of the background image aligns with the lowest element on the table header.

## Context

Specifies whether to use alternate rows or columns in a table header. Alternate shading of rows or columns is a useful layout to help minimize the number of lines and amount of information displayed, while organizing the data in a way that users can easily read and understand.

## Default period

Specifies the number of rows or columns that use the default color in the table header row. You must first set the default table header row color in the **Color** field of the **Header background** properties. Specify a minimum value of **1**. If you specify a value of **0**, ePublisher does not use alternate rows or columns and displays entire table header row in the default color.

## Alternate period

Specifies how many consecutive table header rows or columns will use the alternate color. Set this value to a value greater than **0**.

## Alternate Color

Specifies the alternate color for the table header row. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

## Image (Alternate Info/Background Image Area)

Specifies the alternate background image you want to use for your table header row. This field lists only the image files located in the `Files` folder of your project. If this field does not list the background image you want, press `F12` and verify the image is in the `Files` folder for your project. Once you copy the image into the `Files` folder, ePublisher displays it in the field list.

ePublisher Designer automatically transforms your source document images during project generation to an optimized version for online distribution. However, you can manually modify your project image settings through the **Graphic Styles** properties in Style Designer. In **Graphic Styles**, you can specify the following image settings: border styles and colors; width, height, and positioning; surrounding space (padding); maximum width or height; size, resolution, and color bit depth; file format and quality level; and transparency or grayscale appearance.

## Tiling (Alternate Info/Background Image Area)

Specifies whether to repeat the alternate background image in table header rows. The table must be larger than the background image to see the image repeated in the header. The background image size also determines if an image repeat shows when set horizontally or vertically. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>No repeat</b>	Specifies that the background image does not repeat.
<b>Repeat</b>	Specifies that the background image repeats.
<b>Repeat horizontally</b>	Specifies that the background image repeats horizontally.
<b>Repeat vertically</b>	Specifies that the background image repeats vertically.

### **Scrolling (Alternate Info/Background Image Area)**

Specifies whether the specified image remains in one fixed location as the user scrolls through content. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Scroll</b>	Scrolls the background image as the user scrolls through content.
<b>Fixed</b>	Keeps the background image in a fixed location as the user scrolls through content.

### **Horizontal (Alternate Info/Background Image Area)**

Specifies the horizontal position of the alternate background image in table header rows. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The alternate background image aligns in the table header based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-width.
<b>Left</b>	The left side of the alternate background image aligns with the leftmost side of the table header.
<b>Center</b>	The center of the alternate background image aligns with the center of the table header.
<b>Right</b>	The right side of the alternate background image aligns with the right of the table header.

### **Vertical (Alternate Info/Background Image Area)**

Specifies the vertical position of the alternate background image in table header rows. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	The alternate background image aligns in the table header based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). For example, if you enter a value of 20 and select the percentage (%) unit of measure, the top left corner is considered 0 percent and the bottom right corner is 100 percent. So, ePublisher aligns the background image in a 20% value of the line-height.
<b>Top</b>	The top of the alternate background image aligns with the top of the tallest element in the table header.
<b>Center</b>	The center of the alternate background image aligns with the middle of the parent element in the table header.
<b>Bottom</b>	The bottom of the alternate background image aligns with the lowest element on the table header.

## HTML (Layout) Properties

Part of the power of ePublisher lies in its ability to produce as many different outputs of your information as you need. This can include everything from print material, to online help files and web content. While online output formats can vary a great deal from one another, most are based on HTML. By modifying styles in Style Designer to include HTML coding properties, you can set up your files to take advantage of web formatting in properties such as paragraphs, characters, tables, images, and page styles. Once you have added HTML properties to your content styles, you can quickly and easily publish content to the web or online.

**Note:** In the **PDF - XSL-FO** format, **HTML** properties are referred to as **Layout** properties. These properties share the names, however FO does not technical support HTML properties, so this tab has the name Layout.

### Tag



Specifies the HTML tag to use for the selected item. By default, ePublisher uses a DIV tag for each paragraph and assigns a class based on the style name. You can change the DIV tags to P tags for specific styles by specifying a P in this property for those paragraph styles.

## **Visibility**

Specifies whether to show or hide the selected item by default. This property allows you to hide a paragraph initially, and then provide a method to show the paragraph.

## **Width**

Specifies the fixed width, including a custom value and the unit of measure, for the selected item. Leave this value blank if you do not want to specify a fixed width for the selected item. For example, if you want your margin settings to control the width of a paragraph, do not specify a width value for your paragraph style.

## **Height**

Specifies the fixed height, including a custom value and the unit of measure, for the selected item. Leave this value blank if you do not want to specify a fixed height for the selected item. For example, if you want the amount of content in a paragraph to control the height of that paragraph, do not specify a height value for your paragraph style.

## **Left (Positioning Area)**

Specifies how far the left edge of the item, such as a paragraph or image, is offset to the right of the left edge of the block that contains the item. Percentages refer to the containing block. This value works with the value selected for the **Type** property.

## **Top (Positioning Area)**

Specifies how far the top edge of the item, such as a paragraph or image, is offset below the top edge of the block that contains the item. Percentages refer to the height of the containing block. This value works with the value selected for the **Type** property.

## **Right (Positioning Area)**

Specifies how far the right edge of the item, such as a paragraph or image, is offset to the left of the right edge of the block that contains the item. Percentages refer to the width of the containing block. This value works with the value selected for the **Type** field.

## **Bottom (Positioning Area)**

Specifies how far the bottom edge of the item, such as a paragraph or image, is offset above the bottom of the block that contains the item. Percentages refer to the height of the containing block. This value works with the value selected for the **Type** field.

## **Type**

Specifies the type of position scheme. This property controls how the browser processes the positioning values of **Left**, **Top**, **Right**, and **Bottom**. The values for this setting are defined as follows:

Value	Description
<b>Static</b>	Places the item, such as a paragraph or image, in the same location as it would normally occupy in the normal flow. <b>Left</b> and <b>Top</b> positioning properties do not apply when you select <b>Static</b> .
<b>Relative</b>	Offsets the position of the item, such as a paragraph or image, from its normal position in the normal flow and ignores other objects positioned outside of the normal flow with the <b>Absolute</b> property.
<b>Absolute</b>	Positions the item, such as a paragraph or image, in a specific location. The item is not considered as part of the normal flow and it is not considered in the layout of objects that follow. <b>Absolute</b> objects do not affect objects that use the <b>Static</b> or <b>Relative</b> value.

## Float

Specifies whether to shift the paragraph to the left or right on the current line and allow content to flow along the side of the paragraph. Use this field to assign images to align on the left or right and allow the text to flow into the open area next to the image. You can also create pull quotes or other sidebars using the **Float** field, as well as create interesting side effects when combined with border styles.

Use of the **Float** property may also be prohibited by the **Clear** property. If the content of the paragraph stretches across the width of the entire page, you may not see any noticeable change unless you assign a smaller width for the paragraph using the **Width** property. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>None</b>	Does not allow other content in the normal flow to float to the right or left of this item.
<b>Left</b>	Aligns this item to the left and allows content in the normal flow to be displayed along the right side.
<b>Right</b>	Aligns this item to the right and allows content in the normal flow to be displayed along the left side.

## **Clear**

Specifies which sides of an item, such as a paragraph or image, may not be adjacent to an earlier floating box. If you set this property on a floating paragraph, it modifies the rules for positioning the float. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>None</b>	Does not constrain the position with respect to floats.
<b>Left</b>	Increases the top margin of the item enough so that the top border edge is below the bottom outer edge of any left-floating boxes that resulted from elements earlier in the source document.
<b>Right</b>	Increases the top margin of the item enough so that the top border edge is below the bottom outer edge of any right-floating boxes that resulted from elements earlier in the source document.
<b>Both</b>	Moves the item below all floating boxes of earlier elements in the source document.

## Display

Specifies the Cascading Style Sheet (CSS) box type, or how to generate CSS boxes for the selected item, such as a paragraph or image. The values for this setting are defined as follows:

Value	Description
<b>None</b>	Does not generate boxes in the formatting structure. When you select this option, the item does not display and the browser treats it as though it does not exist.
<b>Block</b>	Generates a principal block box.
<b>Inline</b>	Uses the layout properties of the previous element.
<b>List item</b>	Displays the item that does not ordinarily display as a list item, such as a bullet or number, as a list item if it is inside a list.

## Z-Index

Specifies an integer that corresponds with the position of the paragraph on the z-axis. Elements with a lower value, such as **1**, are displayed behind other elements with a higher value, such as **2**. You can also specify negative numbers.

## Left (Clipping Area)

Specifies the area of the left side of an image that is visible, when the image is larger than the web element that it sits inside. ePublisher disregards the **Clipping** area fields when the **Overflow** field is set to **Visible**.

## Top Left (Clipping Area)

Specifies the area to the top of an image that is visible, when the image is larger than the web element that it sits inside. ePublisher disregards the **Clipping** area fields when the **Overflow** field is set to **Visible**.

## Right Left (Clipping Area)

Specifies the area to the right side of an image that is visible, when the image is larger than the web element that it sits inside. ePublisher disregards the **Clipping** area fields when the **Overflow** field is set to **Visible**.

## Bottom Left (Clipping Area)

Specifies the area to the bottom of an image that is visible, when the image is larger than the web element that it sits inside. ePublisher disregards the **Clipping** area fields when the **Overflow** field is set to **Visible**.

## **Cursor**

Specifies the type of cursor to be displayed when pointing on an element in your HTML output. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Default</b>	Displays the default cursor.
<b>Auto</b>	Specifies that the browser sets the cursor style.
<b>Crosshair</b>	Specifies that the browser sets the cursor as a crosshair.
<b>Pointer (Hand)</b>	Specifies that the browser sets the cursor as a pointer (hand), which indicates the cursor is over a link.
<b>Move</b>	Specifies that the browser sets the cursor to indicate an element must be moved.
<b>Resize East</b>	Specifies that the browser sets the cursor to indicate an element must be moved to the right (east).
<b>Resize Northeast</b>	Specifies that the browser sets the cursor to indicate an element must be moved up and to the right (northeast).
<b>Resize Northwest</b>	Specifies that the browser sets the cursor to indicate an element must be moved up and to the left (northwest).
<b>Resize North</b>	Specifies that the browser sets the cursor to indicate an element must be moved up (north).
<b>Resize Southeast</b>	Specifies that the browser sets the cursor to indicate an element must be moved down and to the right (southeast).
<b>Resize Southwest</b>	Specifies that the browser sets the cursor to indicate an element must be moved down and to the left (southwest).
<b>Resize South</b>	Specifies that the browser sets the cursor to indicate an element must be moved down (south).



<b>Value</b>	<b>Description</b>
<b>Resize West</b>	Specifies that the browser sets the cursor to indicate an element must be moved left (west).
<b>Text</b>	Specifies that the browser sets the cursor to indicate an element is text.
<b>Wait</b>	Specifies the browser sets the cursor to indicate that the program is busy (and often displays a watch or an hourglass).
<b>Help</b>	Specifies the browser sets the cursor to indicate that help is available (and often displays a question mark or balloon).

## Overflow

Specifies how the output displays if the content of an element overflow its area. If you set the **Overflow** field to **Visible**, any Clipping area properties you set are disregarded. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Auto</b>	Indicates the browser displays a scroll bar for viewing content that overflows its area.
<b>Visible</b>	Indicates the browser does not clip the image or content that overflows its area. Instead, it displays the content outside the element.
<b>Hidden</b>	Indicates the browser clips the content that overflows its area, but does not display a scroll-bar for viewing the rest of the content.
<b>Scroll</b>	Indicates the browser clips the content that overflows its area, but displays a scroll-bar for viewing the rest of the content.

## Margin Properties

You can alter the white space around a paragraph by adjusting the margin and the padding. In terms of the CSS box model, modifying the margin properties adjusts the space outside the border area. For example, if you create a border or background color for a paragraph and you increase the size of the margins around the paragraph, the border remains the same distance from the text in the paragraph. However, the position of the paragraph changes because there is more white space between the modified paragraph and the other elements on the page. You can set margin properties for paragraphs, characters, tables, images, and pages from Style Designer.

### Left

Specifies the amount of space to the left of a border area. Set a custom margin value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select **Auto**, ePublisher automatically sets the margin value based on the settings already indicated in the project.

### Top

Specifies the amount of space above a border area. Set a custom margin value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select **Auto**, ePublisher automatically sets the margin value based on the settings already indicated in the project.

## Right

Specifies the amount of space to the right of a border area. Set a custom margin value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select **Auto**, ePublisher automatically sets the margin value based on the settings already indicated in the project.

## Bottom

Specifies the amount of space below a border area. Set a custom margin value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select **Auto**, ePublisher automatically sets the margin value based on the settings already indicated in the project.

# Markdown++ Properties

You can specify the markdown syntax used for your paragraph and character styles when using ePublisher to generate Markdown++ output. By default, your project will try to use the most logical syntax, but with the Style Designer you can control the syntax on a per style basis.

## Indentation Level

By default, ePublisher will generate all of your paragraphs as if they have no indentation. However, if you want your markdown syntax to reflect indentation as if it were originally authored in markdown, then you can use this property to control your unstructured paragraph's indentation level. Using this property has the effect of injecting light-weight structure into your original content. For example, if you have a paragraph that is designed to be a bulleted (unordered) list that appears indented from the previous paragraph, you would want to set this paragraph's **Indentation Level** to the value: **1**.

Value	Description
<b>None</b>	No markdown indentation will be created in the output.
$n$	<p>ePublisher will determine how many levels to create for this paragraph using the value <math>n</math> (1 or higher) as the preferred indentation level. Used in conjunction with paragraphs that use Unordered List, Ordered List, or Blockquote syntax.</p> <p><b>Note:</b> ePublisher will not indent the paragraph by more levels than the previous level plus one. This will prevent invalid markdown structure from being generated.</p>

## Syntax

By default, this property will set its value based on what it knows about your source content, which is not really that much, but in cases where it is very clear that the syntax should be a heading, list, or code fence paragraph, the default will generate a good match. However, in other cases you may want to set this property to **None**, which generates a normal paragraph. It is recommended that you always set this property explicitly rather than rely upon the default (**Auto-Detect**).

Value	Description
<b>Auto-Detect</b>	<p>Most often will set your paragraph to use no syntax (<b>None</b>). However, in cases where the paragraph is clearly a heading, then it will set to a <b>Heading 1</b> through <b>Heading 6</b> syntax. It can also detect <b>Ordered List</b> and <b>Unordered List</b>. And can sometimes detect when to use <b>Code Fence</b>.</p> <p><b>Note:</b> It is recommended to not rely on this property value for your production work. Use the <b>None</b> value if your not sure what syntax to use.</p>
<b>Title 1</b>	<p>Highest level heading structure.</p> <pre>Title Text =====</pre>
<b>Title 2</b>	<p>Alternate heading structure for: <b>Title 1</b></p> <pre>Title Text -----</pre>
<b>Heading 1</b>	<p>Most common heading for starting topics.</p> <pre># Heading Text</pre>
<b>Heading 2</b>	<p>Heading structure with varying level of significant from 2 down to 6.</p> <pre>## Heading Text</pre>
<b>Heading 3</b>	<p>Heading structure with varying level of significant from 2 down to 6.</p> <pre>### Heading Text</pre>
<b>Heading 4</b>	<p>Heading structure with varying level of significant from 2 down to 6.</p> <pre>#### Heading Text</pre>
<b>Heading 5</b>	<p>Heading structure with varying level of significant from 2 down to 6.</p> <pre>##### Heading Text</pre>

Value	Description
<b>Heading 6</b>	Heading structure with varying level of significant from 2 down to 6. ##### Heading Text
<b>Unordered List</b>	Used for bulleted list paragraphs. Use in conjunction with the <b>Indentation Level</b> property. - unordered list
<b>Ordered List</b>	Used for numbered list paragraphs. Use in conjunction with the <b>Indentation Level</b> property. 1. ordered list
<b>Blockquote</b>	Used for paragraphs that are offset or discontinuous from the current flow of paragraphs. Use in conjunction with the <b>Indentation Level</b> property. > blockquote
<b>Code Fence</b>	Used to create a paragraph with code syntax. Useful for blocks of text that need to embed special characters and fixed width character spacing. ```\n\ncode text embedded\n```\n
<b>None</b>	Creates a paragraph without any syntax which most commonly represents normal or body content. Will create an empty line between consecutive paragraphs. Normal paragraph text followed by a blank line.

## Master Page Properties (Pages)

You can control the layout of all your generated pages for output formats that require handling of pagination (i.e. *PDF - XSL-FO* output format). For detailed information on these master page properties see: [http://www.w3.org/TR/xsl11/#fo\\_simple-page-master](http://www.w3.org/TR/xsl11/#fo_simple-page-master).

## Even Master Page

This tab allows you to specify the main layout properties of all *even* pages in your generated output. You can modify parameters such as the overall page size, number of columns, size of the column gap, and the margin with this set of properties.

### Even Master Page: After Region

This tab allows you to specify the layout properties for the FO defined `region-after` area of all *even* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Even Master Page: Before Region

This tab allows you to specify the layout properties for the FO defined `region-before` area of all *even* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Even Master Page: Body Region

This tab allows you to specify the layout properties for the FO defined `region-body` area of all *even* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Even Master Page: End Region

This tab allows you to specify the layout properties for the FO defined `region-end` area of all *even* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Even Master Page: Start Region

This tab allows you to specify the layout properties for the FO defined `region-start` area of all *even* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

## Odd Master Page

This tab allows you to specify the main layout properties of the first page and all *odd* pages in your generated output. You can modify parameters such as the overall page size, number of columns, size of the column gap, and the margin with this set of properties.

### Odd Master Page: After Region

This tab allows you to specify the layout properties for the FO defined `region-after` area of all *odd* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Odd Master Page: Before Region

This tab allows you to specify the layout properties for the FO defined `region-before` area of all *odd* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Odd Master Page: Body Region

This tab allows you to specify the layout properties for the FO defined `region-body` area of all *odd* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Odd Master Page: End Region

This tab allows you to specify the layout properties for the FO defined `region-end` area of all *odd* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

### Odd Master Page: Start Region

This tab allows you to specify the layout properties for the FO defined `region-start` area of all *odd* pages in your generated output. You can modify parameters such as the background, clipping region, alignment, extent, overflow behavior, reference orientation, and writing mode.

## Navigation Properties (Pages)

Effective online documentation requires extra navigation features over print documentation. In addition, online output follows different rules regarding the number of page breaks and includes unique page layouts for specific information. From Style Designer, you can specify navigation details for content pages in **Page Styles** properties.

If you include navigation buttons in your online output, ePublisher lets you specify several options for where the navigation features display. Keep in mind that not all output formats support navigation links.



You can form a linked path to show users the location of the current topic in your online content. This clickable path, called **breadcrumbs**, steps you through the topics that are responsible for getting you to the topic being viewed. Breadcrumbs can display at the top of the page, at the bottom of the page, or both. ePublisher sets the breadcrumb trail to the top of the output page by default.

### Top (Navigation Alignment)

Specifies how to align the top navigation browse buttons in your online output, such as **Left**, **Center** or **Right**. To display navigation browse buttons at the top of the page, click the **Options** tab and set the **Navigation links shown at top of page** option to **Enabled** for this page style. Then, set the **Navigation Properties** from the **Properties** tab in **Page Styles**.

### Bottom (Navigation Alignment)

Specifies how to align the bottom navigation browse buttons in your online output, such as **Left**, **Center** or **Right**. To display navigation browse buttons at the bottom of the page, click the **Options** tab and set the **Navigation links shown at top of page** option to **Enabled** for this page style. Then, set the **Navigation Properties** from the **Properties** tab in **Page Styles**.

### Separator

Specifies the separator characters to use between breadcrumb items. The default character is a colon (:).

### Top (Breadcrumb Alignment)

Specifies how to align the breadcrumbs at the top of the page. To display breadcrumbs at the top of the page, you must also set the **Breadcrumbs shown at top of page** option on the **Options** tab to **Enabled** for this page style.

### Bottom (Breadcrumb Alignment)

Specifies how to align the breadcrumbs at the bottom of the page. To display breadcrumbs at the bottom of the page, you must also set the **Breadcrumbs shown at bottom of page** option on the **Options** tab to **Enabled** for this page style.

## Padding Properties

You can adjust the white space around a paragraph by altering the margin and the padding values. **Padding** refers to the space between the border and the content. You can change the **Left**, **Top**, **Right**, and **Bottom** padding values independently. Padding values must be positive numbers.

In terms of the CSS box model, modifying the padding properties adjusts the space inside the border area. For example, if you create a border or background color for a paragraph and you increase the size of the padding, the border moves away from the text in the paragraph.

You can set padding properties for paragraphs, characters, pages, tables, and graphic styles in Style Designer. Refer to “Table Properties (Tables)” for details on how to set padding properties in a table.

## Left

Specifies the left padding, or the space between the left border and the left content margin. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select **Auto**, ePublisher automatically sets the padding value based on the settings already indicated in the project.

## Top

Specifies the top padding, or the space between the top border and where the content aligns at the top of the page. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select **Auto**, ePublisher automatically sets the padding value based on the settings already indicated in the project.

## Right

Specifies the right padding, or the space between the right border and the right content margin. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select **Auto**, ePublisher automatically sets the padding value based on the settings already indicated in the project.

## Bottom

Specifies the bottom padding, or the space between the bottom border and where the content ends at the bottom of the page. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). If you select

**Auto**, ePubublisher automatically sets the padding value based on the settings already indicated in the project.

## Pagination Properties

Available only in the *PDF - XSL-FO* format.

These properties are for use with output formats that require print related pagination characteristics.

### Force Page Count

Forces the generated pages to end in a certain manner. The values for this property are defined as follows:

<b>Value</b>	<b>Description</b>
Auto	Force the last page in this page-sequence to be an odd-page if the initial-page-number of the next page-sequence is even. Force it to be an even-page if the initial-page-number of the next page-sequence is odd. If there is no next page-sequence or if the value of its initial-page-number is "auto" do not force any page.
Even	Force an even number of pages in this page-sequence.
Odd	Force an odd number of pages in this page-sequence.
End on Even	Force the last page in this page-sequence to be an even-page.
End on Odd	Force the last page in this page-sequence to be an odd-page.
Do Not Force	Do not force either an even or an odd number of pages in this page-sequence.
Inherit	Use the value of the parent page style's page property of the same name.

## **Initial Page Number**

Specifies the initial folio number to be used in this page-sequence. The values for this property are defined as follows:

<b>Value</b>	<b>Description</b>
Auto	The initial folio number shall be set to 1 if no previous page sequence exists in the document. If a preceding page-sequence exists, the initial folio number will be one greater than the last number for that sequence.
Next Odd Page	A value is determined in the same manner as for <b>Auto</b> . If that value is an even number 1 is added.
Next Even Page	A value is determined in the same manner as for <b>Auto</b> . If that value is an odd number 1 is added.
n	A positive integer.

## Page Number Format

Specifies the page numbering style to use for the generated page numbers in the output. For detailed information on page numbering see: <https://www.w3.org/TR/xslt#number>. The values for this property are defined as follows:

Value	Description
1	Specify a numbering sequence that starts with this token. If an implementation does not support a numbering sequence that starts with that token, it must use a format token of 1.
a	Use lowercase alphabetic sequences for the page numbering.
A	Use uppercase alphabetic sequences for the page numbering.
i	Use lowercase roman numeral page numbering.
I	Use uppercase roman numeral page numbering.

### Page Number Letter Value

Specifies characteristics for numbering sequences that use letter values. For detailed information on page numbering see: <https://www.w3.org/TR/xslt#number>. The values for this property are defined as follows:

Value	Description
Auto	Use the default behavior of the implementation.
Alphabetic	Use letter values based on their alphabetic sequence.
Traditional	Use letter values based on what is traditional for the language that is being generated.

## Table Properties (Tables)

ePublisher gives you the power to adjust padding, spacing, alignment and layout properties for your content tables. **Padding** refers to the space between the table cell border and the table cell content. You can change left, top, right, and bottom padding values independently. Padding values must be positive numbers.

### Left

Specifies the left padding for the entire table. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

### Top

Specifies the top padding the entire table. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

### Right

Specifies the right padding for the entire table. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

### Bottom

Specifies the bottom padding for the entire table. Set a custom padding value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

## **Border**

Specifies the size of the border for a table. **Borders** are lines that you can draw around any or all of the four sides of a table. This field sets the border for the outer edge of the entire table, not the border of cells within the table. This is an HTML property, so you do not need to specify a unit of measure. If you do not specify a border attribute, ePublisher displays the table without any borders.

## **Cell spacing**

Specifies the amount of space between cells of a table. Use cell spacing to create a transparent space, or increase the distance between table cells. This is an HTML property, so you do not need to specify a unit of measure.

## **Cell padding**

Specifies the amount of space between the cell content and its borders. This is an HTML property, so you do not need to specify a unit of measure.

## **Border collapse**

Specifies whether table borders collapse into a single border or remain detached as with standard HTML. The values for this setting are defined as follows:



<b>Value</b>	<b>Description</b>
<b>Collapse</b>	Collapses borders into a single border.
<b>Separate</b>	Keeps borders detached.

### **Border spacing**

Specifies the distance between the borders of adjacent cells. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). Border spacing values cannot be negative numbers.

### **Caption side**

Specifies the position of the table caption in relation to the table. The values for this setting are defined as: **Top** and **Bottom**.

### **Empty cells**

Specifies whether to display empty table cells or to hide them. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>hide</b>	Indicates no borders are drawn around empty table cells.
<b>show</b>	Indicates borders are drawn around empty cells.

### **Table layout**

Specifies whether the table layout is automatic or fixed. The values for this setting are defined as follows:

Value	Description
<b>Auto</b>	Specifies the browser sets the column width of a table by the contents of the table cells. This setting can slow down browser display time because the browser must access all table content before it can determine the final layout.
<b>Fixed</b>	Specifies the browser displays the table layout using settings for table width and the width of columns, not based on the content in the table cells. By setting a fixed table layout, the browser can quickly display the table once it receives the first row of table content.

### Horizontal

Specifies the horizontal alignment of the content in a table. The values for this settings are defined as: **Left**, **Center**, **Right**, and **Justify**.

### Vertical

Specifies the vertical alignment of the content in a table. The values for this setting are defined as: **Top**, **Middle**, and **Bottom**.

## Text Properties

### Color

Specifies the color for the text. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

### Transform

Specifies whether to display the text in a specific way regardless of how the text is in the source documents. You can transform the text to capitalize the first letter of each word, all upper case letters, or all lower case letters. To leave the text as it is in the source documents, select **None**.

### White space

Specifies how the browser displays white space, or extra spaces, within text. The values for this setting are defined as follows:

Value	Description
<b>Normal</b>	Specifies the browser ignores extra white space.
<b>Pre</b>	Specifies the browser preserves all white space. This setting indicates the browser acts as it does with the <code>&lt;pre&gt;</code> tag in HTML.
<b>No wrap</b>	Specifies the browser does not wrap text. The browser displays text on the same line until it encounters a <code>&lt;br&gt;</code> tag.

## Word spacing

Specifies the amount of space between words. This field allows you to adjust the kerning between words and defaults to **Normal** spacing. Set a custom spacing value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). You can adjust the spacing within words with the **Letter spacing** field.

## Letter spacing

Specifies the amount of space between letters within a word. This field allows you to adjust the kerning within words and defaults to **Normal** spacing. Set a custom spacing value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px). You can adjust the spacing between words with the **Word spacing** field.

## Line height

Specifies the amount of space between each line within a paragraph. This spacing is also known as **letting or line height**. This field defaults to **Normal** spacing between lines. Set a custom height value by selecting **Custom**, entering a value, and selecting the unit of measure. Units of measure you can select include: percentages (%) of the line height, centimeters (cm), em units (em), x-height (ex), inches (in), millimeters (mm), picas (pc), point sizes (pt), and pixels (px).

## Underline style

Specifies whether to display a line underneath a character or the text of a paragraph. The underline may go through the bottoms of letters that drop below the main letter line, such as lowercase g, p, and q, which can make reading the text more difficult.

### **Underline color**

Specifies the color of the line you choose to display underneath a character or the text of a paragraph. Select a color from the list or type the RGB value of the color, such as `FFFFFF`. **RGB value** refers to the web standard hexadecimal notation for Red, Green, and Blue color values. RGB values can range from the lowest RGB value of `0` (hex `#00`) to the highest value of `255` (hex `#FF`). For example, the RGB hexadecimal notation for black is `000000` and white is `FFFFFF`.

### **Overline style**

Specifies whether to display a line above a character or the text of a paragraph.

### **Line through style**

Specifies whether to display a line through the middle of a character or the text of a paragraph, also called a strike-through line.

### **Blink**

Specifies whether the character or the text of a paragraph blinks or flashes. Keep in mind, blinking text can be difficult to read and distracting on a page, so use this feature sparingly.

### **Horizontal**

Specifies the horizontal position for the character or the text of a paragraph. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Custom</b>	Aligns the character or paragraph based on the custom value entered and the unit of measure selected. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px).
<b>Left</b>	Aligns the left side of a paragraph with the left margin of the layout.
<b>Center</b>	Aligns the center of a paragraph with the center of the layout.
<b>Right</b>	Aligns the right side of the paragraph with the far right margin of the layout.
<b>Justify</b>	Aligns a paragraph of text along both the left and right margins. In justified text, the spaces between words and letters (kerning) is stretched or sometimes compressed to make the text align with the left and right margins.

## Vertical

Specifies the vertical alignment for a character or the text of a paragraph. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Baseline</b>	Aligns the paragraph or character on the baseline of the text line.
<b>Sub</b>	Aligns the paragraph or character as subscript.
<b>Super</b>	Aligns the paragraph or character as superscript.
<b>Top</b>	Aligns the paragraph or character with the top of the tallest item on the line.
<b>Text Top</b>	Aligns the paragraph or character with the top of the text line.
<b>Middle</b>	Aligns the paragraph or character with the middle of text line.
<b>Bottom</b>	Aligns the paragraph or character with the bottom of the lowest item on the line.
<b>Text Bottom</b>	Aligns the paragraph or character with the bottom of the text font.

## Indent

Specifies a custom indentation value for a paragraph. Enter the indentation value and select the unit of measure. Units of measure you can select include: percentage (%) of the line height, centimeters (cm), em unit (em), x-height (ex), inches (in), millimeters (mm), pica (pc), point sizes (pt), and pixels (px).

## Direction

Specifies the direction the text should be read. This field works with the **Unicode Bidi** field to ensure the browser correctly displays your content. If unspecified, ePublisher sets the text direction as left-to-right by default.

If a document contains right-to-left characters, and if the browser can display the language with the proper character set, the browser must apply the



bidirectional algorithm. If you prefer to control the handling of a particular phrase, you can apply a character style to that phrase and then define the character style with the **Direction** and **Unicode Bidi** fields.

## Unicode Bidi

Specifies the bidirectional aspect to use when there are some characters in the text within a single paragraph that can be read from left to right, while other text within the paragraph can be read from right to left. In some documents, such as those written with the Arabic or Hebrew script, and in some mixed-language contexts, text within a single paragraph may appear with mixed directionality. This phenomenon is called **bidirectionality**, or **bidi** for short.

If a document contains right-to-left characters, and if the browser is able to display the language with the proper character set, the browser must apply the bidirectional algorithm. The proper character set means to not display arbitrary substitutes such as a question mark, a hex code, or a black box for some characters,

This property allows you to display text within a single paragraph with mixed directionality. The values for this property are defined as follows:

Value	Description
<b>Normal</b>	Allows implicit use of bidirectional.
<b>Embed</b>	Allows the Unicode-bidi algorithm to choose when it is appropriate to use the value specified for the <b>Direction</b> property. Only the specific text that is read in a different direction will be modified. The rest of the paragraph remains in its default direction.
<b>Bidi override</b>	Forces the characters to be displayed using the value specified for the <b>Direction</b> property. All text in the paragraph is modified to reflect a different direction, completely overriding any instructions to display the text in the default direction.

## Markdown++ Options

These options allow you to extend the publishing capabilities of your generated markdown output without impacting the interoperability of the generated markdown content. Enabling Markdown++ options in your conversion process will maximize your content's publishing capabilities, while not effecting its ability to be used with a wide range of markdown tools and systems, including other publishing systems that are not Markdown++ aware.

### Markdown++ anchor

Default: **Enabled**.

Specifies whether the selected style includes a unique anchor value that can be used for linking from other content. This option is currently only available for paragraph styles.

### Markdown++ markers

Default: **Enabled**.

Specifies whether the selected style includes markers from the original source content and emitted as Markdown++ markers.

### Markdown++ style name

Default: Enabled.

Specifies whether the selected style includes the style name from the original source content.

## Table rendering

By default, ePublisher will generate all of your tables using multi-line markdown pipes tables. These are recommended for tables that require preserving content structure and high fidelity styling. When using **Pipes Multiline** tables with non-Markdown++ compatible tools, your tables will downgrade to traditional markdown pipes tables, where each line in the original content becomes a row.

Value	Description
<b>Pipes Multiline</b>	Markdown++ highest quality table, preserving both structure and style of your original source content's table. Compatible with traditional markdown tools and systems, but quality will be degraded and the table will appear to have more rows than in the original source content.
<b>Pipes</b>	Generates a traditional markdown pipe table with concise syntax that works well for numbers and basic content. Not recommended for complex tables.

## Paragraph Styles Options

ePublisher provides many options to allow you to customize your content transformation process and implement the online features you need. For example, the options define table of contents levels, popup windows, and related topics, as well as many other features and behaviors.

### Additional CSS classes

Specifies an additional class name to include in the output class name for the selected style. This option is helpful if you are using a custom `.css` file. ePublisher supports this option only in output formats that support cascading style sheets. This option is available all styles except marker styles.

This option lets you specify other classes to add to the class statement in the generated output for the selected style. For example, if you have a class in your `.css` file called `Red` that defines the text color as red, you can specify `Red` in the **Additional CSS classes** option for each style you want to have red text. ePublisher adds this class to the class name in the generated output, such as `classname="sourcestylename Red"`.

### Apply character styles

Specifies whether ePublisher uses the character style formatting within the selected paragraph style. Character styles specify characteristics like font size and style for one or more characters. When you set the **Apply character styles** option to **Enabled**, ePublisher applies character formatting to the content within the selected paragraph style in your source documents. This option is available only for paragraph styles.

## Citation

Specifies that ePublisher recognizes and formats the text with this style as quotation content. A **citation** is a reference or footnote to a book, article, or other material that specifies the source from which a quotation was borrowed. The citation for a quote enables users to go to a Web site that contains more information about the quote.

You must define a Citation paragraph or character format and a Citation marker to specify a URL citation. The paragraph or character format with this option enabled identifies the quote in the generated output. The Citation marker defines the URL for the Web site with more information. When you hover over the quote in the output file, a popup window displays the URL. This option is available only for paragraph and character styles.

## Dropdown

Specifies whether to use the selected paragraph style to define an expand/collapse section, also called a dropdown hotspot. Some output formats do not support expand/collapse sections.

You can configure the expand/collapse section to initially be hidden or shown. When the user clicks the dropdown hotspot, the content expands and collapses to allow the user to view more or less information. The expand/collapse content can include multiple paragraph styles. You can use the DropDownEnd marker to identify the end of the expand/collapse content. If you have a specific paragraph style that always serves as the ending point for your expand/collapse content and is not included in the expand/collapse content, set this option for that paragraph style to **Break**. By default, all paragraph styles are included in the expand/collapse content. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Break</b>	Specifies the paragraph style is never included in expand/collapse content. A paragraph style with this option set to <b>Break</b> ends the expand/collapse content section.
<b>Continue</b>	Specifies the selected paragraph style does not change the expand/collapse content output. If the selected paragraph style is within an expand/collapse content section, the <b>Continue</b> value directs ePublisher to include the content as part of the expand/collapse content. This value is the default for paragraph styles.
<b>End</b>	Allows the content to be inside the dropdown selection but it will signify the end of the dropdown similar to the behavior of a marker.
<b>Start open</b>	Displays the content in the expand/collapse section when the page is initially displayed. The paragraph with this option set provides the hotspot where the user can click to show or hide the expand/collapse content. This paragraph style starts the expand/collapse section.
<b>Start closed</b>	Hides the content in the expand/collapse section when the page is initially displayed. The paragraph with this option set provides the hotspot where the user can click to show or hide the expand/collapse content. This paragraph style starts the expand/collapse section.

### Element name

Specifies the name of the element that contains the content of the associated style when ePublisher converts it to XML output. This option is available only for paragraph styles and graphic styles for the XML+XSL output format.

### Generate output

Specifies whether the selected style is included in the generated output. This option is available for all styles except marker styles. By default, the **Generate output** option is set to **Enabled** so that all content is included in

the output. If you select **Disabled**, the content with the selected style does not appear in the generated output.

### **Glossary behavior**

Specifies whether ePublisher uses the paragraph style to format glossary terms and definitions displayed on the Glossary tab in Sun JavaHelp. This option is available only for the Sun JavaHelp output format.

### **Keep empty paragraphs**

Specifies whether ePublisher generates output for paragraphs with the selected style that do not have any content. For example, at the end of chapters in Microsoft Word, you may have multiple blank paragraphs to create the footer on a blank even page. To exclude the empty paragraphs from your generated content, select the paragraph style of the blank paragraphs in Style Designer and set the **Keep empty paragraphs** option to **Disabled**. If you leave this option set to its default value of **Enabled**, ePublisher can display blank lines in your generated output.

### **Keep paragraph numbering**

Specifies whether ePublisher includes the auto-generated numbers for the selected paragraph style in your source documents. For example, chapter titles, headings, figure captions, and table captions may have auto-numbering in your source documents that you do not want to include in your online content. To exclude the autonumbering, select the paragraph style that contains autonumbering and set the **Keep paragraph numbering** option to **Disabled**.

### **Keep paragraph numbering in TOC**

Specifies whether ePublisher includes the auto-generated numbers for the selected paragraph style in your source documents that have been defined to have a Table of Contents level by the Paragraph Style's Options. To exclude the autonumbering, select the paragraph style that contains autonumbering and set the **Keep paragraph numbering in TOC** option to **Disabled**.

### **Mini-TOC levels**

Specifies whether to include a miniature table of contents (mini-TOC), also known as a partial table of contents, and what table of contents levels to include in the mini-TOC. A **mini-TOC** provides a small-scale look at the topics in the upcoming section. The topic titles are displayed as links beneath the current topic heading for easier navigation within the section.

ePublisher does not create mini-TOCs by default. To create a mini-TOC following a paragraph style, set this option to a value other than **None**. ePublisher generates the mini-TOC on all pages on which the selected paragraph style occurs. ePublisher creates the mini-TOC for all styles with TOC

level settings in the specified range between the selected paragraph style of the current topic and the next occurrence of this same style. The values for this setting are defined as follows:



Value	Description
<b>All</b>	Creates a mini-TOC for all TOC levels up to the next occurrence of the current paragraph style and displays the mini-TOC on all output pages on which this paragraph style occurs.
<b>None</b>	Does not create a mini-TOC.
<i>n</i>	Creates a mini-TOC for all TOC levels from 1 through the number specified (where <i>n</i> equals any numeric value), up to the next occurrence of the current paragraph style. The mini-TOC is displayed on all output pages on which this paragraph style occurs. For example, if you specify <b>1</b> , ePublisher creates a mini-TOC for all TOC levels 1, up to the next occurrence of the current paragraph style. If you specify <b>4</b> , ePublisher creates a mini-TOC for all TOC levels 1 through 4, up to the next occurrence of the current paragraph style.

For more information, see “Defining TOCs and Mini-TOCs”.

### Page break priority

Used in HTML-based output formats. This option specifies how to handle page breaks for the selected paragraph style. This option specifies whether to create a new page starting with the selected paragraph style and allows you to avoid new pages with a heading and no content if two headings occur in a row in your source documents. The default value is `Use table of contents level`, which only creates a new page when there is a **Table of contents level**.

Value	Description
<b>Use table of contents level</b>	Uses the value as determined by this paragraph style's <b>Table of contents level</b> value. Keep in mind that if that value is set to Auto-Detect, then this value will be based on the actual document's setting.
<b>None</b>	Does not create a page break under any condition.
<i>n</i>	Creates a page break so long as the previous paragraph has a <b>Page break priority</b> value that is numerically lower or that is set to <code>None</code> .

This option works with the **Page break handling** format setting to give you complete control for how ePublisher handles page breaks. To access the **Page break handling** format setting, click **Target Settings** on the **Target** menu. With the **Page break handling** setting value set to `Combine`, to create a new page every time the selected paragraph style exists, select the highest priority level of **1**. To create a new page only if the preceding paragraph did not create a new page, select a lower priority level of **2** through **5**. For more information, see "Defining New Pages (Page Breaks)".

## Pass Through

Specifies whether ePublisher should process the content with the selected paragraph style applied or insert the content directly into your output without being transformed and coded for your output format. By default this option is disabled. If you enable this option, ePublisher does not process the content when generating output. For example, you can use this option to embed HTML code into your generated output.

## Popup

Specifies whether a paragraph style is displayed in a popup window and how the content is displayed in your generated output. Define a paragraph style for the first paragraph to display in a popup window. If the popup window contains more than one paragraph of content, create a second paragraph style and apply it to all subsequent paragraphs that follow the first paragraph in the popup window. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>None</b>	Does not create a popup window.
<b>Define</b>	Specifies the first paragraph of a popup window. The content of the selected paragraph is also displayed in the generated output file. Select this option only if you apply the paragraph style to the first paragraph of popup content and you want the content to be displayed in the topic as well as in a popup window.
<b>Define with no output</b>	Specifies the first paragraph of a popup window. Use this option to display the text only in a popup window. The content is not included in the topic where the content is defined. Select this option only if you apply the paragraph style to the first paragraph of popup content and you want the content to be displayed only in a popup window.
<b>Append</b>	Specifies additional paragraphs to include in a popup window that you already defined by a paragraph style with the <b>Popup</b> option set to <b>Define</b> .
<b>Append with no output</b>	Specifies additional paragraphs to include in a popup window that you already defined by a paragraph style with the <b>Popup</b> option set to <b>Define with no output</b> .

For more information, see “Defining Popup Windows”.

### **Popup page style**

Specifies an alternate page style for the popup window associated with the selected paragraph style. ePublisher creates default popup windows with the same appearance as all other topic pages, including breadcrumbs and company information. You can define additional page styles to use for popup windows. To assign a page style other than the default page style to popup windows, you must first create the new page style in Style Designer. Then, select the paragraph style you use to designate popup window content and select the new page style in the **Popup page style** option.

If you are using marker styles to create popups, you cannot use page styles to control the appearance of popup windows. This method applies only to popup windows created with paragraph styles.

### **Related topic**

Specifies whether the selected paragraph style provides links to topics that are similar to the current topic or may be of additional interest to the user.

**Related topic** links are essentially cross-references to other corresponding topics within your output. Instead of displaying cross-references directly in your output, you can create a list of Related Topics links. The values for this setting are defined as follows:

Value	Description
<b>None</b>	Does not include the paragraph style in related topics.
<b>Define</b>	Displays the related topics link both within the body of the topic and when the user clicks the <b>Related Topics</b> button.
<b>Define with no output</b>	Displays the related topics link only when the user clicks the <b>Related Topics</b> button.

To create a **Related Topics** button and have the Related Topics links displayed in the output, create a paragraph style in the source documents that you apply to each related topic link. Then, create a list of Related Topics links for each topic. For more information, see “Defining Related Topics”.

## Search relevance weight

Reverb allows you to add a **Search relevance weight** to your **Paragraph Styles**, giving you the opportunity to control the order of the Search Results, assigning to every word with that paragraph style the specified weight and then saved in the index file. The default value is 1.

The predefined values for this setting are defined as follows (but you can define any integer value): **Ignore**, **1**, **10**, **20**, **50** and **100**. If you choose to **Ignore** it (which is going to be 0), it means that the style is not going to be shown in your results.

## See Also

Defines See Also link functionality for the selected paragraph or character style. **See Also** links are associative links, also called Alinks, that identify other topics that may be of interest to the user of the current topic. These links use internal identifiers to define links and the browser builds the list dynamically based on the topics available when the user displays the links.

To create a See Also link as inline text without a button, create a unique character style and select the **See Also** option for that style.

To use a button to display See Also links, create a unique paragraph style, select the **See Also** option for that style, and type the See Also text on that paragraph. The paragraph style properties selected in Style Designer affect the text of the **See Also** button.

You must also define SeeAlsoLink and SeeAlsoKeyword markers. HTML Help also supports the SeeAlsoLinkDisplayType and SeeAlsoLinkWindowType markers. These markers allow you to change how the See Also links are displayed in HTML Help. See Also links are not supported by all output formats, such as Dynamic HTML. For more information, see “Defining See Also Links”.

### Start new page sequence

Used in the **PDF - XSL-FO** output format. By default this option is disabled, but if enabled, it creates a new page sequence to be generated in the PDF output. When used in conjunction with a **PageStyle** marker, you can change the generated page layout properties to match the configuration of the Page style in use.

### Table of contents level

Specifies the level of the table of contents (TOC) entry for the selected paragraph style. For example, setting this option to **1** specifies that all the paragraphs with the selected style appear in the TOC at the first level in the TOC hierarchy. Setting this option to **2** specifies that all paragraphs with the selected style appear in the table of contents at the second level. The **Table of contents level** option works in conjunction with the **Page break priority** option.

The `Auto-Detect` option takes the source file’s information and will automatically assign values for the paragraph. From Adobe FrameMaker, this will be taken from the Bookmarks tab in the PDF Setup from the Print Document dialog box. From Microsoft Word, ePublisher uses the Heading levels (usually from Heading 1-Heading 9). From DITA, the map hierarchy becomes the TOC.

Value	Description
<b>Auto-Detect</b>	Uses the value as determined by the original content's TOC setting.
<b>None</b>	Does not create a TOC entry under any condition.
<i>n</i>	Creates a TOC entry in the online help system with this explicit value.

The value you specify for each of your paragraph styles determines the structure of the online version of your table of contents. These values also define the structure of mini-TOCs. ePublisher defaults this value to `Auto-Detect`, which typically results in your online table of contents looking the same as your document's table of contents. This option is available only for paragraph styles. For more information, see "Defining TOCs and Mini-TOCs" S.

## Variable

Specifies the name for a variable that ePublisher creates. ePublisher sets the value of the variable to the contents of the selected paragraph or character style. You can then use this variable in your page layout override `Page.asp` file.

For example, if you want to include a section name in a banner area at the top of each topic in that section, you can assign a variable for the paragraph style of the section name. Then, you can modify the override `Page.asp` file to include this variable value in the page layout where you want it. Each topic in a section use the section name for that section, until the next section name defined by the paragraph style changes the value for the next section of topics. To insert a variable in the override `Page.asp` file, insert the following code in the file, where *variable* is the name of the variable you specify in this option:

```
<div wwpage:content="wwvars:variable">replaced variable value</div>
```

After processing the `Page.asp` file for a topic where the variable value is `User Information`, ePublisher replaces this code with the following code in the output file:

```
<div>User Information</div>
```

## **'What is this' marker**

Available only in the Microsoft HTML Help 1.x format. This option specifies whether the selected paragraph style should define What's This help content for field-level context-sensitive help. When the user right-clicks on a field in the user interface and then clicks What's This on the right-click menu, the application displays this type of content. This content does not support formatting and is not supported in all output formats. The values for this setting are defined as follows:



<b>Value</b>	<b>Description</b>
<b>None</b>	Does not include the content in What's This help content.
<b>Define</b>	Displays the content both in What's This help and in the topic where the content is defined, such as in a glossary or a terminology list.
<b>Define with no output</b>	Displays the content only as What's This help. The content is not included in the topic where the content is defined.

## Character Styles Options

ePublisher provides many options to allow you to customize your content transformation process and implement the online features you need. For example, the options define table of contents levels, popup windows, and related topics, as well as many other features and behaviors.

### Abbreviation

Specifies that ePublisher provides alternate or expansion text for all abbreviations with the selected character format in your online output. This option is available only for character styles.

You must define an Abbreviation character format and an AbbreviationTitle marker to specify alternate text for abbreviations. The character format with this option enabled identifies the abbreviation in the generated output. The AbbreviationTitle marker defines the alternate text for the abbreviation. ePublisher adds the abbreviation alternate text you specify to the title attribute of the `abbr` tag in the output. When you hover over the abbreviation in the output file, a popup window displays the alternate or expansion text for the abbreviation.

### Acronym

Specifies that ePublisher provides alternate or definition text for all acronyms with the selected character format in your online output. This option is available only for character styles.

You must define an Acronym character format and an Acronym marker to specify alternate text for acronyms. The character format with this option enabled identifies the acronym in the generated output. The Acronym marker defines the alternate text for the acronym. ePublisher adds the acronym alternate text you specify to the title attribute of the `acronym` tag in the output. When you hover over the acronym in the output file, a popup window displays the alternate or definition text for the acronym.

## Additional CSS classes

Specifies an additional class name to include in the output class name for the selected style. This option is helpful if you are using a custom `.css` file. ePublisher supports this option only in output formats that support cascading style sheets. This option is available all styles except marker styles.

This option lets you specify other classes to add to the class statement in the generated output for the selected style. For example, if you have a class in your `.css` file called `Red` that defines the text color as red, you can specify `Red` in the **Additional CSS classes** option for each style you want to have red text. ePublisher adds this class to the class name in the generated output, such as `classname="sourcestylename Red"`.

## Citation

Specifies that ePublisher recognizes and formats the text with this style as quotation content. A **citation** is a reference or footnote to a book, article, or other material that specifies the source from which a quotation was borrowed. The citation for a quote enables users to go to a Web site that contains more information about the quote.

You must define a Citation paragraph or character format and a Citation marker to specify a URL citation. The paragraph or character format with this option enabled identifies the quote in the generated output. The Citation marker defines the URL for the Web site with more information. When you hover over the quote in the output file, a popup window displays the URL. This option is available only for paragraph and character styles.

## Generate output

Specifies whether the selected style is included in the generated output. This option is available for all styles except marker styles. By default, the **Generate output** option is set to **Enabled** so that all content is included in the output. If you select **Disabled**, the content with the selected style does not appear in the generated output.

## Pass Through

Specifies whether ePublisher should process the content with the selected character style applied or insert the content directly into your output without

being transformed and coded for your output format. By default this option is disabled. If you enable this option, ePublisher does not process the content when generating output. For example, you can use this option to embed HTML code into your generated output.

## See Also

Defines See Also link functionality for the selected paragraph or character style. **See Also** links are associative links, also called Alinks, that identify other topics that may be of interest to the user of the current topic. These links use internal identifiers to define links and the browser builds the list dynamically based on the topics available when the user displays the links.

To create a See Also link as inline text without a button, create a unique character style and select the **See Also** option for that style.

To use a button to display See Also links, create a unique paragraph style, select the **See Also** option for that style, and type the See Also text on that paragraph. The paragraph style properties selected in Style Designer affect the text of the **See Also** button.

You must also define SeeAlsoLink and SeeAlsoKeyword markers. HTML Help also supports the SeeAlsoLinkDisplayType and SeeAlsoLinkWindowType markers. These markers allow you to change how the See Also links are displayed in HTML Help. See Also links are not supported by all output formats, such as Dynamic HTML.

## Variable

Specifies the name for a variable that ePublisher creates. ePublisher sets the value of the variable to the contents of the selected paragraph or character style. You can then use this variable in your page layout override `Page.asp` file.

For example, if you want to include a section name in a banner area at the top of each topic in that section, you can assign a variable for the paragraph style of the section name. Then, you can modify the override `Page.asp` file to include this variable value in the page layout where you want it. Each topic in a section uses the section name for that section, until the next section name defined by the paragraph style changes the value for the next section of topics. To insert a variable in the override `Page.asp` file, insert the following code in the file, where *variable* is the name of the variable you specify in this option:

```
<div wwpage:content="wwvars:variable">replaced variable value</div>
```

After processing the `Page.asp` file for a topic where the variable value is `User Information`, ePublisher replaces this code with the following code in the output file:

```
<div>User Information</div>
```

## Table Styles Options

ePublisher provides many options to allow you to customize your content transformation process and implement the online features you need. For example, the options define table of contents levels, popup windows, and related topics, as well as many other features and behaviors.

### Additional CSS classes

Specifies an additional class name to include in the output class name for the selected style. This option is helpful if you are using a custom `.css` file. ePublisher supports this option only in output formats that support cascading style sheets. This option is available all styles except marker styles.

This option lets you specify other classes to add to the class statement in the generated output for the selected style. For example, if you have a class in your `.css` file called `Red` that defines the text color as red, you can specify `Red` in the **Additional CSS classes** option for each style you want to have red text. ePublisher adds this class to the class name in the generated output, such as `classname="sourcestyle Red"`.

### Dropdown

Specifies whether to use the selected table style to define an expand/collapse section, also called a dropdown hotspot. Some output formats do not support expand/collapse sections.

You can configure the expand/collapse section to initially be hidden or shown. When the user clicks the dropdown hotspot, the content expands and collapses to allow the user to view more or less information. The expand/collapse content can include multiple paragraph and table styles. You can use the DropDownEnd marker to identify the end of the expand/collapse content. If you have a specific table style that always serves as the ending point for your expand/collapse content and is not included in the expand/collapse content, set this option for that table style to **Break**. By default, all table styles are included in the expand/collapse content. The values for this setting are defined as follows:

<b>Value</b>	<b>Description</b>
<b>Break</b>	Specifies the table style is never included in expand/collapse content. A table style with this option set to <b>Break</b> ends the expand/collapse content section.
<b>Continue</b>	Specifies the selected table style does not change the expand/collapse content output. If the selected paragraph style is within an expand/collapse content section, the <b>Continue</b> value directs ePublisher to include the content as part of the expand/collapse content. This value is the default for table styles.
<b>End</b>	Allows the content to be inside the dropdown selection but it will signify the end of the dropdown similar to the behavior of a marker.
<b>Start open</b>	Displays the content in the expand/collapse section when the page is initially displayed. The table with this option set provides the hotspot where the user can click to show or hide the expand/collapse content. This table style starts the expand/collapse section.
<b>Start closed</b>	Hides the content in the expand/collapse section when the page is initially displayed. The table with this option set provides the hotspot where the user can click to show or hide the expand/collapse content. This table style starts the expand/collapse section.

### **Emit table markup**

Specifies whether the selected style includes all generated markup in the output.

### **First Table Cell Width**

Specifies the width of the first cell of the table. Useful when working with tables where you only need to specify the width of the first column while allowing the other columns to flow naturally in the output. Typically this means that you would also disable the **Use document cell widths** option when enabling this option.

## Generate output

Specifies whether the selected style is included in the generated output. This option is available for all styles except marker styles. By default, the **Generate output** option is set to **Enabled** so that all content is included in the output. If you select **Disabled**, the content with the selected style does not appear in the generated output.

## Last Table Cell Width

Specifies the width of the last cell of the table. Useful when working with tables where you only need to specify the width of the last column while allowing the other columns to flow naturally in the output. Typically this means that you would also disable the **Use document cell widths** option when enabling this option.

## Page break priority

Specifies how to handle page breaks for the selected paragraph style. This option specifies whether to create a new page starting with the selected paragraph style and allows you to avoid new pages with a heading and no content if two headings occur in a row in your source documents. The default value is **None**, which does not create a new page.

This option works with the **Page break handling** format setting to give you complete control for how ePublisher handles page breaks. To access the **Page break handling** format setting, click **Target Settings** on the **Target** menu. With the default **Page break handling** format setting, to create a new page every time the selected paragraph style exists, select the highest priority level of **1**. To create a new page only if the preceding paragraph did not create a new page, select a lower priority level of **2** through **5**. For more information, see "Defining New Pages (Page Breaks)".

## Use % table cell widths

This option works similar to the Use document cell widths, except that it converts the specific dimensions into relative percentages thus overcoming the problem of changing the font size of the generated output.

## Use document cell widths

Specifies whether ePublisher sets all cell widths for tables of the selected table style to the same values as defined in the source document. If you set this option to **Disabled**, ePublisher sets table cell widths based on the size of the viewer window.

If the font size for your online table text differs from your source documents, enabling the **Use document cell widths** option may not give you the desired results. For example, if fonts for online tables are larger than in your source

documents, some of the table text may wrap in the cells in the generated output.

## Page Styles Options

ePublisher provides many options to allow you to customize your content transformation process and implement the online features you need. For example, the options define table of contents levels, popup windows, and related topics, as well as many other features and behaviors.

### Additional CSS classes

Specifies an additional class name to include in the output class name for the selected style. This option is helpful if you are using a custom `.css` file. ePublisher supports this option only in output formats that support cascading style sheets. This option is available all styles except marker styles.

This option lets you specify other classes to add to the class statement in the generated output for the selected style. For example, if you have a class in your `.css` file called `Red` that defines the text color as red, you can specify `Red` in the **Additional CSS classes** option for each style you want to have red text. ePublisher adds this class to the class name in the generated output, such as `classname="sourcestyle Red"`.

### Breadcrumbs shown at bottom of page

Specifies whether to include breadcrumbs at the bottom of each page for the selected page style. **Breadcrumbs** form a linked path to show users the hierarchical location of the current topic in your online content. This clickable path steps you through the topics at the higher levels in the online organization of topics. This option is not available in some output types.

You can include breadcrumbs at the top of the page, at the bottom of the page, or both. The breadcrumb trail at the top of the output page is enabled by default.

### Breadcrumbs shown at top of page

Specifies whether to include breadcrumbs at the top of each page for the selected page style. **Breadcrumbs** form a linked path to show users the hierarchical location of the current topic in your online content. This clickable path steps you through the topics at the higher levels in the online organization of topics. This option is not available in some output types.

You can include breadcrumbs at the top of the page, at the bottom of the page, or both. The breadcrumb trail at the top of the output page is enabled by default.

### Company info bottom alignment

Specifies the alignment of the company information displayed at the bottom of each page for the selected page style. For example, you can align the company information to the left or to the right. To display the company information at the bottom, set the **Company info displayed at bottom** option to **Enabled**. This option is not available in some output types.

### Company info displayed at bottom

Specifies whether to display the company information at the bottom of pages with the selected page style. If you set this option to **Enabled**, ePublisher displays the information specified in the company-related format settings at the bottom of these pages. This option is not available in some output types.

Use the **Company info bottom alignment** option to adjust how ePublisher displays the company information. ePublisher does not display the company information in the Preview pane.

If you add a logo image to your project, ePublisher displays the logo next to your company contact information on the top or the bottom of your output pages. To include a company logo, you must first store the image file in the **Files** folder of your project.

### Company info displayed at top

Specifies whether to display the company information at the top of pages with the selected page style. If you set this option to **Enabled**, ePublisher displays the information specified in the company-related format settings at the top of these pages. This option is not available in some output types.

Use the **Company info top alignment** option to adjust how ePublisher displays the company information. ePublisher does not display the company information in the Preview pane.

If you add a logo image to your project, ePublisher displays the logo next to your company contact information on the top or the bottom of your output pages. To include a company logo, you must first store the image file in the **Files** folder of your project.

### Company info top alignment

Specifies the alignment of the company information displayed at the top of each page for the selected page style. For example, you can align the company information to the left or to the right. To display the company information at the top, set the **Company info displayed at top** option to **Enabled**. This option is not available in some output types.

### Custom document css file



Specifies the name of the custom `.css` file to include in each generated output file. You can use a custom `.css` file to modify the appearance of your content instead of using the Style Designer, or to augment the Style Designer settings. When you specify the name of the custom `.css` file in this option, ePublisher inserts the link tag within all HTML pages and links to the `.css` file specified. This option is available only for page styles.

You must store the custom `.css` file in the `Files` folder of your project for it to be recognized by ePublisher. If you use a subfolder within the `Files` folder for your `.css` file, click the **Browse** button next to the **Custom document css file** option to navigate to the subfolder. ePublisher populates this option with the correct path to the custom `.css` file.

### Generate output

Specifies whether the selected style is included in the generated output. This option is available for all styles except marker styles. By default, the **Generate output** option is set to **Enabled** so that all content is included in the output. If you select **Disabled**, the content with the selected style does not appear in the generated output.

### Navigation links shown at bottom of page

Specifies whether to display navigation links, also known as browse buttons, at the bottom of each page for the selected page style. The **Navigation Alignment** property for the page controls whether the navigation links are aligned to the left or right. To set the **Navigation Alignment** property, on the **Properties** tab, click **Navigation**. This option is not available in some output types.

### Navigation links shown at top of page

Specifies whether to display navigation links, also known as browse buttons, at the top of each page for the selected page style. The **Navigation Alignment** property for the page controls whether the navigation links are aligned to the left or right. To set the **Navigation Alignment** property, on the **Properties** tab, click **Navigation**. This option is not available in some output types.

### Output file extension

Specifies the file extension to use for output files. If you select a page style, specify a value such as **.htm** or **.html** for your output files. If you select a graphic style, specify a value that corresponds to the graphic file format you selected for the **Format** option, such as **.jpg**, **.gif**, or **.png**. This option is available only for graphic and page styles.

## Page Styles Options (PDF - XSL-FO)

ePublisher provides many options to allow you to customize your content transformation process and implement the online features you need. The **Page Styles Options** can be used to customize your PDF-XSL-FO output. Selected styles will be applied to the text in that region.

### **Even page footer style**

Specifies a **Paragraph Style** to be applied to the footer text on even pages.

### **Even page footer text**

Specifies whether to display the first heading's text of each even page as the text in the footer.

### **Even page header style**

Specifies a **Paragraph Style** to be applied to the header text on even pages.

### **Even page header text**

Specifies whether to display the first heading's text of the page as the text on even header pages.

### **First Master Page**

Select whether to enable or disable the first master page. By default, the **First Master Page** option is set to **Disabled**.

### **First page footer style**

Specifies a **Paragraph Style** to be applied to the footer text of the first page.

### **First page footer text**

Specifies whether to display the first heading style of the page as the text in the first page footer.

### **First page header style**

Specifies a **Paragraph Style** to be applied to the header text of the first page.

### **First page header text**

Specifies whether to display the first heading style of the page as the text in the first page header.

### **Generate output**

Specifies whether the selected style is included in the generated output. This option is available for all styles except marker styles. By default, the **Generate output** option is set to **Enabled** so that all content is included in the output. If you select **Disabled**, the content with the selected style does not appear in the generated output.

### **Last Master Page**

Select whether to enable or disable the last master page. By default, the **Last Master Page** option is set to **Disabled**.

### **Last page footer style**

Specifies a **Paragraph Style** to be applied to the footer text of the last page.

### **Last page footer text**

Specifies whether to display the first heading style of the page as the text in the last page footer.

### **Last page header style**

Specifies a **Paragraph Style** to be applied to the header text of the last page.

### **Last page header text**

Specifies whether to display the first heading style of the page as the text in the last page header.

### **Odd page footer style**

Specifies a **Paragraph Style** to be applied to the footer text on odd pages.

### **Odd page footer text**

Specifies whether to display the first heading style of the page as the text in each odd page footer.

### **Odd page header style**

Specifies a **Paragraph Style** to be applied to the header text on odd pages.

### **Odd page header text**

Specifies whether to display the first heading style of the page as the text in each odd page header.

### **Output file extension**

Specifies the file extension to use for output files. If you select a page style, specify a value such as **.htm** or **.html** for your output files. If you select a graphic style, specify a value that corresponds to the graphic file format you selected for the **Format** option, such as **.jpg**, **.gif**, or **.png**. This option is available only for graphic and page styles.

## Graphic Styles Options

ePublisher provides many options to allow you to customize your content transformation process and implement the online features you need. For example, the options define the size and quality of the images in the generated output, as well as many other features and behaviors. For more information, see “Defining the Appearance of Images”

### Additional CSS classes

Specifies an additional class name to include in the output class name for the selected style. This option is helpful if you are using a custom `.css` file. ePublisher supports this option only in output formats that support cascading style sheets. This option is available all styles except marker styles.

This option lets you specify other classes to add to the class statement in the generated output for the selected style. For example, if you have a class in your `.css` file called `Red` that defines the text color as red, you can specify `Red` in the **Additional CSS classes** option for each style you want to have red text. ePublisher adds this class to the class name in the generated output, such as `classname="sourcestylename Red"`.

### Allow by-reference vector images (SVG)

Allows vector image `.svg` files to be imported by-reference in the source documents. ePublisher includes these images in the generated output files. This option is available only for graphic styles.

### By reference graphics

Allows images to be imported in source documents by reference. ePublisher preserves the original image file names that are imported by reference and does not assign file names to these images. This option is available only for graphic styles.

### By reference graphics use document dimensions

Resizes all images imported by reference in the source documents to use the size defined within the source documents. Enabling this option can reduce the clarity of images by causing ePublisher to rasterize the images to achieve the size defined in the source documents. ePublisher preserves the original image

file names that are imported by reference and does not assign file names to these images. This option is available only for graphic styles.

## Color bit depth

Specifies the number of bits of data carrying the color information in your images. The range of colors available through digital computer images is usually expressed in terms of **bit depth**. Common bit depth levels for images include 8-bit and 24-bit color. In general, more bits of data make more colors available. This option applies only to `.gif` and `.png` images. `.jpg` images are 32-bit images. This option is available only for graphic styles.

## Format

Specifies the output format for images with the selected graphic style. By default, ePublisher transfers all images in your source documents, regardless of their file format, to Web-ready `.jpg` images. You may want to use other image formats for online delivery. For example, `.gif` images can produce similar quality images as `.jpg`, but the file size is smaller. The `.gif` format can also support transparent colors. You can also create `.png` images, which combine some of the best qualities of both `.jpg` and `.gif` files.

If you use the same format and size as your source document images, you can avoid the rasterization process and improve the clarity of your images in your generated output. For more information, see “Defining the Appearance of Images”.

This option works with the **Output file extension** and **JPG Quality** options. If you select **JPG**, set **Output file extension** to `.jpg` and set **JPG Quality** to the appropriate value to transform your images with the quality level you need. This option is available only for graphic styles. For more information, see “Choosing an Image File Format and Quality Level”.

## Generate output

Specifies whether the selected style is included in the generated output. This option is available for all styles except marker styles. By default, the **Generate output** option is set to **Enabled** so that all content is included in the output. If you select **Disabled**, the content with the selected style does not appear in the generated output.

## Grayscale

Specifies whether to transform your original color images into grayscale versions in your generated content. If you set the **Grayscale** option to **Enabled**, ePublisher removes the color saturation of the original images when those images are transformed with your content. If the original source images are grayscale, this option has no effect. This option is available only for graphic styles.

## Interlaced

Specifies whether to transform images as interlaced images. **Interlaced images** initially load more quickly in browsers and become more clear as the browser displays additional detail. This option is available only for graphic styles.

## JPG Quality

Specifies the quality of the `.jpg` image files ePublisher creates from the images in your source documents with the selected graphic style. This value is a percentage of the original source document quality. For example, set this option to **100** to have ePublisher duplicate the quality of your original images. The default value is **75**.

Once you select **JPG** in the **Format** option for a graphic style, set the **JPG Quality** option. The quality level impacts both the visual aspects of your images and the size of the generated files. Higher-quality images require larger files, which require more time to download and display. This option does not affect `.gif` or `.png` images. This option is available only for graphic styles.

## Maximum image height (px)

Specifies the maximum height, in pixels, for images with the selected graphic style. During conversion, ePublisher resizes source document images taller than this value and maintains the original aspect ratio of each image. This option works with the **Render DPI**, **Scale %**, and **Maximum image width** options to make sure your images are correctly sized for online delivery. This option is available only for graphic styles.

ePublisher automatically converts images in your source document into Web-ready formats. However, the size of your print image may not be appropriate for online delivery. ePublisher provides several ways to modify the size of your images for online delivery without affecting the original images. While you can specify a fixed image width and height for all images using a specific graphic style, it is often most efficient to define the maximum height and width for an image.

## Maximum image width (px)

Specifies the maximum width, in pixels, for images with the selected graphic style. During conversion, ePublisher resizes source document images wider than this value and maintains the original aspect ratio of each image. This option works with the **Render DPI**, **Scale %**, and **Maximum image height** options to make sure your images are correctly sized for online delivery. This option is available only for graphic styles.

ePublisher automatically converts images in your source document into Web-ready formats. However, the size of your print image may not be appropriate for online delivery. ePublisher provides several ways to modify the size of your images for online delivery without affecting the original images. While you can specify a fixed image width and height for all images using a specific graphic style, it is often most efficient to define the maximum height and width for an image.

### Maximum thumbnail height (px)

Specifies the maximum height, in pixels, for thumbnails of images with the selected graphic style. A **thumbnail** is a small copy of an image displayed in place of the original image online to conserve space and download time. This option is available only for graphic styles.

When you enter a value in this option, ePublisher automatically creates thumbnails for any image that exceeds the maximum thumbnail height specified. When the user clicks the thumbnail in the generated output, the full-size image is displayed. Use the **Thumbnail page style** option to alter the appearance of the window in which ePublisher displays the full-size images. If you do not want thumbnail images used in your online output, leave this option value blank.

### Maximum thumbnail width (px)

Specifies the maximum width, in pixels, for thumbnails of images with the selected graphic style. A **thumbnail** is a small copy of an image displayed in place of the original image online to conserve space and download time. This option is available only for graphic styles.

When you enter a value in this option, ePublisher automatically creates thumbnails for any image that exceeds the maximum thumbnail width specified. When the user clicks the thumbnail in the generated output, the full-size image is displayed. Use the **Thumbnail page style** option to alter the appearance of the window in which ePublisher displays the full-size images. If you do not want thumbnail images used in your online output, leave this option value blank.

### Output file extension

Specifies the file extension to use for output files. If you select a page style, specify a value such as **.htm** or **.html** for your output files. If you select a graphic style, specify a value that corresponds to the graphic file format you selected for the **Format** option, such as **.jpg**, **.gif**, or **.png**. This option is available only for graphic and page styles.

### Render DPI

Specifies the dots per inch (DPI) to use for images with the selected graphic style. ePublisher defaults this value to **96**. If you specify a DPI that differs from the DPI of your source images, ePublisher rasterizes the image to create the new image for online delivery. If your source documents images have settings for a print-level resolution, such as 300 DPI or higher, you can reduce the image resolution for online delivery. Higher DPI images create larger files that require more time to download. In addition, most monitors display a resolution of 96 DPI, so higher resolutions do not increase the quality of the image displayed online. This option is available only for graphic styles.

Although transforming an image from 300 DPI to 96 DPI helps optimize your images for online delivery, it also affects the width and height of your images. Because a resolution of 300 DPI contains more dots per inch than a resolution of 96 DPI, the image ePublisher transforms will be roughly 68% smaller than the original image. For example, a 300 DPI image that is 100 x 100 pixels will be 32 x 32 pixels when transformed to a 96 DPI image.

To counter this effect, use the **Scale %** option in conjunction with the **Render DPI** option to control the size of your images. In the example of transforming a 300 DPI image to 96 DPI, set the **Scale %** option to **312**, which generates an image that has roughly the same dimensions as the original source image. You can also use the **Scale %** and **Render DPI** options together with the **Maximum image height** and **Maximum image width** options to make sure your images are correctly sized for online delivery.

## Responsive image sizing

When enabled, generated output will resize the images based on the available screen area of the device. The image will never be larger than its generated size, however, if the available screen size is less than the size of the image, the image will be scaled down in size so that it fits withing the viewable area.

## Scale %

Specifies the percentage ePublisher resizes all images with the selected graphic style. For example, if you specify **50**, ePublisher reduces each image with the selected graphic style to half its original size. When the actual width and height of an image is not important, you can specify a scaling percentage to apply to all images with the selected graphic style. This option allows you to modify the size of all images associated with the graphic style in relation to their original sizes. Modifying images in this way retains the original aspect ratio of each image.

This option works with the **Render DPI**, **Maximum image height**, and **Maximum image width** options to make sure your images are correctly sized for online delivery. This option is available only for graphic styles. For more information, see "Defining the Appearance of Images".



## Thumbnail page style

Specifies the name of the page style that ePublisher uses when displaying full-size images in a separate window after a user clicks a thumbnail image within the generated output. You can create a custom page style and then select the name of the custom page style to use for full-size image windows displayed from thumbnail images. If you leave the **Thumbnail page style** option blank, ePublisher uses the page style of the main content for the full-size image windows displayed from thumbnail images. This option is available only for graphic styles.

You must enter a value in the **Maximum thumbnail height** and **Maximum thumbnail width** options for ePublisher to recognize and use the **Thumbnail page style** option.

## Transparency

Specifies whether to transform all white regions in `.gif` or `.png` images with the selected graphic style to transparent areas. In some images, you may want to set a color to transparent. For example, if your source document has a white background, images with a white background display as though they do not have a background. However, if your online content has a different color background, the background of these images appear as white areas.

Select the graphic style and enable the **Transparency** option in ePublisher to transform the white background into a transparent one. Only `.gif` and `.png` images support transparent colors. This option is available only for graphic styles.

# Marker Styles Options

ePublisher provides many options to allow you to customize your content transformation process and implement the online features you need. For example, the options define table of contents levels, popup windows, and related topics, as well as many other features and behaviors.

## Marker type

Specifies the function associated with the selected marker style. ePublisher defines default names for the marker styles to support selected features. For example, if you have a marker style named Filename, ePublisher sets this option to allow that marker to define file names for output files. These default associations automate ePublisher configuration. You can specify your own marker styles and assign the function for each marker style. This option is available only for marker styles. For more information, see “Defining Marker Types”.

## Navigation title

Specifies whether the selected marker type supports the **Navigation title** function. If you enable this option for a marker type, you can use the marker to specify alternate text to display in the table of contents for a topic. For example, if you have a topic that does not have a heading, such as a topic that contains just a table or chart, you can add a marker with this option enabled to that topic and specify the text you want to be displayed in the table of contents for that topic.

## Search relevance weight

Reverb allows you to add a **Search relevance weight** to your **Marker Styles**, giving you the opportunity to control the order of the Search Results, assigning to every word with that marker style the specified weight and then saved in the index file. The default value is 1.

The predefined values for this option are defined as follows (but you can define any integer value): **Ignore**, **1**, **10**, **20**, **50** and **100**. If you choose to **Ignore** it (which is going to be 0), it means that the style is not going to be shown in your results.

## Variable

Specifies the name for a variable that ePublisher creates. ePublisher sets the value of the variable to the contents of the selected marker, paragraph or character style. You can then use this variable in your page layout override `Page.asp` file.

For example, if you want to include a section name in a banner area at the top of each topic in that section, you can assign a variable for the paragraph style of the section name. Then, you can modify the override `Page.asp` file to include this variable value in the page layout where you want it. Each topic in a section use the section name for that section, until the next section name defined by the paragraph style changes the value for the next section of topics. To insert a variable in the override `Page.asp` file, insert the following code in the file, where *variable* is the name of the variable you specify in this option:

```
<div wwpage:content="wwvars:variable">replaced variable value</div>
```

After processing the `Page.asp` file for a topic where the variable value is `User Information`, ePublisher replaces this code with the following code in the output file:

```
<div>User Information</div>
```

# Customizing WebWorks Reverb 2.0

[Changing the Appearance of WebWorks Reverb 2.0](#)  
[Target Settings for WebWorks Reverb 2.0](#)  
[Customizing the Top-Level Entry File](#)  
[Customizing TOC Menu Item Display](#)  
[Customizing the Splash Page in WebWorks Reverb 2.0](#)  
[Using Context-Sensitive Help in WebWorks Reverb 2.0](#)  
[Configuring Client-Side Search for Reverb 2.0](#)  
[Searching WebWorks Reverb 2.0 - URL Method](#)  
[Incorporating Google Analytics for Your Reverb 2.0 Files](#)  
[Configuring Commenting and End-User Feedback for Reverb 2.0](#)  
[Customizing Icons in Your Reverb 2.0 Output Using Font Awesome](#)  
[Incorporating Web Fonts in Your Reverb 2.0 Output](#)  
['Was This Helpful?' Buttons](#)  
[Dropdown Collapse/Expand All Toggle Button](#)  
[Document Last Modified Date/Publish Date](#)  
[Customizing Related Topics](#)

WebWorks Reverb 2.0 uses SASS technology to build a robust and responsive layout. Using ePublisher's implementation of this technology, the user can customize any aspect of the visual layout.

## Changing the Appearance of WebWorks Reverb 2.0

Most customizations to WebWorks Reverb 2.0 rely upon changing one of three file types:

- *Page Templates* - Found in the Pages folder and have the extension .asp. Users may create overrides for these files in their project to change overall layouts.
- *SASS Files* - Found in the Pages\sass folder. Users may change the color, icons, and other layout specific behaviors of the skin with these files. Changes to these files will not have an effect on the content pages if styles have been defined within the source documents or in the Style Designer.
- *Images* - Found in the Pages\images folder. These files are injected into the layout via Connect.asp and can be replaced with another image.

For easy reference, see the below table for a list of template files.

Page Template	Use
<code>Connect.asp</code>	Entry point to the Reverb 2.0 run-time. This file controls the placement of the toolbar buttons, menu (table of contents/index), page, and footer.
<code>Header.asp</code>	Defines the layout and appearance of the Header.
<code>Footer.asp</code>	Defines the layout and appearance of the Footer.
<code>Index.asp</code>	Data file for each Reverb parcel's Index information.
<code>Page.asp</code>	Defines the layout and appearance of generated pages.
<code>Parcel.asp</code>	Data file for each Reverb parcel (help set).
<code>Search.asp</code>	Search integration page.
<code>Splash.asp</code>	Defines the layout and appearance of the Reverb 2.0 splash page.
<code>Unsupported_Browser.asp</code>	Defines the layout and appearance of the message that displays when a user visits a Reverb 2.0 help set with an unsupported browser.

## Using SASS To Customize WebWorks Reverb 2.0

WebWorks Reverb 2.0 makes use of SASS to produce a responsive HTML5 layout. These files are found in the Pages/sass folder of the Format. SASS file names end with the .scss extension. There are many files in this folder and all of the files have a specific purpose.

Of these files, they fall into two categories:

- Template Files - These are identified by their filename. A Template File's name will always begin with the underscore character. These files contain variables

that hold values that can be changed by the user to affect areas of the layout without changing the implementation of the layout itself.

- Layout Files - These files have filenames that do not begin with the underscore character. These files contain the implementations of the variables defined in template files.

For reference, refer to the below table for a summary of what each SASS file pertains to.

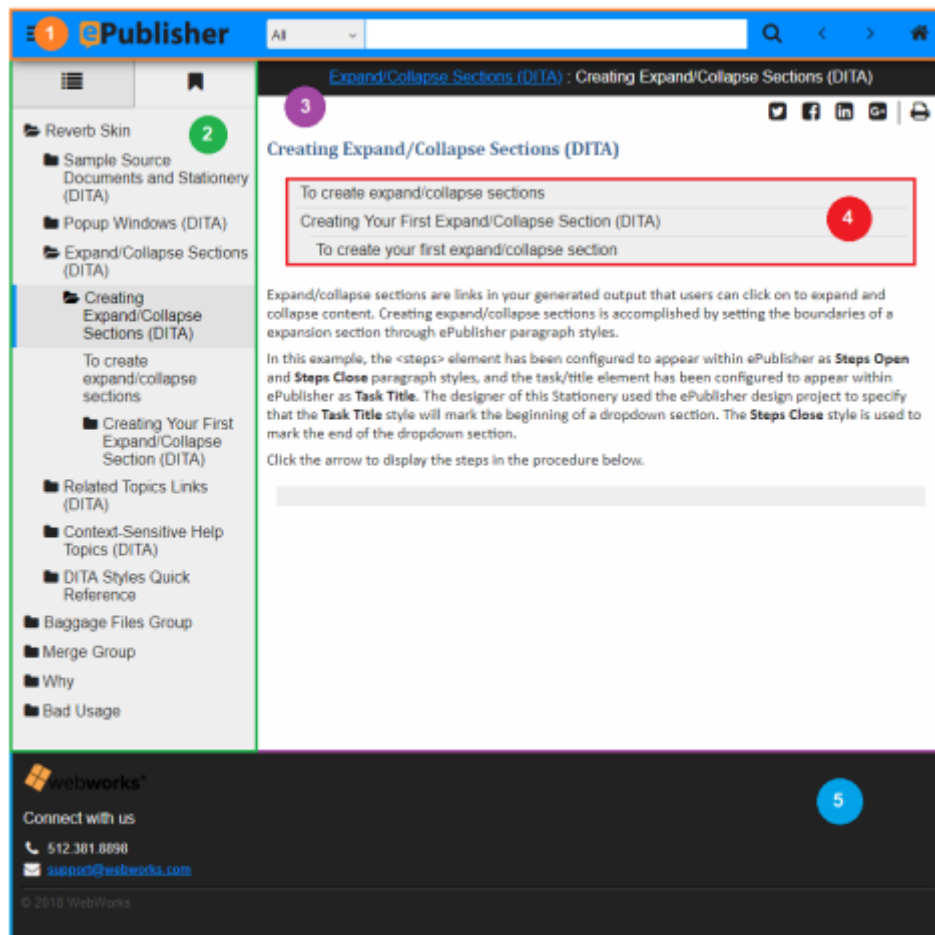
<b>SASS Filename</b>	<b>Pertains To</b>
<code>_borders.scss</code>	Contains variables for border values to be applied across the layout.
<code>_colors.scss</code>	Contains variables for color values to be applied across the layout.
<code>_fonts.scss</code>	Contains variables for font values to be applied across the layout.
<code>_functions.scss</code>	Contains definitions of custom SASS functions created by WebWorks for use in the layout.
<code>_icons.scss</code>	Contains variables for icon values to be applied across the layout.
<code>_sizes.scss</code>	Contains variables for size values to be applied across the layout.
<code>connect.scss</code>	Contains the implementation for top-level structure and core behavior for the layout.
<code>menu_initial_closed.scss</code>	Contains the menu configuration for when the menu is set to be closed upon first visit of the help set.
<code>menu_initial_open.scss</code>	Contains the menu configuration for when the menu is set to be open upon first visit of the help set.
<code>print.scss</code>	Contains style information for the presentation when a page is to be printed.
<code>search.scss</code>	Contains style information for the presentation of the search page.
<code>skin.scss</code>	Contains the aesthetic implementation for the layout. This file is where colors, fonts, borders, icons, and other visual style settings are implemented. This file

<b>SASS Filename</b>	<b>Pertains To</b>
	also contains the various implementations for individual skins.
<code>social.scss</code>	Contains style information for the social buttons in the page toolbar.
<code>webworks.scss</code>	Contains default style information for tables, the Mini-TOC, and other various small items in the layout. This file is implemented last and works well for custom implementations as well.

## SASS Variables In WebWorks Reverb 2.0

WebWorks Reverb 2.0 contains an implementation of SASS variables that has been constructed with efficiency and user-friendliness in mind. In each Template File, there are many variables that contain values that are implemented in the layout. Each variable has been given a name that describes the use of the value it holds. Variables are named in a procedural manner, each name starting with the section of the layout that it resides, and with every consecutive word defining more specifically what it pertains to.

As a reference, these graphics provide a visualization of the regions of the layout. The graphics can provide a starting point for finding the variable for the item that needs to be customized. The regions in the graphics have been color coded and numbered from 1 to 7. These numbers can be matched with a variable prefix that matches with a collection of variables that affect that region.







Number	Variable Prefix
1	\$toolbar_*
2	\$menu_*
3	\$page_*
4	\$mini_toc_*
5	\$footer_*
6	\$search_*
7	\$back_to_top_*

## Migrating SASS Overrides To Newer Format Versions

Upon installation of the latest version of ePublisher, it is recommended that a migration of overrides is performed on any project that contains overridden files that match to a prior format version. The migration procedure works for all file types in an ePublisher project, but is highlighted for SASS because of it's interconnected nature between the files. Sometimes, when a project is updated to the latest Format Version, users will get an error like this one when generating:

```
[Error] Traceback (most recent call last):
  File "C:\Program Files (x86)\WebWorks\ePublisher\2018.2\Helpers\python\WebWorks\compile_sass.py", line 25, in <module>
    compile_sass(input_file, output_file)
  File "C:\Program Files (x86)\WebWorks\ePublisher\2018.2\Helpers\python\WebWorks\compile_sass.py", line 13, in compile_sass
    _cssoutput = sass.compile(string=data)
  File "C:\Program Files (x86)\WebWorks\ePublisher\2018.2\Helpers\python\lib\site-packages\sass.py", line 640, in compile
    raise CompileError(v)
sass.CompileError: Error: Undefined variable: "$related-topics-title-background-color".
    on line 143 of stdin
>>> background: $related_topics_title_background_color;
```

-----^

This error happens most commonly when a newer version of one of the SASS files has a reference to a variable that should be inside an older, overridden SASS file. The solution for this is to migrate the overridden file to the latest version, comparing the overridden version to the latest version in a diff tool, and copying your local changes to the latest version of the same file.

Refer to this article for a comprehensive list of free and paid diff tools available for Windows:

<https://www.git-tower.com/blog/diff-tools-windows>

## Layout Colors In WebWorks Reverb 2.0 Skins

In the file `_colors.scss`, a convention has been created to allow for ease of use and user friendliness. The file makes use of Layout Colors that are defined at the top of the page. Every other color value in the layout inherits from these layout colors, so with this logic, you could change the color value of one layout color, and this value would be applied to many aspects of the layout. For the best results, the designer should take a top-down approach, first changing the values of colors at the top level, and then changing individual values for items that need minor adjustments.

For reference, we have created a table for each skin that details every variable that inherits from a certain layout color. These tables may seem like a lot, but just remember, the fact that one Layout Color affects so many values means that for the designer, the workload in the end is greatly reduced.

## Neo

Layout Color	Inherited By
\$_layout_color_1	\$back_to_top_background_color, \$footer_link_color, \$header_link_color, \$link_default_color, \$menu_nav_buttons_background_color_hover, \$menu_toc_item_current_highlight_color, \$mini_toc_entry_background_color_hover, \$page_breadcrumbs_link_color, \$page_breadcrumbs_link_color_hover, \$page_helpful_button_icon_color_selected, \$related_topics_entry_text_color, \$related_topics_title_background_color, \$toolbar_background_color, \$toolbar_button_background_color, \$toolbar_button_home_background_color, \$toolbar_button_menu_background_color, \$toolbar_button_next_background_color, \$toolbar_button_previous_background_color, \$toolbar_button_search_background_color, \$toolbar_button_translate_background_color, \$toolbar_logo_section_background_color, \$toolbar_search_section_background_color, \$toolbar_tab_background_color, \$toolbar_tabs_container_background_color, \$unsupported_browser_heading_text_color
\$_layout_color_2	\$back_to_top_caret_color, \$menu_index_link_color, \$menu_index_text_color, \$menu_index_title_color, \$menu_nav_buttons_current_icon_color, \$menu_nav_buttons_current_icon_color_click, \$menu_nav_buttons_current_icon_color_hover, \$menu_nav_buttons_icon_color, \$menu_nav_buttons_icon_color_hover, \$menu_text_color, \$menu_toc_item_icon_color, \$menu_toc_item_text_color, \$menu_toc_title_text_color, \$mini_toc_entry_text_color, \$mini_toc_entry_text_color_visited,

Layout Color	Inherited By
	\$page_dropdown_arrow_color, \$page_helpful_button_icon_color, \$page_helpful_message_text_color, \$page_toolbar_icon_divider_color, \$page_toolbar_social_icon_color, \$page_toolbar_social_icon_facebook_color, \$page_toolbar_social_icon_linkedin_color, \$page_toolbar_social_icon_twitter_color, \$page_toolbar_tool_icon_color, \$related_topics_dropdown_toggle_icon_color, \$related_topics_entry_icon_color, \$related_topics_title_text_color, \$search_filter_message_text_color, \$search_plain_text_color, \$search_result_count_message_text_color, \$search_result_icon_color, \$search_result_summary_highlight_text_color, \$search_result_summary_text_color, \$search_title_text_color, \$toolbar_button_home_icon_color, \$toolbar_button_icon_color, \$toolbar_button_menu_icon_color, \$toolbar_button_next_icon_color, \$toolbar_button_previous_icon_color, \$toolbar_button_search_icon_color, \$toolbar_button_translate_icon_color, \$toolbar_icon_color, \$toolbar_logo_link_text_color, \$toolbar_logo_text_color, \$toolbar_search_scope_option_text_color, \$toolbar_search_scope_options_text_color, \$toolbar_search_scope_selector_icon_color, \$toolbar_search_scope_selector_text_color, \$toolbar_tab_current_text_color, \$toolbar_tab_current_text_color_hover, \$toolbar_tab_text_color, \$toolbar_text_color, \$unsupported_browser_message_text_color
\$_layout_color_3	\$disqus_background_color, \$footer_text_color, \$header_text_color, \$menu_background_color, \$menu_index_background_color, \$menu_nav_buttons_current_background_color,

Layout Color	Inherited By
	\$menu_toc_background_color, \$menu_toc_item_background_color, \$mini_toc_background_color, \$toolbar_search_scope_option_background_color, \$toolbar_search_scope_options_background_color, \$toolbar_search_scope_selector_background_color
\$_layout_color_4	\$lightbox_button_close_icon_color, \$mini_toc_entry_text_color_hover, \$mini_toc_entry_text_color_visited_hover, \$no_javascript_text_color, \$page_breadcrumbs_text_color
\$_layout_color_5	\$footer_background_color, \$header_background_color, \$page_breadcrumbs_background_color
\$_layout_color_6	\$page_background_color, \$search_background_color, \$search_result_background_color

## Classic

Layout Color	Inherited By
<code>\$_layout_color_1</code>	<code>\$footer_link_color, \$header_link_color,</code> <code>\$link_default_color,</code> <code>\$mini_toc_entry_background_color_hover,</code> <code>\$page_helpful_button_icon_color_selected,</code> <code>\$related_topics_entry_text_color,</code> <code>\$toolbar_button_home_icon_color_click,</code> <code>\$toolbar_button_home_icon_color_hover,</code> <code>\$toolbar_button_menu_icon_color_click,</code> <code>\$toolbar_button_menu_icon_color_hover,</code> <code>\$toolbar_button_next_icon_color_click,</code> <code>\$toolbar_button_next_icon_color_hover,</code> <code>\$toolbar_button_previous_icon_color_click,</code> <code>\$toolbar_button_previous_icon_color_hover,</code> <code>\$toolbar_button_search_icon_color_click,</code> <code>\$toolbar_button_search_icon_color_hover,</code> <code>\$toolbar_button_translate_icon_color_click,</code> <code>\$toolbar_button_translate_icon_color_hover,</code> <code>\$toolbar_tab_current_background_color,</code> <code>\$unsupported_browser_heading_text_color</code>
<code>\$_layout_color_2</code>	<code>\$back_to_top_caret_color,</code> <code>\$menu_nav_buttons_current_icon_color,</code> <code>\$menu_nav_buttons_current_icon_color_click,</code> <code>\$menu_nav_buttons_current_icon_color_hover,</code> <code>\$menu_nav_buttons_icon_color,</code> <code>\$menu_text_color,</code> <code>\$menu_toc_item_icon_color,</code> <code>\$menu_toc_title_text_color,</code> <code>\$mini_toc_entry_text_color,</code> <code>\$mini_toc_entry_text_color_visited,</code> <code>\$page_dropdown_arrow_color,</code> <code>\$page_helpful_button_icon_color,</code> <code>\$page_toolbar_icon_divider_color,</code> <code>\$page_toolbar_social_icon_color,</code> <code>\$page_toolbar_social_icon_facebook_color,</code> <code>\$page_toolbar_social_icon_linkedin_color,</code> <code>\$page_toolbar_social_icon_twitter_color,</code> <code>\$page_toolbar_tool_icon_color,</code> <code>\$related_topics_dropdown_toggle_icon_color,</code> <code>\$related_topics_entry_icon_color,</code>

Layout Color	Inherited By
	\$related_topics_title_text_color, \$search_filter_message_text_color, \$search_plain_text_color, \$search_result_count_message_text_color, \$search_result_icon_color, \$search_result_summary_highlight_text_color, \$search_result_summary_text_color, \$search_title_text_color, \$toolbar_button_home_icon_color, \$toolbar_button_icon_color, \$toolbar_button_menu_icon_color, \$toolbar_button_next_icon_color, \$toolbar_button_previous_icon_color, \$toolbar_button_search_icon_color, \$toolbar_button_translate_icon_color, \$toolbar_icon_color, \$toolbar_logo_link_text_color, \$toolbar_logo_text_color, \$toolbar_search_scope_option_text_color, \$toolbar_search_scope_options_text_color, \$toolbar_search_scope_selector_icon_color, \$toolbar_search_scope_selector_text_color, \$toolbar_tab_text_color, \$toolbar_text_color, \$unsupported_browser_message_text_color
\$ _layout_color_3	\$disqus_background_color, \$footer_text_color, \$header_text_color, \$menu_background_color, \$menu_index_background_color, \$menu_nav_buttons_current_background_color, \$menu_toc_background_color, \$menu_toc_item_background_color, \$mini_toc_background_color, \$toolbar_search_scope_option_background_color, \$toolbar_search_scope_options_background_color, \$toolbar_search_scope_selector_background_color, \$toolbar_tab_background_color
\$ _layout_color_4	\$lightbox_button_close_icon_color, \$menu_nav_buttons_icon_color_click, \$menu_nav_buttons_icon_color_hover, \$menu_toc_item_current_icon_color, \$menu_toc_item_current_icon_color_click,



Layout Color	Inherited By
	\$menu_toc_item_current_icon_color_hover, \$menu_toc_item_current_text_color, \$menu_toc_item_current_text_color_click, \$menu_toc_item_current_text_color_hover, \$menu_toc_item_icon_color_click, \$menu_toc_item_icon_color_hover, \$menu_toc_item_text_color_click, \$menu_toc_item_text_color_hover, \$mini_toc_entry_text_color_hover, \$mini_toc_entry_text_color_visited_hover, \$no_javascript_text_color, \$page_breadcrumbs_link_color, \$page_breadcrumbs_link_color_hover, \$page_breadcrumbs_link_color_visited, \$page_breadcrumbs_link_color_visited_hover, \$page_breadcrumbs_text_color, \$toolbar_search_scope_option_text_color_hover, \$toolbar_search_scope_selector_icon_color_hover, \$toolbar_search_scope_selector_text_color_hover, \$toolbar_tab_current_text_color, \$toolbar_tab_current_text_color_hover, \$toolbar_tab_text_color_hover
\$_layout_color_5	\$footer_background_color, \$footer_copyright_message_text_color, \$footer_hr_color, \$footer_publish_date_text_color, \$header_background_color, \$page_breadcrumbs_background_color
\$_layout_color_6	\$page_background_color, \$search_background_color, \$search_result_background_color
\$_layout_color_7	\$back_to_top_background_color, \$back_to_top_background_color_hover, \$menu_toc_item_current_highlight_color, \$related_topics_title_background_color, \$toolbar_background_color, \$toolbar_button_background_color, \$toolbar_button_home_background_color, \$toolbar_button_home_background_color_click, \$toolbar_button_home_background_color_hover, \$toolbar_button_menu_background_color,

Layout Color	Inherited By
	\$toolbar_button_menu_background_color_click, \$toolbar_button_menu_background_color_hover, \$toolbar_button_next_background_color, \$toolbar_button_next_background_color_click, \$toolbar_button_next_background_color_hover, \$toolbar_button_previous_background_color, \$toolbar_button_previous_background_color_click, \$toolbar_button_previous_background_color_hover, \$toolbar_button_search_background_color, \$toolbar_button_search_background_color_click, \$toolbar_button_search_background_color_hover, \$toolbar_button_translate_background_color, \$toolbar_button_translate_background_color_click, \$toolbar_button_translate_background_color_hover, \$toolbar_logo_section_background_color, \$toolbar_search_section_background_color
\$_layout_color_8	\$menu_toc_item_current_background_color, \$menu_toc_item_current_background_color_click, \$menu_toc_item_current_background_color_hover
\$_layout_color_9	\$menu_nav_buttons_background_color_click, \$menu_nav_buttons_background_color_hover, \$menu_toc_item_background_color_click, \$menu_toc_item_background_color_hover, \$toolbar_search_scope_option_background_color_hover, \$toolbar_search_scope_selector_background_color_hover
\$_layout_color_10	\$menu_index_link_color, \$menu_index_text_color, \$menu_index_title_color, \$menu_toc_item_text_color, \$page_helpful_message_text_color

## Corporate

Layout Color	Inherited By
<code>\$_layout_color_1</code>	<code>\$mini_toc_entry_background_color_hover,</code> <code>\$page_breadcrumbs_link_color,</code> <code>\$page_breadcrumbs_link_color_hover,</code> <code>\$page_breadcrumbs_link_color_visited,</code> <code>\$page_breadcrumbs_link_color_visited_hover,</code> <code>\$related_topics_entry_text_color,</code> <code>\$unsupported_browser_heading_text_color</code>
<code>\$_layout_color_2</code>	<code>\$back_to_top_caret_color,</code> <code>\$menu_nav_buttons_icon_color_click,</code> <code>\$menu_nav_buttons_icon_color_hover,</code> <code>\$page_helpful_button_icon_color,</code> <code>\$page_toolbar_icon_divider_color,</code> <code>\$page_toolbar_social_icon_color,</code> <code>\$page_toolbar_social_icon_facebook_color,</code> <code>\$page_toolbar_social_icon_linkedin_color,</code> <code>\$page_toolbar_social_icon_twitter_color,</code> <code>\$page_toolbar_tool_icon_color,</code> <code>\$related_topics_dropdown_toggle_icon_color,</code> <code>\$related_topics_entry_icon_color,</code> <code>\$related_topics_title_text_color,</code> <code>\$toolbar_button_home_icon_color,</code> <code>\$toolbar_button_home_icon_color_click,</code> <code>\$toolbar_button_home_icon_color_hover,</code> <code>\$toolbar_button_icon_color,</code> <code>\$toolbar_button_menu_icon_color,</code> <code>\$toolbar_button_menu_icon_color_click,</code> <code>\$toolbar_button_menu_icon_color_hover,</code> <code>\$toolbar_button_next_icon_color,</code> <code>\$toolbar_button_next_icon_color_click,</code> <code>\$toolbar_button_next_icon_color_hover,</code> <code>\$toolbar_button_previous_icon_color,</code> <code>\$toolbar_button_previous_icon_color_click,</code> <code>\$toolbar_button_previous_icon_color_hover,</code> <code>\$toolbar_button_search_icon_color,</code> <code>\$toolbar_button_search_icon_color_click,</code> <code>\$toolbar_button_search_icon_color_hover,</code> <code>\$toolbar_button_translate_icon_color,</code> <code>\$toolbar_button_translate_icon_color_click,</code> <code>\$toolbar_button_translate_icon_color_hover,</code>

Layout Color	Inherited By
	\$toolbar_icon_color, \$toolbar_logo_link_text_color, \$toolbar_logo_text_color, \$toolbar_tab_text_color, \$toolbar_text_color
\$_layout_color_3	\$disqus_background_color, \$footer_text_color, \$header_text_color
\$_layout_color_4	\$menu_index_link_color, \$menu_index_text_color, \$menu_index_title_color, \$menu_toc_title_text_color, \$mini_toc_entry_text_color, \$mini_toc_entry_text_color_visited, \$no_javascript_text_color, \$page_breadcrumbs_text_color, \$page_dropdown_arrow_color, \$page_helpful_message_text_color, \$search_filter_message_text_color, \$search_plain_text_color, \$search_result_count_message_text_color, \$search_result_icon_color, \$search_result_summary_highlight_text_color, \$search_result_summary_text_color, \$search_title_text_color, \$toolbar_search_scope_option_text_color, \$toolbar_search_scope_option_text_color_hover, \$toolbar_search_scope_options_text_color, \$toolbar_search_scope_selector_icon_color, \$toolbar_search_scope_selector_icon_color_hover, \$toolbar_search_scope_selector_text_color, \$toolbar_search_scope_selector_text_color_hover, \$unsupported_browser_message_text_color
\$_layout_color_5	\$toolbar_button_home_background_color_hover, \$toolbar_button_menu_background_color_hover, \$toolbar_button_next_background_color_hover, \$toolbar_button_previous_background_color_hover, \$toolbar_button_translate_background_color_hover, \$toolbar_search_section_background_color_hover
\$_layout_color_6	\$menu_background_color, \$menu_index_background_color,

Layout Color	Inherited By
	\$menu_nav_buttons_current_background_color, \$menu_toc_background_color, \$menu_toc_item_background_color, \$mini_toc_background_color, \$page_background_color, \$page_breadcrumbs_background_color, \$search_background_color, \$search_result_background_color, \$toolbar_search_scope_option_background_color, \$toolbar_search_scope_options_background_color, \$toolbar_search_scope_selector_background_color
\$_layout_color_7	\$back_to_top_background_color, \$back_to_top_background_color_hover, \$menu_toc_item_current_highlight_color, \$related_topics_title_background_color, \$toolbar_background_color, \$toolbar_button_background_color, \$toolbar_button_home_background_color, \$toolbar_button_menu_background_color, \$toolbar_button_next_background_color, \$toolbar_button_previous_background_color, \$toolbar_button_search_background_color, \$toolbar_button_search_background_color_click, \$toolbar_button_search_background_color_hover, \$toolbar_button_translate_background_color, \$toolbar_logo_section_background_color, \$toolbar_search_section_background_color
\$_layout_color_8	\$toolbar_button_home_background_color_click, \$toolbar_button_menu_background_color_click, \$toolbar_button_next_background_color_click, \$toolbar_button_previous_background_color_click, \$toolbar_button_translate_background_color_click
\$_layout_color_9	\$menu_nav_buttons_background_color_click, \$menu_nav_buttons_background_color_hover
\$_layout_color_10	\$lightbox_button_close_icon_color, \$mini_toc_entry_text_color_hover, \$mini_toc_entry_text_color_visited_hover, \$toolbar_tab_current_text_color,

Layout Color	Inherited By
	\$toolbar_tab_current_text_color_hover, \$toolbar_tab_text_color_hover
\$_layout_color_11	\$menu_nav_buttons_current_icon_color, \$menu_nav_buttons_current_icon_color_click, \$menu_nav_buttons_current_icon_color_hover, \$menu_nav_buttons_icon_color, \$menu_text_color, \$menu_toc_item_current_icon_color, \$menu_toc_item_current_icon_color_click, \$menu_toc_item_current_icon_color_hover, \$menu_toc_item_current_text_color, \$menu_toc_item_current_text_color_click, \$menu_toc_item_current_text_color_hover, \$menu_toc_item_icon_color, \$menu_toc_item_icon_color_click, \$menu_toc_item_icon_color_hover, \$menu_toc_item_text_color, \$menu_toc_item_text_color_click, \$menu_toc_item_text_color_hover
\$_layout_color_12	\$footer_background_color, \$header_background_color, \$toolbar_tab_background_color, \$toolbar_tab_current_background_color, \$toolbar_tabs_container_background_color

## Metro

Layout Color	Inherited By
<code>\$_layout_color_1</code>	<code>\$back_to_top_background_color,</code> <code>\$footer_link_color, \$header_link_color,</code> <code>\$link_default_color,</code> <code>\$mini_toc_entry_background_color_hover,</code> <code>\$page_breadcrumbs_link_color,</code> <code>\$page_breadcrumbs_link_color_hover,</code> <code>\$page_helpful_button_icon_color_selected,</code> <code>\$related_topics_entry_text_color,</code> <code>\$related_topics_title_background_color,</code> <code>\$toolbar_tab_background_color,</code> <code>\$unsupported_browser_heading_text_color</code>
<code>\$_layout_color_2</code>	<code>\$back_to_top_caret_color,</code> <code>\$menu_toc_item_current_text_color,</code> <code>\$menu_toc_item_current_text_color_click,</code> <code>\$menu_toc_item_current_text_color_hover,</code> <code>\$menu_toc_item_text_color_click,</code> <code>\$menu_toc_item_text_color_hover,</code> <code>\$mini_toc_entry_text_color,</code> <code>\$mini_toc_entry_text_color_visited,</code> <code>\$page_dropdown_arrow_color,</code> <code>\$page_helpful_button_icon_color,</code> <code>\$page_helpful_message_text_color,</code> <code>\$page_toolbar_icon_divider_color,</code> <code>\$page_toolbar_social_icon_color,</code> <code>\$page_toolbar_social_icon_facebook_color,</code> <code>\$page_toolbar_social_icon_linkedin_color,</code> <code>\$page_toolbar_social_icon_twitter_color,</code> <code>\$page_toolbar_tool_icon_color,</code> <code>\$related_topics_dropdown_toggle_icon_color,</code> <code>\$related_topics_entry_icon_color,</code> <code>\$related_topics_title_text_color,</code> <code>\$search_filter_message_text_color,</code> <code>\$search_plain_text_color,</code> <code>\$search_result_count_message_text_color,</code> <code>\$search_result_icon_color,</code> <code>\$search_result_summary_highlight_text_color,</code> <code>\$search_result_summary_text_color,</code> <code>\$search_title_text_color,</code> <code>\$toolbar_logo_link_text_color,</code>

Layout Color	Inherited By
	\$toolbar_logo_text_color, \$toolbar_search_scope_option_text_color, \$toolbar_search_scope_options_text_color, \$toolbar_search_scope_selector_icon_color, \$toolbar_search_scope_selector_text_color, \$toolbar_tab_current_text_color, \$toolbar_tab_current_text_color_hover, \$toolbar_text_color, \$unsupported_browser_message_text_color
\$_layout_color_3	\$disqus_background_color, \$footer_text_color, \$header_text_color, \$menu_index_link_color, \$menu_index_text_color, \$menu_index_title_color, \$menu_nav_buttons_background_color_click, \$menu_nav_buttons_current_icon_color, \$menu_nav_buttons_current_icon_color_click, \$menu_nav_buttons_current_icon_color_hover, \$menu_nav_buttons_icon_color_hover, \$menu_text_color, \$menu_toc_item_current_icon_color, \$menu_toc_item_current_icon_color_click, \$menu_toc_item_current_icon_color_hover, \$menu_toc_item_icon_color_click, \$menu_toc_item_icon_color_hover, \$menu_toc_item_text_color, \$menu_toc_title_text_color, \$mini_toc_background_color, \$toolbar_button_home_background_color_click, \$toolbar_button_menu_background_color_click, \$toolbar_button_next_background_color_click, \$toolbar_button_previous_background_color_click, \$toolbar_button_translate_background_color_click, \$toolbar_search_scope_option_background_color, \$toolbar_search_scope_options_background_color, \$toolbar_search_scope_selector_background_color
\$_layout_color_4	\$lightbox_button_close_icon_color, \$mini_toc_entry_text_color_hover, \$mini_toc_entry_text_color_visited_hover, \$no_javascript_text_color, \$page_breadcrumbs_text_color, \$toolbar_button_home_icon_color,



Layout Color	Inherited By
	\$toolbar_button_home_icon_color_hover, \$toolbar_button_icon_color, \$toolbar_button_menu_icon_color, \$toolbar_button_menu_icon_color_hover, \$toolbar_button_next_icon_color, \$toolbar_button_next_icon_color_hover, \$toolbar_button_previous_icon_color, \$toolbar_button_previous_icon_color_hover, \$toolbar_button_search_icon_color, \$toolbar_button_search_icon_color_hover, \$toolbar_button_translate_icon_color, \$toolbar_button_translate_icon_color_hover, \$toolbar_icon_color, \$toolbar_tab_text_color, \$toolbar_tab_text_color_hover
\$_layout_color_5	\$footer_background_color, \$header_background_color, \$menu_toc_item_current_highlight_color, \$page_breadcrumbs_background_color, \$toolbar_background_color, \$toolbar_tabs_container_background_color
\$_layout_color_6	\$page_background_color, \$search_background_color, \$search_result_background_color
\$_layout_color_7	\$menu_background_color, \$menu_index_background_color, \$menu_nav_buttons_background_color, \$menu_nav_buttons_background_color_hover, \$menu_nav_buttons_current_background_color, \$menu_toc_background_color, \$menu_toc_item_background_color, \$toolbar_button_background_color, \$toolbar_button_home_background_color, \$toolbar_button_home_background_color_hover, \$toolbar_button_menu_background_color, \$toolbar_button_menu_background_color_hover, \$toolbar_button_next_background_color, \$toolbar_button_next_background_color_hover, \$toolbar_button_previous_background_color, \$toolbar_button_previous_background_color_hover, \$toolbar_button_translate_background_color,

Layout Color	Inherited By
	\$toolbar_button_translate_background_color_hover, \$toolbar_logo_section_background_color, \$toolbar_search_section_background_color
\$_layout_color_8	\$menu_toc_item_background_color_click, \$menu_toc_item_background_color_hover
\$_layout_color_9	\$menu_toc_item_current_background_color, \$menu_toc_item_current_background_color_click, \$menu_toc_item_current_background_color_hover, \$toolbar_tab_current_background_color

## ***Social***

<b>Layout Color</b>	<b>Inherited By</b>
\$_layout_color_1	\$back_to_top_background_color, \$footer_link_color, \$header_link_color, \$link_default_color, \$menu_nav_buttons_background_color_hover, \$menu_toc_item_current_highlight_color, \$mini_toc_entry_background_color_hover, \$page_breadcrumbs_link_color, \$page_breadcrumbs_link_color_hover, \$page_helpful_button_icon_color, \$page_helpful_button_icon_color_selected, \$related_topics_entry_text_color, \$related_topics_title_background_color, \$toolbar_background_color, \$toolbar_button_background_color, \$toolbar_button_home_background_color, \$toolbar_button_menu_background_color, \$toolbar_button_next_background_color, \$toolbar_button_previous_background_color, \$toolbar_button_search_background_color, \$toolbar_button_translate_background_color, \$toolbar_logo_section_background_color, \$toolbar_search_section_background_color, \$toolbar_tab_background_color, \$toolbar_tabs_container_background_color, \$unsupported_browser_heading_text_color
\$_layout_color_2	\$back_to_top_caret_color, \$menu_index_link_color, \$menu_index_text_color, \$menu_index_title_color, \$menu_nav_buttons_current_icon_color, \$menu_nav_buttons_current_icon_color_click, \$menu_nav_buttons_current_icon_color_hover, \$menu_nav_buttons_icon_color, \$menu_text_color, \$menu_toc_item_icon_color, \$menu_toc_item_text_color, \$menu_toc_title_text_color, \$mini_toc_entry_text_color, \$mini_toc_entry_text_color_visited,

Layout Color	Inherited By
	\$page_breadcrumbs_text_color, \$page_dropdown_arrow_color, \$page_helpful_message_text_color, \$page_toolbar_icon_divider_color, \$page_toolbar_social_icon_color, \$page_toolbar_social_icon_facebook_color, \$page_toolbar_social_icon_linkedin_color, \$page_toolbar_social_icon_twitter_color, \$page_toolbar_tool_icon_color, \$related_topics_entry_icon_color, \$search_filter_message_text_color, \$search_plain_text_color, \$search_result_count_message_text_color, \$search_result_icon_color, \$search_result_summary_highlight_text_color, \$search_result_summary_text_color, \$search_title_text_color, \$toolbar_search_scope_option_text_color, \$toolbar_search_scope_options_text_color, \$toolbar_search_scope_selector_icon_color, \$toolbar_search_scope_selector_text_color, \$unsupported_browser_message_text_color
\$_layout_color_3	\$disqus_background_color, \$footer_text_color, \$header_text_color, \$menu_background_color, \$menu_index_background_color, \$menu_nav_buttons_current_background_color, \$menu_toc_background_color, \$menu_toc_item_background_color, \$mini_toc_background_color, \$toolbar_search_scope_option_background_color, \$toolbar_search_scope_options_background_color, \$toolbar_search_scope_selector_background_color
\$_layout_color_4	\$lightbox_button_close_icon_color, \$menu_nav_buttons_icon_color_hover, \$mini_toc_entry_text_color_hover, \$mini_toc_entry_text_color_visited_hover, \$no_javascript_text_color, \$related_topics_dropdown_toggle_icon_color, \$related_topics_title_text_color, \$toolbar_button_home_icon_color, \$toolbar_button_home_icon_color_click,

Layout Color	Inherited By
	\$toolbar_button_home_icon_color_hover, \$toolbar_button_icon_color, \$toolbar_button_menu_icon_color, \$toolbar_button_menu_icon_color_click, \$toolbar_button_menu_icon_color_hover, \$toolbar_button_next_icon_color, \$toolbar_button_next_icon_color_click, \$toolbar_button_next_icon_color_hover, \$toolbar_button_previous_icon_color, \$toolbar_button_previous_icon_color_click, \$toolbar_button_previous_icon_color_hover, \$toolbar_button_search_icon_color, \$toolbar_button_search_icon_color_click, \$toolbar_button_search_icon_color_hover, \$toolbar_button_translate_icon_color, \$toolbar_button_translate_icon_color_click, \$toolbar_button_translate_icon_color_hover, \$toolbar_icon_color, \$toolbar_logo_link_text_color, \$toolbar_logo_text_color, \$toolbar_tab_current_text_color, \$toolbar_tab_current_text_color_hover, \$toolbar_tab_text_color, \$toolbar_tab_text_color_hover, \$toolbar_text_color
\$_layout_color_5	\$footer_background_color, \$header_background_color
\$_layout_color_6	\$page_background_color, \$page_breadcrumbs_background_color, \$search_background_color, \$search_result_background_color
\$_layout_color_7	\$toolbar_button_icon_color_disabled, \$toolbar_button_menu_icon_color_disabled

## Target Settings for WebWorks Reverb 2.0

WebWorks Reverb defines several format specific settings.

# **WebWorks Reverb 2.0 Target Settings**

The complete reference of WebWorks Reverb 2.0 Settings.

<b>Format Setting</b>	<b>Use</b>
<b>Browser Tab Icon (favicon)</b>	Specifies a .png .ico or .jpg file to use as the Browser Tab Icon (favicon) for the Reverb 2.0 help set.
<b>Display large images in lightbox</b>	When a thumbnail is used for an image, the full size version of the image can be viewed by clicking the image which is then displayed in a lightbox. When disabled the image displays in a separate file.
<b>Enable Print Icon</b>	Enables/disables the print icon in generated pages
<b>Entry Filename</b>	Specifies the name of the Reverb entry-point file name (default is "index.html")
<b>Feedback Email</b>	Defines the feedback email address for use in generated pages
<b>Feedback Email Message</b>	Defines the contents of the feedback email subject line and body section. Use \$Location; to insert the URL of the current page
<b>Skin</b>	Specifies an alternate plugin file to use for changing the look-and-feel of the Reverb skin.
<b>Use first document as splash page</b>	Determines initial page displayed
<b>WebWorks Help API Compatibility</b>	Enables/disables compatability with the WebWorks Help API allowing Reverb to replace existing implementations of WebWorks Help while using the same API.

## WebWorks Reverb 2.0 Toolbar Target Settings

The complete reference of WebWorks Reverb 2.0 Target Settings that affect the toolbar.





<b>Format Setting</b>	<b>Use</b>
<b>Google Translate Button</b>	Enables/disables Google Translate on content pages. The Reverb 2.0 output must be hosted on a server in order for the button to render in the toolbar.
<b>Home</b>	Enables/disables the home navigation button which navigates users to the first page in the browse sequence
<b>Linked Toolbar Logo</b>	Enables or Disables linking of the Toolbar Logo. If this setting is Enabled, the user must also set an address in the Toolbar Logo Link Address setting.
<b>Toolbar Logo</b>	Determines what Company Info item should be used, if any. Users can select None, Company name, or Company logo image.
<b>Toolbar Logo Link Address</b>	Determines an address the Linked Toolbar Logo directs to when a user clicks on the Toolbar Logo.
<b>Toolbar Logo Override</b>	Allows the user to set a custom image for the Toolbar Logo. As the name suggests, this setting will override any setting selected in the <b>Toolbar Logo setting</b> .

## WebWorks Reverb 2.0 Analytics Target Settings

The complete reference of WebWorks Reverb 2.0 Target Settings that pertain to analytics.

<b>Format Setting</b>	<b>Use</b>
<b>Google Tracking ID</b>	Sets the Tracking ID associated with a view on your Google Analytics account. Setting a value in this field enables analytics.
<b>Page “Was This Helpful?” Buttons</b>	Enables a set of buttons on content pages that provides the user with a pair of icon buttons that reports to analytics whether a page was helpful or not.
<b>Search “Was This Helpful?” Buttons</b>	Enables a set of buttons the search results page that provides the user with a pair of icon buttons that reports to analytics whether a set of search results was helpful or not.

## WebWorks Reverb 2.0 Menu Target Settings

The complete reference of WebWorks Reverb 2.0 Target Settings that affect the menu.

<b>Format Setting</b>	<b>Use</b>
<b>Generate Menu</b>	Determines whether the menu is generated or not.
<b>Menu Initial State</b>	Determines whether the menu is visible or hidden to users when they first visit the help set.
<b>Minimum Page Width for Docked Menu</b>	Sets the minimum width that the viewport must be to allow the menu to use docked behavior. If the viewport is less wide than this value, the user will be presented with the mobile view.

## WebWorks Reverb 2.0 Page Target Settings

The complete reference of WebWorks Reverb 2.0 Target Settings that affect the page.

<b>Format Setting</b>	<b>Use</b>
<b>Document Last Modified Date</b>	When enabled, this setting will display the date that the source document a content page is generated from was last modified. The date will be added to the content page in the output after source content.
<b>Dropdown Expand/Collapse Toggle Button</b>	This setting will render a button in the page toolbar that, when clicked, will simultaneously toggle all available dropdowns on a content page to an expanded or collapsed state. This button is only rendered on a content page if there are also dropdowns present on that page.
<b>Not Found Page</b>	When enabled, this setting will cause invalid links into the online help to be redirected to the <code>NotFound.asp</code> generated page.
<b>Not Found Page Style</b>	Use this setting to assign a specific Page style to control the layout and appearance of your <code>NotFound.asp</code> generated page, allowing this page to look and behave differently from other pages in the online help.
<b>Publish Date</b>	When enabled, this setting will display the date that the output was generated from ePublisher. The date will be added to the content page in the output after source content.
<b>Related Topic Dropdown Icon Position</b>	This setting determines whether the icon that handles the expand/collapse behavior for Related Topics is positioned to the Left or Right of the title of the Related Topics.
<b>Related Topic Dropdown Start Behavior</b>	This setting determines whether a set of Related Topics will be expanded or collapsed upon first visit of a content page.
<b>Reverb 2.0 Page Style</b>	Specifies the page style to use when processing the Reverb entry-point file that encapsulates the TOC, Index, Search, and content panels.

<b>Format Setting</b>	<b>Use</b>
<b>Splash Page Style</b>	Specifies the page style to use when processing the splash page (move)
<b>Use Dropdown for Related Topics</b>	When enabled, this setting applies dropdown behavior to Related Topics, allowing it to be expanded or collapsed with the click of a button.

## WebWorks Reverb 2.0 Footer Target Settings

The complete reference of WebWorks Reverb 2.0 Target Settings that affect the footer.

<b>Format Setting</b>	<b>Use</b>
<b>Footer Location</b>	Determines where the footer is generated. If End of Layout is selected, the footer will be generated underneath both the menu and the page content. If End of Page is selected, the footer will be generated next to the menu and underneath the page content.
<b>Footer Logo</b>	Determines what Company Info item should be used, if any. Users can select None, Company name, or Company logo image.
<b>Footer Logo Link Address</b>	Determines an address the Linked Footer Logo directs to when a user clicks on the Footer Logo.
<b>Footer Logo Override</b>	Allows the user to set a custom image for the Footer Logo. As the name suggests, this setting will override any setting selected in the <b>Footer Logo setting</b> .
<b>Generate Footer</b>	Determines whether the footer is generated or not.
<b>Linked Footer Logo</b>	Enables or Disables linking of the Footer Logo. If this setting is Enabled, the user must also set an address in the Footer Logo Link Address setting.
<b>Publish Date</b>	When enabled, this setting will display the date that the output was generated from ePublisher. The date will be added to the footer in the same area that the copyright information is displayed by default.

## Social Target Settings

The complete reference of Social Media Settings.

<b>Format Setting</b>	<b>Use</b>
<b>Disqus Identifier</b>	Enables support for user comments using the Disqus comment web service
<b>Disqus - Allow non-public networks</b>	Enable users to post Disqus comments behind a firewall or non-public website
<b>FaceBook Like</b>	Enables users to report "I like this!" in Facebook
<b>LinkedIn Share</b>	Enables users to share the current page to their LinkedIn community as well as increment the counter that shows how many times the link has been shared.
<b>Tweet This!</b>	Enable Twitter "Tweet This!" support

## Selecting an Alternate Skin for WebWorks Reverb 2.0

In WebWorks Reverb 2.0, you can select among several alternate skins to define the appearance of your help. When you specify an alternate skin, the underlying ASP and CSS files will be automatically updated to reflect the differences in that skin.

## To choose an alternate skin for WebWorks Reverb

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. In the **WebWorks Reverb 2.0** category, select the right column of the **Skin** entry to display the file picker button.
4. Click the file picker button to bring up an **Open** file dialog which will display a list of skin plugin files. Each skin plugin file is identifiable by a `.weplugin` extension.
5. Browse to the plugin file that you wish to use and double-click it to set the skin to that value.

Once you have set an alternate skin, you can later change it again using the same procedure.

**Warning:** If you have customized any files using the **Advanced** menu to create a Target override, then you will need to remove that customization and then re-implement it again after you change the skin setting. For more information on implementing and managing Target overrides see "Creating Target Overrides".

**Note:** If you are going to customize an alternative Reverb 2.0 skin, first set the skin type in the **Target Settings** then create a **Target Customization** override for the file(s) that you want to customize.

# Customizing the Top-Level Entry File

## Specifying the Entry Page Name

When you generate output for your project, a top-level entry file is created. This file defines the frameset for the help and gives the user a file to open that displays the complete help. By default, the top-level entry file is named `index.html`, but you can specify any name you need for the top-level entry file as long as you specify an `.html` or `.htm` extension.



### To rename the top-level file

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. In **Entry filename**, type the name you want to use for the top-level entry file.
4. Click **OK**.
5. Regenerate the project and review the results.

## Specifying the Entry Page Style

WebWorks Reverb 2.0 allows users to leverage ePublisher's Style Designer to set color and layout properties of the generated entry page. Users can specify a page style to use under the WebWorks Reverb 2.0 section of the Target Settings dialog.

## Customizing TOC Menu Item Display

### Specifying TOC Item CSS Class

WebWorks Reverb 2.0 allows users to be able to customize the style of each individual entry in the Table of Contents. Users can specify an entry style by simply editing the TOCEntryStyle marker to contain the name of the CSS class.

## Customizing the Splash Page in WebWorks Reverb 2.0

You can modify or remove the splash page that is displayed while your WebWorks Reverb 2.0 opens. You can customize the splash page in the Stationery, and then writers can override this customization in each project, as needed.

### Specifying the Splash Page Style

WebWorks Reverb 2.0 allows users to leverage ePublisher's Style Designer to set color and layout properties of the generated splash page. Users can specify a page style to use under the WebWorks Reverb 2.0 section of the Target Settings dialog.

## Replacing the Splash Image

The splash page is the first page that displays in the topic pane when the help set launches initially. By default, WebWorks Reverb 2.0 systems display the WebWorks Reverb 2.0 splash image. You can replace the default splash image with a custom image.

## To replace the splash page image

1. Identify the theme in use for your WebWorks Reverb 2.0 target that you want to modify.
2. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see “Stationery, Projects, and Overrides”.
3. **If you want to override the image for all WebWorks Reverb 2.0 targets**, create the **Formats** \WebWorks Reverb 2.0\Pages\images folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
4. **If you want to override the image for one WebWorks Reverb 2.0 target**, create the **Targets** \<Target Name>\Pages\images folder in your *projectname* folder, where *projectname* is the name of your ePublisher project and *Target Name* is the name of the specific target that will use this splash page image.
5. Copy the `splash.png` file from the following folder to the `images` override folder you created within your project folder:  
  
`Program Files\WebWorks\ePublisher Designer\Formats  
\WebWorks Reverb 2.0\ Pages\images`
6. Open the `splash.png` file you copied to your project override folder and modify it to be the splash page image you want.
7. Save and close the `splash.png` file.
8. Regenerate your project to review the changes.

## Modifying the Splash Page

Users may also create an override for the splash page template, `Splash.asp`. This allows users to change every aspect of a splash page’s appearance.

## Removing the Splash Page

When WebWorks Reverb 2.0 opens, it displays the splash page. However, instead of displaying the splash page, you can configure WebWorks Reverb 2.0 to display the first topic in the help.

## To remove the splash page

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. Set **Use first document as splash page** to **Enabled**.
4. Click **OK**.
5. Regenerate the project and review the results.

# Using Context-Sensitive Help in WebWorks Reverb 2.0

Context-sensitive help links provide content based on the context of what the user is doing. In many cases, this help content is based on the window that is open and active. For example, the **Help** button on a window in a software product can open a specific help topic that provides important information about the window and links to related topics.

WebWorks Reverb 2.0 allows you to use a TopicAlias marker to define an internal identifier for each topic. The benefit of using an internal identifier is that it allows file names to change without impacting the links from the product. The writer inserts this marker in a topic and specifies a unique value for that topic. ePublisher creates a mapping file to identify each topic associated with a unique value. Then, WebWorks Reverb 2.0 uses this internal identifier and the mapping file to display the correct topic. Before you can reference topics in WebWorks Reverb 2.0 using topic aliases, you must enable TopicAlias markers in your Stationery. For more information, see "Defining Filename Markers for Context-Sensitive Help Links".

## Mapping Files in WebWorks Reverb 2.0

WebWorks Reverb 2.0 does not generate a mapping file. However, you can see a list of defined contexts and topics in the parcel file. You can also use the Topics Report to verify that context-sensitive help topics have been created for each topic ID specified in your source document. The Topics Report lists the topic ID and the topic file created for each topic ID. Topic aliases report. For more information about the topics report, see "Topics Reports".

## Opening Context-Sensitive Help in WebWorks Reverb 2.0 using Standard URLs

You can open WebWorks Reverb 2.0 from the application using standard URLs.

To open a specific topic in WebWorks Reverb 2.0, use the following URL:

```
helplocation/index.html#context/topic_alias
```

or with the now optional `group_context` parameter:

```
helplocation/index.html#context/group_context/topic_alias
```

The variable parts of this URL are defined as follows:

### ***helplocation***

Specifies the location of the desired WebWorks Reverb 2.0 help set. If the help is on a Web server, specify the location using the `http` protocol and the Web site path to the root of the help, such as `http://www.webworks.com/help`.

### ***group\_context***

Now optional, but useful when you have the same topic alias value used in more than one group.

Specifies the group context value for the top-level group in which the topic resides in Document Manager. This group context is specified in merge settings for each top-level group.

### ***topic\_alias***

Specifies the value of the TopicAlias marker in the topic to open.

## **URL Commands Support by WebWorks Reverb 2.0**

WebWorks Reverb 2.0 supports additional commands in addition to context-sensitive help.

Command	Related Action
<code>index.html#context / group_name</code>	Display specified context-sensitive help topic
<code>index.html#page / child_page</code>	Displays and define child page
<code>index.html#search / search_words</code>	Initiates search for specific terms
<code>index.html#search / search_words #scope / group_name</code>	Initiates search for specific terms using only the specified groups
<code>index.html#toc /</code>	Display the table of contents panel
<code>index.html#index /</code>	Display the index panel
<code>index.html#parcels / group_name</code>	Loads the help set with only the specified groups

## Opening Context-Sensitive Help in WebWorks Reverb 2.0 using JavaScript

You can use JavaScript to open your context-sensitive help links when working with web applications or websites designed with HTML and JavaScript. For complete details of using WebWorks Reverb with web applications and websites, see <http://wiki.webworks.com/DevCenter/Projects/Reverb/For Web Applications>.

## Opening Context-Sensitive Help in WebWorks Reverb 2.0 using the WebWorks Help API

In addition to using simple URLs for opening context-sensitive help, you can alternatively use an API that is incorporated into your application. Using an API makes it easier to enable help buttons in your application so that the load the correct page in your Reverb 2.0 output.

WebWorks Reverb 2.0 uses the same API as WebWorks Help 4 and 5.

## Steps for Enabling the Context-Sensitive Help API for use with WebWorks Reverb 2.0

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. In the **WebWorks Reverb 2.0** category, set **WebWorksHelpAPICompatibility** to **Enabled**.
4. Click **OK** to save the target settings.
5. Have your application developers download the WebWorks Help SDK located at: <http://wiki.webworks.com/DevCenter/Projects/WebWorksHelp/WebWorksHelpSDK>. Depending on the technologies your application uses, your developers will be able to incorporate the API and then make context-sensitive API calls to your published Reverb 2.0 output.

## Configuring Client-Side Search for Reverb 2.0

This is the default search configuration for WebWorks Reverb 2.0 in the Target Settings. This search implementation requires no additional setup or configuration. The search implementation supports both wildcard (\*) and phrase searching.

## Configuring Synonyms

By defining a list of synonyms, you can increase the effectiveness of a search by grouping similar words together as a single result. You can define as synonyms words that are not in your source documents, but the main word does need to be in a source file. For example, if you define `video` and `avi` as synonyms of `movie` (main word), then when a user searches for `video`, a match will be scored showing the source files where the word itself is in, but also where the main word `movie` is as well. Please note that the inverse is not true, meaning that if a user searches instead for `movie`, files where the word `video` is will not be scored because of the synonym entry.

To add these definitions, you'll need to perform an override on a file called `locales.xml`, located by default in the ePublisher Designer installation directory here:

```
\Formats\WebWorks Reverb 2.0\Shared\common\locale
```

Once the file is copied to the correct location in your project directory, open it in any text- or XML-editing application (e.g., NotePad), and add entries to the `<Synonyms>` section. For example, your synonyms might look like this:

```
<Synonyms>

  <Word value="movie">

    <Synonym value="avi" />

    <Synonym value="video" />

  </Word>

  <Word value="designer">

    <Synonym value="pro" />

    <Synonym value="publisher" />

  </Word>

</Synonyms>
```

Please note that the Synonyms feature does not recognize multiple word values or synonyms. And, it ignores words (and synonyms) shorter in length than the specified value for `MinimumWordLength` defined in the same file. Synonyms are not case sensitive.

## Searching WebWorks Reverb 2.0 - URL Method

When viewing the URL for your help, you can use a search term after the URL, for example you can go to the link here: <https://www.webworks.com/Documentation/Reverb/index.html#search/ePublisher>

When clicked, the link above searches for the term "ePublisher" this can be changed to any search term, which will produce results accordingly.

## Incorporating Google Analytics for Your Reverb 2.0 Files

WebWorks Reverb 2.0 is not only a fully functional help system for your content, it's also a collection of distinct web page files that can be viewed individually on a website. In fact, you can very easily share URLs of individual page files from your Reverb 2.0 help system. These URLs will load just like a single web page within a browser. While this is very nice from an end-user perspective, because the load times are very minimal, it is even better from a publisher perspective. Simply put, you can now easily track the usage patterns and frequencies of all your Reverb



2.0 help pages, which is where the value of Google Analytics comes in; all you have to do is either determine the appropriate Google *Analytics Account ID* that your website is currently using (if Google Analytics has already been setup for your website), or create a Google Analytics Account which will then provide a unique identifier for your website or website path.

Once you have obtained the appropriate Google *Analytics Account ID*, then you simply set your ePublisher project's target setting located at: **Analytics > Google Tracking ID**.

## Configuring Commenting and End-User Feedback for Reverb 2.0

WebWorks Reverb 2.0 uses the *Disqus* commenting platform for enabling a very powerful and cost-effective commenting and discussion system. The integration with WebWorks Reverb 2.0 is not only transparent, it is optimized to ensure fast downloads. To enable this feature, all you need to do is set up a Disqus account and then create a *Disqus Site* for your Reverb 2.0 help volume(s). Once you create a *Disqus Site*, you will then use the site's *Disqus Site Shortname* as the setting value for your ePublisher project's target setting located at: **Social > Disqus Identifier**. The Disqus Site can then be used to track and manage all the threads used within your deployed Reverb 2.0 help volume(s). For the reports, you can use the Disqus engagement features: <https://disqus.com/>.

For more information regarding the API, refer to the documentation: <https://disqus.com/api>

## Customizing Icons in Your Reverb 2.0 Output Using Font Awesome

In **WebWorks Reverb 2.0**, all icons in the output can be configured to use another icon that is a part of Font Awesome's 5.15.4 package. In **Reverb 2.0**, all icons are injected to the output by using a unique character code defined by Font Awesome.

The icons can be found in the file `_icons.scss`.

Steps to customize a font awesome icon:

1. Create a format or target override for the file `_icons.scss`.
2. Open `_icons.scss` in a text editor.
3. Open the Font Awesome cheatsheet. You can use the cheatsheet found on the web.

Cheatsheet web location:

<https://fontawesome.com/v5/cheatsheet/free>

4. Find the 4-character code for the icon that is to be used.

**Note:** The web cheatsheet's character codes look like: `f2bb`. You only need the 4 characters starting at 'f'.

5. Find the variable for the icon that is to be changed.
6. Change the value of the variable to the new character code, wrapped in quotes with a leading backslash, ex: `\f2bb`.
7. Save the file and generate a new **Reverb 2.0** output.

## Incorporating Web Fonts in Your Reverb 2.0 Output

In **WebWorks Reverb 2.0**, you can select from many of the built-in fonts in the **Font Family Picker** window. However, it is possible to incorporate web fonts in to your **Reverb 2.0** output.

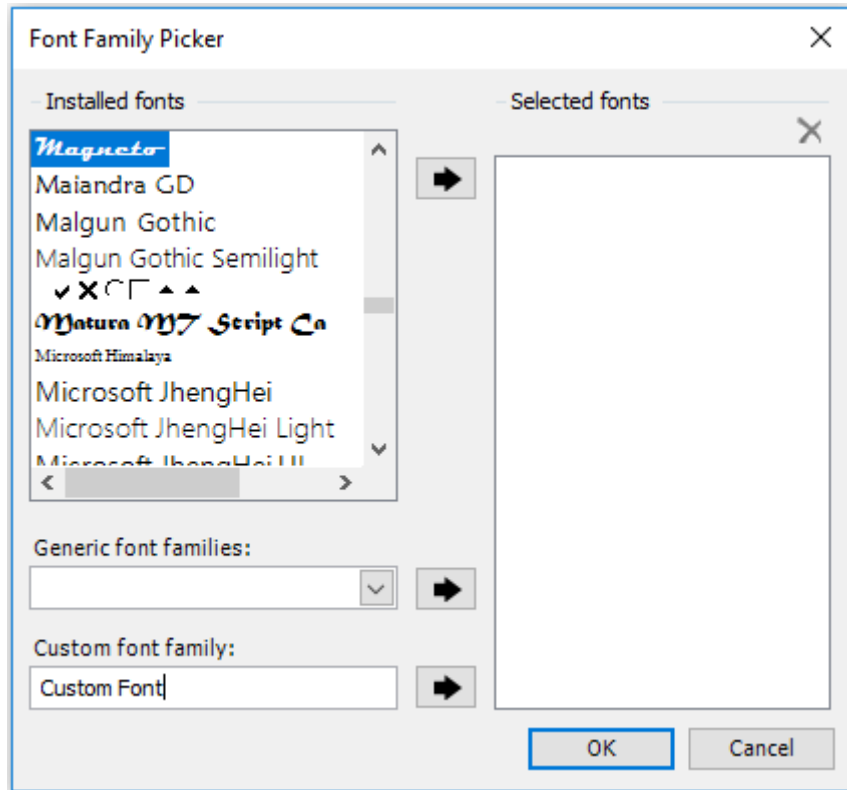
### Steps for integrating web fonts

1. Save the font file in the **Files** folder of your project. For the example below, `mycustomfont.ttf` would be placed into this folder.
2. Add the following to your `webworks.scss` file, just below the `@import` statements at the top of the file.

```
@font-face {  
  font-family: 'Custom Font';  
  src: url('../mycustomfont.ttf') format('truetype');  
}
```

**Note:** The URL contains the path to the location where the font is saved. The path must be entered correctly for the font to be located and used by the browser. By default, **ePublisher** copies any file inside the files folder of your project to the appropriate location of your output. The URL in the example above is a relative path from the output CSS file to the font file.

3. In the **WebWorks Reverb 2.0 Font Family Picker** window, enter the name of your custom font in the **Custom font family** input. Press the right arrow button to add the custom font to your **selected fonts**.



4. Click **OK** to save the target settings.

### Steps for integrating Google Fonts

1. Copy the `@import` statement for the selected Google font (Do not copy the style tags).

Example:

```
@import url('https://fonts.googleapis.com/css?family=Rubik');
```

2. Paste the `@import` statement in to your `webworks.scss` file, just below the other `@import` statements at the top of the file.
3. Add the following code just below the `@import` statement, replacing the font-family in the example with the the font-family of the selected font.

```
@font-face {
  font-family: 'Rubik', sans-serif;
}
```

4. In the **WebWorks Reverb 2.0 Font Family Picker** window, enter the name of your selected Google font in the **Custom font family** input. Press the right arrow button to add the custom font to your selected fonts
5. Click **OK** to save the target settings.

# Steps to Create Your First Disqus Site

1. Visit the URL: <https://disqus.com/>. Then select the **Sign Up** button. You will then be taken through a few simple forms which will create your *Disqus Login* as well as your initial *Disqus Site*.
2. In the field `Site URL`, specify the URL that best describes where your Reverb volume(s) will exist on your website.

**Note:** You can use a single Disqus site for all your Reverb deployments that exist on the same website or you can use separate Disqus sites for each separately deployed Reverb volume. Disqus provides a wide range of solutions for managing and moderating threads so choosing either strategy will be effective and depends more on how your organization's process will be handled.

3. In the field `Site Name`, specify a unique, human readable name that best describes the site. Then check the `Site Shortname` text box field and correct it to suit your preference. The *Site Shortname* will not only be used by ePublisher, but will also become the sub-domain used on Disqus. So if your *Site Shortname* is `epub`, then the URL used for this site's threads will be `http://epub.disqus.com`.
4. On the same page, specify the information for the *Primary Moderator*, other moderators can be added later. Then select the **Continue** button.
5. On the Settings page, specify your language and any optional features that you wish to also use. Then select the **Continue** button. You now have a *Disqus Login* as well as your first *Disqus Site*. No additional steps are required to begin using your new site

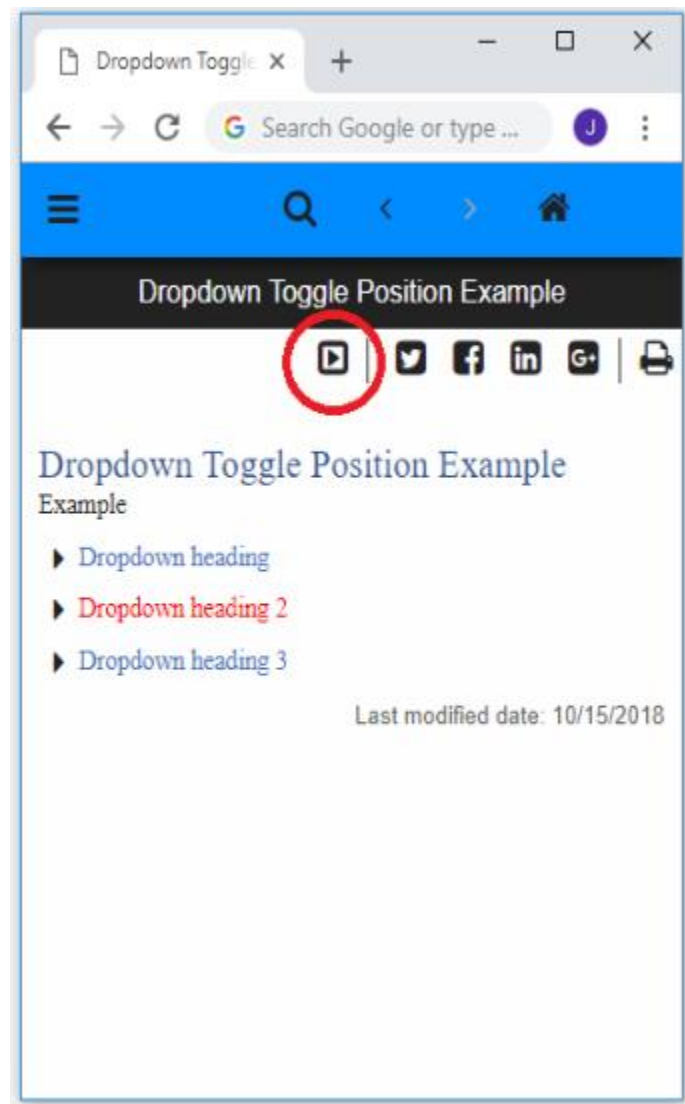
## 'Was This Helpful?' Buttons

The **Was This Helpful?** buttons feature allows you to provide your end-users with the ability to give anonymous feedback about their experience. The **Was This Helpful?** buttons feature can be used to record feedback on the current page or search results being viewed. A Google Tracking ID is required for these buttons to record analytic events.

## Dropdown Collapse/Expand All Toggle Button

Enables/disables a button that toggles all paragraph dropdown on the currently displayed page. This applies to all paragraphs with dropdown behavior, as well as

Related Topics with dropdown behavior enabled. This feature is made available by enabling the Dropdown Expand/Collapse Toggle Button Target Setting.



## Document Last Modified Date/Publish Date

ePublisher can add a time stamp to Reverb 2.0 output that displays the date of the most recent generation, as well as the last time a source document was modified. The last modified date can be displayed in the page content area just after the content by default, and the publish date is available in the same area as the last modified date, as well as near the copyright information in the footer. These features are made available by enabling the Document Last Modified Date and Publish Date Target Settings.

If you would like to change the date format, it can be done with an override to the locales.xml file.

Copy locales.xml from:

```
C:\Program Files\WebWorks\ePublisher\2024.1\Formats\Shared\common\locale\locales.xml
```

To:

```
<Project Folder>\Formats\Shared\common\locale\locales.xml
```

In the locales.xml file, locate the element with the name PublishDateFormat. The value inside this element can be changed to match the desired date format. For more information on how to write proper format values in this element, refer to Microsoft's documentation on Custom date and time format strings at: <https://docs.microsoft.com/en-us/dotnet/standard/base-types/custom-date-and-time-format-strings>.

## Customizing Related Topics

In Reverb 2.0, Related Topics have been given a generous set of customization options related to sizing, color, and an assignable icon for each topic entry.



## Customizing Related Topic Styling/Appearance

The appearance of Related Topics can be customized by making SASS changes to the file `webworks.scss`. In this file, all CSS declarations for items with a class name that contains `.Related_Topics` can be modified to change the appearance of Related Topics.

## Enabling/Disabling Related Topic Dropdown Behavior

Related Topics can be given dropdown behavior by enabling the Use Dropdown for Related Topics Target Setting. This will add an icon to the title bar, and enable collapsing and expanding of the Related Topic list when the title bar is clicked. This feature also works with the Dropdown Toggle Button feature.

# Customizing PDF - XSL-FO

Custom Header and Footers

Document Last Modified Date/Publish Date

Page Template Customizations

The *PDF - XSL-FO* output format is used by ePublisher to generate PDF files from the documents that ePublisher is generating. All of the styling and layout is controlled using the user interface of ePublisher, however some customizations require specific knowledge. This section covers common customizations.

## Custom Header and Footers

One of the advantages of selecting this format over the traditional PDF format is the ability to customize the footer and header depending on the Page Styles in the Style Designer. Please keep in mind that this will only be available per ePublisher Designer as the Style Designer is not available in ePublisher Express.

## **To customize the Header or Footer**

- 1.** In ePubublisher Designer, go to the **Style Designer** -> **Page Styles**
- 2.** Select the Page Style that you have defined and click the **Options** tab
- 3.** Depending on where you would like to have your information you can enter in information for Even or Odd pages
- 4.** Chose to Enable or Disable Header and Footer text under the Generate Output option
- 5.** Chose from the following Variables to customize the content



\$Title;	This variable expands to the title of the PDF. For the document-level PDF, this value is the title of the document or the value of the first paragraph. For the group-level PDF, this value is the name of the Group. For the project-level PDF, this value is the name of the Target
\$PageNumber;	This variable expands to the current page number
\$RunningTitle;	This variable expands to the text of the last paragraph that has a "Table of contents level" option value in the page sequence which generated a TOC entry
\$ChapterTitle;	This variable expands to the text of the top-most TOC entry of the document.
\$PublishDate;	This variable expands to the date of output generation using the same format settings configured in the <code>locales.xml</code> file.

**Note:** **Project Variables** and **Style Variables** can also be used in this option.

## Document Last Modified Date/Publish Date

ePublisher can add a time stamp to PDF XSL-FO Title Page that displays the date of the most recent generation. This feature is made available by enabling the Generate Publish Date on Title Page Target Setting.

If you would like to change the date format, it can be done with an override to the `locales.xml` file.

Copy `locales.xml` from:

```
C:\Program Files\WebWorks\ePublisher\2024.1\Formats\Shared\common\locale\locales.xml
```

To:

```
<Project Folder>\Formats\Shared\common\locale\locales.xml
```

In the `locales.xml` file, locate the element with the name `PublishDateFormat`. The value inside this element can be changed to match the desired date format. For

more information on how to write proper format values in this element, refer to Microsoft's documentation on Custom date and time format strings.

## Page Template Customizations

To customize your **PDF-XSL-FO** output, you can edit the following files:

**Note:** `Page.asp` in **PDF-XSL-FO** output is not the same as the `Page.asp` file in **Reverb** and **Reverb 2.0** outputs. It is used to layout the other `.asp` files.

### Page.asp

It is recommended that you do not make edits to this file.

### Body.asp

Edit this file if you would like to make customizations to the body pages of your output.

### Index.asp

Edit this file if you would like to make customizations to the index pages of your output.

### Title.asp

Edit this file if you would like to make customizations to the title page of your output.

### TOC.asp

Edit this file if you would like to make customizations to the table of contents of your output.

# Customizing Dynamic HTML

Using SASS to change the Appearance of Dynamic HTML

Modifying the Appearance of the Table of Contents in Dynamic HTML

Modifying the Appearance of the Index in Dynamic HTML

Other Changes to Text in the TOC and Index in Dynamic HTML

Document Last Modified Date/Publish Date

You can customize the appearance of Dynamic HTML in several ways. For example, you can customize the appearance of the table of contents and the index entries. The following sections describe these Dynamic HTML-specific customizations.

## Using SASS to change the Appearance of Dynamic HTML

To facilitate more advanced CSS handling you can enable a Target Setting in the group **Files** called **Use SASS to compile CSS**. If this setting is **Enabled** `webworks.css` will be generated from the SASS file called `webworks.scss`. This file is located in the `css` folder next to the `webworks.css` file.

## Modifying the Appearance of the Table of Contents in Dynamic HTML

ePublisher stores CSS settings that control the appearance of table of contents entries in the `webworks.css` file. You can create an override file to modify these settings for specific levels of the table of contents. For example, you can define a different font size and margin for each level in the table of contents.

## To modify the appearance of the table of contents

1. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see “Stationery, Projects, and Overrides”.
2. **If you want to override the CSS settings for all Dynamic HTML targets in the project**, create the `Formats\Dynamic HTML\Pages\css` folder in your project folder.
3. **If you want to override the CSS settings for a specific target**, create the `Targets\targetname\Pages\css` folder in your project folder, where *targetname* is the name of the target you want to override.
4. Copy the `webworks.css` file from the following folder to the override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats\Dynamic HTML\Pages\css
```

5. Open the `webworks.css` file you copied to your project override folder.
6. Find the code for `div.WebWorks_TOC_Levelx`, where *x* is the level number you want to modify. Then, specify the values within the braces to modify the font or margin:
  - To modify the font of all table of contents entries for the specified level, specify the name of the font you want, such as `font-family: Arial;`.
  - To modify the font size of all table of contents entries for the specified level, specify the size of the font you want, such as `font-size: 14pt;`.
  - To modify the left margin indent of all table of contents entries for the specified level, specify the indent you want, such as `margin-left: 10px;`.
7. Save the `webworks.css` file.
8. Regenerate your project to review the changes.

For example, the following figure illustrates how you could customize your table of contents entries.

```
div.WebWorks_TOC_Level1
{ font-size: 14pt;
  font-family: Arial;
  margin-left: 12px;
}
div.WebWorks_TOC_Level2
```

```
{ font-size: 12pt;  
  font-family: Arial;  
  margin-left: 24px;  
}
```

## Modifying the Appearance of the Index in Dynamic HTML

ePublisher stores CSS settings that control the appearance of index entries in the `webworks.css` file. You can create an override file to modify these settings for specific index entry levels. For example, you can define a different font size and margin for each level in the index.

## To modify the appearance of the index

1. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see “Stationery, Projects, and Overrides”.
2. **If you want to override the CSS settings for all Dynamic HTML targets in the project**, create the `Formats\Dynamic HTML\Pages\css` folder in your project folder.
3. **If you want to override the CSS settings for a specific target**, create the `Targets\targetname\Pages\css` folder in your project folder, where *targetname* is the name of the target you want to override.
4. Copy the `webworks.css` file from the following folder to the override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats\Dynamic HTML\Pages\css
```

5. Open the `webworks.css` file you copied to your project override folder.
6. Find the code for `div.WebWorks_Index_Levelx`, where *x* is the level number you want to modify. Then, specify the values within the braces to modify the font or margin:
  - To modify the font of all index entries for the specified level, specify the name of the font you want, such as `font-family: Arial;`.
  - To modify the font size of all index entries for the specified level, specify the size of the font you want, such as `font-size: 14pt;`.
  - To modify the left margin indent of all index entries for the specified level, specify the indent you want, such as `margin-left: 10px;`.
7. Save the `webworks.css` file.
8. Regenerate your project to review the changes.

For example, the following figure illustrates how you could customize your index entries.

```
div.WebWorks_Index_Level1
{ font-size: 14pt;
  font-family: Arial;
  margin-left: 12px;
}
div.WebWorks_Index_Level2
{ font-size: 12pt;
```

```
font-family: Arial;  
margin-left: 24px;  
}
```

## Other Changes to Text in the TOC and Index in Dynamic HTML

You can change the text in the table of contents and the index by adding the proper CSS coding between the braces for the appropriate style and class. Put the value after a colon, and put a semicolon at the end of the added coding. The following table summarizes the CSS coding for some common modifications.

Markup	Possible Values	Explanation
<code>font-style:</code>	<code>normal</code>   <code>italic</code>   <code>oblique</code>	Specifies whether the font should use the normal, also known as upright or roman, italic, or oblique faces within a font family.
<code>font-weight:</code>	<code>normal</code>   <code>bold</code>   <code>bolder</code>   <code>lighter</code>   <code>100</code>   <code>200</code>   <code>300</code>   <code>400</code>   <code>500</code>   <code>600</code>   <code>700</code>   <code>800</code>   <code>900</code>	Specifies the thickness of the font. The value <code>normal</code> is equal to <code>400</code> , and the value <code>bold</code> is equal to <code>700</code> .
<code>text-transform:</code>	<code>capitalize</code>   <code>uppercase</code>   <code>lowercase</code>   <code>none</code>	Specifies whether to transform the case of the text.
<code>text-align:</code>	<code>left</code>   <code>right</code>   <code>center</code>   <code>justify</code>	Specifies how to align the text on the page.

## Document Last Modified Date/Publish Date

ePublisher can add a time stamp to Dynamic HTML output that displays the date of the most recent generation. The publish date can be displayed in the page content area just after the content by default. This feature is made available by enabling the Page Style Option Publish Date shown at bottom of page.

If you would like to change the date format, it can be done with an override to the locales.xml file.

Copy locales.xml from:

```
C:\Program Files\WebWorks\ePublisher\2024.1\Formats\Shared\common\locale\locales.xml
```

To:

```
<Project Folder>\Formats\Shared\common\locale\locales.xml
```

In the locales.xml file, locate the element with the name PublishDateFormat. The value inside this element can be changed to match the desired date format. For



more information on how to write proper format values in this element, refer to Microsoft's documentation on Custom date and time format strings.

# Customizing Eclipse Help

[Using Markers to Specify Context Plug-ins in Eclipse Help](#)

[Using Markers to Specify Topic Descriptions for Context-Sensitive Help Topics in Eclipse Help](#)

You can customize the appearance and behavior of Eclipse Help in several ways. For example, you can specify context plug-in IDs for your Eclipse Help system. You can also specify topic descriptions for context-sensitive help topics.

## Using Markers to Specify Context Plug-ins in Eclipse Help

You can specify Eclipse Help context plug-ins by using Context Plugin markers in your source documents. Obtain the context plug-in IDs you need to specify for your source document groups from your development team. ePublisher places the context plug-ins you specify in your source documents in the `plugin.xml` file generated for each source document group you have in your project. The Eclipse developers use the context plug-ins defined in `plugin.xml` files to call your Eclipse Help system as appropriate from Eclipse plug-ins.

To enable specifying context plug-ins for Eclipse Help systems, you need to enable the Context Plugin marker. By default, ePublisher sets the **Marker type** option for a marker named Context Plugin to **Context Plugin**. You can create a marker with a different name and set the **Marker type** option for that marker to **Context Plugin**.

Then, writers can use this marker in the source documents to define context plug-in IDs for each of the source document groups in their project. Context plug-in IDs must follow these guidelines:

- Must be unique
- May specify only one context plug-in ID in each Context Plugin marker
- May contain alphanumeric characters
- Should not contain special characters or spaces, with the exception of underscore (`_`) characters

## To assign context plug-in behavior to context plug-in markers

1. Open your Stationery design project.
2. On the **View** menu, click **Style Designer**.
3. In the **Marker Styles**, select the marker style you want to modify.
4. On the **Options** tab, set **Marker type** to **Context Plugins**.

# Using Markers to Specify Topic Descriptions for Context-Sensitive Help Topics in Eclipse Help

In Eclipse Help, you can specify the topic description you want to display for each context-sensitive link. When you use a TopicAlias marker to create context-sensitive links, Eclipse creates a `contexts.xml` file that lists all of the context IDs for the Eclipse Help system you created using TopicAlias markers. In the `context.xml` file, Eclipse also provides a description of the context-sensitive link. By default, the description Eclipse provides for the context-sensitive link is the text of the first paragraph of the topic. However, if you want to specify a different description for the context-sensitive link, you can do this by using the TopicDescription marker.

To enable specifying topic descriptions for context-sensitive help topics in Eclipse Help, you need to enable the TopicDescription marker. By default, ePublisher sets the **Marker type** option for a marker named TopicDescription to **TopicDescription**. You can create a marker with a different name and set the **Marker type** option for that marker to **TopicDescription**. Then, writers can use this marker in the source documents to specify topic descriptions for each of context-sensitive help topics in their project.

### **To assign topic description behavior to topic description markers**

- 1.** Open your Stationery design project.
- 2.** On the **View** menu, click **Style Designer**.
- 3.** In the **Marker Styles**, select the marker style you want to modify.
- 4.** On the **Options** tab, set **Marker type** to **TopicDescription**.

# Customizing Oracle Help and Sun JavaHelp

[Defining the Navigation Pane in Oracle Help](#)  
[Using Custom Windows in Oracle Help](#)  
[Defining the Navigation Pane in Sun JavaHelp](#)  
[Using Context-Sensitive Help in Oracle Help and Sun JavaHelp](#)

You can customize the appearance and behavior of Oracle Help and Sun JavaHelp in several ways. For example, you can customize which buttons are included in the toolbar pane. The following sections describe these Oracle Help and Sun JavaHelp-specific customizations.

## Defining the Navigation Pane in Oracle Help

You can specify whether to include the Search tab in your Oracle Help output.

## To specify whether to include the Search tab in your Oracle Help

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. In the **Oracle Help** category, set **Enable Search tab** to **Enabled** or **Disabled**.
4. Click **OK** to save the target settings.

## Using Custom Windows in Oracle Help

By default, Oracle Help uses the standard Oracle Help viewer. You can modify the size, position, and other characteristics of the Oracle Help windows. You can also define and use custom windows in your Oracle Help project. To define custom windows, you need to override the `template.hs` file, which is used to create your helpset `.hs` file.

Once you have defined custom windows in a project, you can assign topics to them by adding the `WindowType` marker in specific topics in your source documents. When you assign a topic to a custom window, the topic is displayed in that window whenever users view the topic.

## To override the template.hs file in Oracle Help

1. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see "Stationery, Projects, and Overrides".
2. ***If you want to override the default settings for all Oracle Help targets in the project***, create the `Formats\Oracle Help\Pages` folder in your project folder.
3. ***If you want to override the default settings for a specific target***, create the `Targets\targetname\Pages` folder in your project folder, where *targetname* is the name of the target you want to override.
4. Copy the `template.hs` file from the following folder to the override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats\Oracle Help  
\Pages
```

5. Open the `template.hs` file you copied to your project override folder.
6. Modify the `template.hs` file as needed. For more information about this file, see the Oracle Help documentation.
7. Save and close the `template.hs` file.
8. Regenerate your output and review the results.

## Defining the Navigation Pane in Sun JavaHelp

By default, the navigation pane includes the Contents, Index, Search, and Favorites tabs. You can select whether to include the Favorites, Glossary, and Search tabs in your output. You can also define your own views in Sun JavaHelp by overriding the `template.hs` file. For example, you can create an alternate list of topics and provide that list as a separate view.

## To specify which tabs to include in your Sun JavaHelp

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. In the **Java Help** category, set each tab you want to include to **Enabled**.
4. Click **OK** to save the target settings.

# Using Context-Sensitive Help in Oracle Help and Sun JavaHelp

Context-sensitive help links allow you to open a specific help topic. For example, the **Help** button on a window in a software product can open a specific help topic that describes the window and provides links to related topics.

You can reference topics in Oracle Help and Sun JavaHelp using the file name or an internal identifier called a topic ID or topic alias. To use file names, use a Filename marker to assign a file name to a topic. Then, you can open that specific topic with that file name. However, if your file naming changes, you need to change the link to the topic. To use an internal identifier, use a TopicAlias marker to define the identifier for each topic. The benefit of using this approach is that it allows file names to change without impacting the links from the product. The writer inserts this marker in a topic and specifies a unique value for that topic. Then, Oracle Help and Sun JavaHelp use a mapping file that defines these topic aliases.

To simplify the coding of your source documents, you can use the same marker to define both the name and the topic alias for each topic file. In Style Designer, set the **Marker type** option for the marker you want to use to **Filename and topic alias**. However, if you change the value of this marker, you need to change the application that uses this value.



## To use context-sensitive help in Oracle Help and Sun JavaHelp

1. Meet with your application developers and define the topic ID for each context-sensitive help topic. Also discuss how ePublisher generates the mapping file.
2. In your source documents, use TopicAlias markers to identify the topic ID for each topic.
3. Generate output from your project and test your context-sensitive help links.

## Mapping Files in Oracle Help and Sun JavaHelp

Implementing context-sensitive help requires cooperation between the help author and the developer of the application that displays the context-sensitive help topics. You both need a mapping `.jhm` file that associates topic IDs with the target URL. When an application calls a context-sensitive help topic, it uses the topic ID to display the correct topic. Therefore, both the help and the application must use the same mapping file. If the topic IDs and target URLs do not match, the application displays either the wrong topic or no topic.

The mapping `.jhm` file lists topic IDs and target URLs. This mapping file is also referred to as a header file. For Oracle Help and Sun JavaHelp, the mapping file is similar to the following sample file:

```
<mapID target="ch1_htm_999374" url="ch1.htm#999374">
<mapID target="ch2_htm_999640" url="ch2.htm#999640">
<mapID target="ch9_htm_999786" url="ch9.htm#999786">
```

In this example, `ch1_htm_99374` is a topic ID, and `ch1.htm#99374` is the target URL for this particular topic ID. The marker in the appropriate topic has the `ch1_htm_999374` value.

When you implement context-sensitive help, you need to work with your application developers to decide how to choose the topic ID for each context-sensitive help topic. You can choose a set of topic IDs and embed them in your source documents using TopicAlias markers. When you generate output, ePublisher generates a mapping file using those topic IDs and assigns the target URL to each topic ID based on your source documents. You can provide the generated mapping file to your application developers, who can embed the topic IDs in the application code. ePublisher generates an updated file each time you generate the help. Remember to give the updated file to your application developers each time.

**Note:** Once you choose a set of topic IDs, embed them in your source documents using TopicAlias markers and do not change them.

# Testing Context-Sensitive Oracle Help and Sun JavaHelp

The best way to verify that your context-sensitive help topics function correctly is to test the help system with the application that displays the help topics. This testing process ensures the whole implementation works correctly.

# Customizing WebWorks Help

[Renaming the Top-Level Entry File](#)

[Selecting a Theme](#)

[Customizing the Splash Page in WebWorks Help](#)

[Customizing the Toolbar in WebWorks Help](#)

[Customizing the Navigation Pane in WebWorks Help](#)

You can customize the appearance and behavior of WebWorks Help in several ways. For example, you can customize the colors and images, and toolbar buttons displayed in the browser. You can also define which buttons are included in the toolbar pane. The following sections describe these WebWorks Help-specific customizations.

## Renaming the Top-Level Entry File

When you generate output for your project, a top-level entry file is created. This file defines the frameset for the help and gives the user a file to open that displays the complete help. By default, the top-level entry file is named `index.html`, but you can specify any name you need for the top-level entry file as long as you specify an `.html` or `.htm` extension.

### To rename the top-level file

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. In **Top level filename**, type the name you want to use for the top-level entry file.
4. Click **OK**.
5. Regenerate the project and review the results.

## Selecting a Theme

In WebWorks Help, you can select a theme (skin) to define the appearance of your help. ePublisher provides many different WebWorks Help themes. Each theme provides a custom skin, which defines the toolbar color, button images, navigation pane, tabs, and splash image.

### To choose a theme for WebWorks Help

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. Click on the **WebWorks Help** setting to reveal its contents.
4. In **Theme**, select the appropriate value.
5. Click **OK**.
6. Regenerate the project and review the results.

You can further customize your theme to deliver the specific appearance you want. For example, you can override images in your selected theme.

## Customizing the Splash Page in WebWorks Help

You can replace or remove the splash page that is displayed while your WebWorks Help opens. You can customize the splash page in the Stationery, and then writers can override this customization in each project, as needed.

### Replacing the Splash Image

The splash page is the first page that displays in the topic pane when the help set launches initially. By default, WebWorks Help systems display the WebWorks splash image. You can replace the default splash image with a custom image.

## To replace the splash page image

1. Identify the theme in use for your WebWorks Help target that you want to modify. For more information, see “Selecting a Theme”.
2. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see “Stationery, Projects, and Overrides”.
3. **If you want to override the image for all WebWorks Help targets**, create the **Formats** \WebWorks Help 5.0\Pages\images folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
4. **If you want to override the image for one WebWorks Help target**, create the **Targets** \WebWorks Help 5.0\Pages\images folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
5. Copy the `splash.jpg` file from the following folder to the `images` override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\WebWorks Help 5.0\ Pages\images
```

6. Open the `splash.jpg` file you copied to your project override folder and modify it to be the splash page image you want.
7. Save and close the `splash.jpg` file.
8. Regenerate your project to review the changes.

## Removing the Splash Page

When WebWorks Help opens, it displays the splash page. However, instead of displaying the splash page, you can configure WebWorks Help to display the first topic in the help.

### To remove the splash page

1. On the **Project** menu, select the **Active Target** you want to specify settings for.
2. On the **Target** menu, click **Target Settings**.
3. Set **Show first document instead of splash page** to **Enabled**.
4. Click **OK**.
5. Regenerate the project and review the results.

## Customizing the Toolbar in WebWorks Help

You can customize the toolbar pane in WebWorks help to provide the buttons and options you need. You can add, remove, and replace toolbar buttons. You can also customize the appearance of the toolbar pane. For more information about the toolbar pane, see “Toolbar Pane in WebWorks Help”.

## Adding and Removing Toolbar Buttons in WebWorks Help

You can add and remove toolbar buttons from the toolbar pane in WebWorks Help. To add a button, set the format setting for that button to **Enabled** for your WebWorks Help target. To remove a button, set the format setting for that button to **Disabled** for your WebWorks Help target.

### **To add or remove one or more toolbar buttons from the toolbar pane**

- 1.** On the **Project** menu, select the **Active Target** you want to specify settings for.
- 2.** On the **Target** menu, click **Target Settings**.
- 3.** Set the following target settings to **Enabled** or **Disabled** to define whether a button is displayed in the toolbar pane:



<b>Button</b>	<b>Format Setting to Modify</b>
Show in Contents	<b>Automatically synchronize in TOC</b>
Bookmark	<b>Show bookmark toolbar button</b>
PDF	<b>Show PDF button</b>
Previous and Next browse buttons	<b>Show previous &amp; next toolbar buttons</b>
Print	<b>Show print toolbar button</b>
Related Topics	<b>Show related topics toolbar button</b>

4. Click **OK**.

## Replacing the Toolbar Buttons in WebWorks Help

WebWorks Help provides several default buttons. These buttons allow the user to navigate through the help, and they provide additional features, such as printing a page from the help or emailing a link to a topic page. The toolbar buttons are `.gif` images for which you can provide override files. You can replace these standard buttons with other `.gif` images.

## To replace one or more toolbar buttons in WebWorks Help

1. Identify the theme in use for your WebWorks Help target that you want to modify. For more information, see “Selecting a Theme”.
2. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see “Stationery, Projects, and Overrides”.
3. ***If you want to override the images for all WebWorks Help targets,*** create the `WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common\images` folder in your *projectname* **Formats** folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
4. ***If you want to override the images for one WebWorks Help target,*** create the `WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common\images` folder in your *projectname* **Targets** folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
5. Paste the `.gif` files you want to use with names identical to those you want to replace in the `images` folder you created. The following table lists several default button images and their file names. For a complete list of image file names, see the appropriate folder in the ePublisher installation folder within `Program Files`. You can copy the files from the installation folder and then modify them in your project.

Button	File Name	Description
Show in Contents	<code>sync.gif</code>	Highlights the currently displayed topic in the table of contents.
Show in Contents (not available)	<code>syncx.gif</code>	Displayed when a topic is not currently displayed in the topic pane.
Previous	<code>prev.gif</code>	Displays the previous topic in the help.
Previous (not available)	<code>prevx.gif</code>	Displayed when there is no previous HTML document available.
Next	<code>next.gif</code>	Displays the next topic in the help.
Next (not available)	<code>nextx.gif</code>	Displayed when there is no next HTML document available.

6. Regenerate your project to review the changes.

## Changing the Background Color of the Toolbar

The toolbar in WebWorks Help is composed of a single repeating `.gif` image. This image is located in the images folder of your WebWorks Help theme. To modify the background color of the toolbar, you can override the `toolsbg.gif` file.

## To change the background color of the toolbar

1. Identify the theme in use for your WebWorks Help target that you want to modify. For more information, see “Selecting a Theme”.
2. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see “Stationery, Projects, and Overrides”.
3. **If you want to override the image for all WebWorks Help targets**, create the `WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common\images` folder in your *projectname* **Formats** folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
4. **If you want to override the image for one WebWorks Help target**, create the `WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common\images` folder in your *projectname* **Targets** folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
5. Copy the `toolsbg.gif` file from the following folder to the `images` override folder you created within your project folder:  
  
`Program Files\WebWorks\ePublisher Designer\Formats  
WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common  
images`
6. Open the `toolsbg.gif` file you copied to your project override folder and modify it to be the background image of the toolbar pane.
7. Save and close the `toolsbg.gif` file.
8. Regenerate your project to review the changes.

## Customizing the Navigation Pane in WebWorks Help

You can customize the navigation pane in several ways. For more information about the navigation pane, see “Navigation Pane in WebWorks Help”.

# Setting the Initial Width of the WebWorks Help Navigation Pane

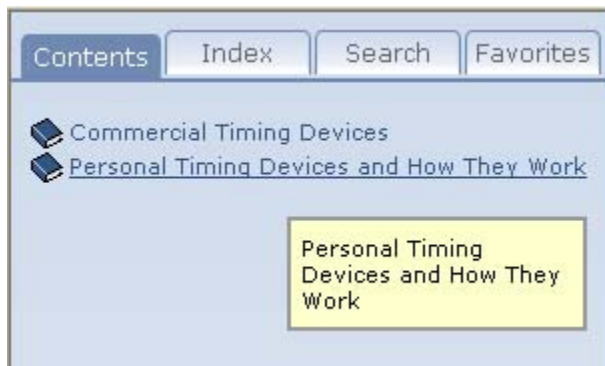
The navigation pane provides the Contents, Index, Search, and Favorites tabs. When the WebWorks Help opens, the initial width of the navigation pane is 300 pixels by default. You can override the `wwhelp.htm` file to define the initial width of the navigation pane.

## To set the initial width of the navigation pane

1. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see "Stationery, Projects, and Overrides".
2. **If you want to override the initial width for all WebWorks Help targets**, create the `Formats\WebWorks Help 5.0\Files\wwhelp\wwhimpl\js\html` folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
3. **If you want to override the initial width for one WebWorks Help target**, create the `Targets\WebWorks Help 5.0\Files\wwhelp\wwhimpl\js\html` folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
4. Copy the `wwhelp.htm` file from the following folder to the `html` override folder you created within your project folder:  
  
`Program Files\WebWorks\ePublisher Designer\Formats\WebWorks Help 5.0\ Files\wwhelp\wwhimpl\js\html`
5. Open the `wwhelp.htm` file you copied to your project override folder.
6. Find the following line of code and modify the `300` to be the number of pixels wide you want the navigation pane to use as its initial width.  
  
`<frameset cols="300,*" onLoad ...`
7. Save and close the `wwhelp.htm` file.
8. Regenerate your project to review the changes.

## Controlling the Navigation Pane Hover Text Appearance

Hover text refers to the popup window that WebWorks Help displays when you mouseover (hover over) a link on the Contents tab, as shown in the following figure.



You can adjust the appearance of the hover text by overriding the `wwhelp_settings.xml` file. This file allows you to modify the font, font color, background color, and border color of the hover text and its popup window. You can also disable the hover text.

## To change the navigation pane hover text appearance

1. Identify the theme in use for your WebWorks Help target that you want to modify. For more information, see "Selecting a Theme".
2. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see "Stationery, Projects, and Overrides".
3. **If you want to override the hover text settings for all WebWorks Help targets**, create the `Formats\WebWorks Help 5.0\Skins\theme` folder in your *projectname* folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
4. **If you want to override the hover text settings for one WebWorks Help target**, create the `Targets\WebWorks Help 5.0\Skins\theme` folder in your *projectname* folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
5. Copy the `wwhelp_settings.xml` file from the following folder to the *theme* override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\WebWorks Help 5.0\ Skins\theme
```

6. Open the `wwhelp_settings.xml` file you copied to your project override folder.
7. Find the following block of code:

```
<HoverText enable="true">  
...  
</HoverText>
```

8. **If you want to disable the hover text**, replace `<HoverText enable="true">` with `<HoverText enable="false">`.
9. **If you want to change the font family of the hover text**, replace `font-family: Verdana, Arial, Helvetica, sans-serif` with the names of the font family you want to use for the text. Make sure you specify fonts that are installed by default.
10. **If you want to change the font size of the hover text**, replace `font-size: 8pt` with the size of the font you want to use for the text, such as `font-size: 10pt`.



- 11. If you want to change the font color of the hover text,** replace the color defined by `foreground="#000000"` with the RGB color value you want to use for the text.
- 12. If you want to change the background color of the popup window,** replace the color defined by `background="#FFFFCC"` with the RGB color value you want to use for the background.
- 13. If you want to change the border color of the popup window,** replace the color defined by `border="#999999"` with the RGB color value you want to use for the border.
- 14.** Save and close the `wwhelp_settings.xml` file.
- 15.** Regenerate your project to review the changes.

## Changing the Font Color on the Navigation Pane Tabs in WebWorks Help

The tabs in the navigation pane all share the same font and text style properties. These style properties are specified in the `wwhelp_settings.xml` file.

## To change the font color on all the tabs in the navigation pane

1. Identify the theme in use for your WebWorks Help target that you want to modify. For more information, see "Selecting a Theme".
2. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see "Stationery, Projects, and Overrides".
3. **If you want to override the tab font color for all WebWorks Help targets**, create the `Formats\WebWorks Help 5.0\Skins\theme` folder in your *projectname* folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
4. **If you want to override the tab font color for one WebWorks Help target**, create the `Targets\WebWorks Help 5.0\Skins\theme` folder in your *projectname* folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
5. Copy the `wwhelp_settings.xml` file from the following folder to the *theme* override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\WebWorks Help 5.0\ Skins\theme
```

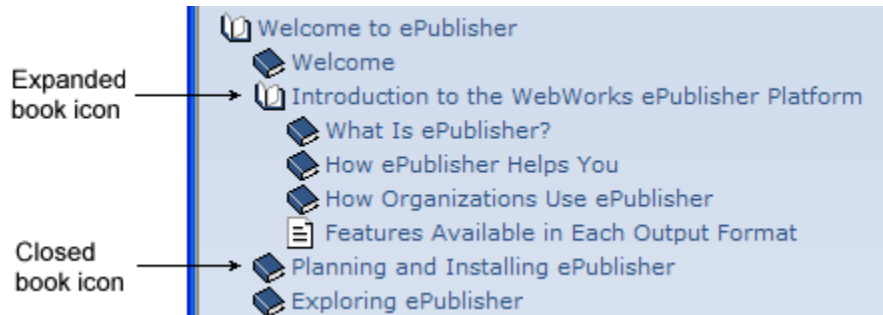
6. Open the `wwhelp_settings.xml` file you copied to your project override folder.
7. Find the following block of code:

```
<Tabs>  
  <DefaultColors foreground="#666666" />  
  <SelectedColors foreground="#FFFFFF" />  
</Tabs>
```

8. **If you want to change the color of the text on the unselected tabs**, replace `#666666` with the RGB color value you want to use for the text on those tabs.
9. **If you want to change the color of the text on the selected tab**, replace `#FFFFFF` with the RGB color value you want to use for the text on that tab.
10. Save and close the `wwhelp_settings.xml` file.
11. Regenerate your project to review the changes.

# Using Custom Icons on the Contents Tab in WebWorks Help

When you navigate through the table of contents on the Contents tab, you can see two types of icons for entries that have subentries. The open book icons (`fo.gif`) represent expanded table of contents levels, and the closed book icons (`fc.gif`) represent unexpanded table of contents levels, as shown in the following figure.



These icons are `.gif` images stored in the WebWorks Help theme folder. You can create override files for the `fo.gif` and `fc.gif` files to replace these files and use your own custom table of content icons instead.

## To replace the book icons used on the Contents tab

1. Identify the theme in use for your WebWorks Help target that you want to modify. For more information, see “Selecting a Theme”.
2. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see “Stationery, Projects, and Overrides”.
3. **If you want to override the images for all WebWorks Help targets**, create the `WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common\images` folder in your *projectname* **Formats** folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
4. **If you want to override the images for one WebWorks Help target**, create the `WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common\images` folder in your *projectname* **Targets** folder, where *theme* is the name of the WebWorks Help theme you want to override, such as `Lobby_Blue`, and *projectname* is the name of your ePublisher project. If the theme name is two words, include an underscore instead of a space between the words.
5. Copy the `fo.gif` and `fc.gif` files from the following folder to the `images` override folder you created within your project folder:  
  
`Program Files\WebWorks\ePublisher Designer\Formats  
WebWorks Help 5.0\Skins\theme\Files\wwhelp\wwhimpl\common  
images`
6. Open the `fo.gif` and `fc.gif` files you copied to your project override folder and modify these files to be the book icons you want to use.
7. Save and close the `fo.gif` and `fc.gif` files.
8. Regenerate your project to review the changes.

## Modifying the Appearance of the Search Message in WebWorks Help

In WebWorks Help, when users click on the Search tab, the message “Type in the word(s) to search for:” is displayed. Once the user types in a word to search for and clicks **Go**, the “(Searching)” message is briefly displayed in the search results box. The appearance of these messages is controlled by the `h2` and `h3` HTML tags, and these tags are located in the `search.js` file. To modify the appearance of the

search message, you need to specify the style properties for `h2` and `h3` tags in a separate `.css` file.

**Note:** *If you customize the appearance of the search message by modifying the `search.js` file*, you will be responsible for maintaining your customizations as needed each time you update your Stationery to work with a new version of ePublisher.

For more information about override files and locations, see “Stationery, Projects, and Overrides”.

To change the appearance of the “Type in the word(s) to search for” message, create a style definition for the `h2` tag. To modify the appearance of the “(Searching)” message, create a style definition for the `h3` tag. After you specify the font properties in a `.css` file for the `h2` and `h3` tags, reference the `.css` file in the `search.js` file.

**Note:** The `messages.xml` file in the `Formats\WebWorks Help 5.0\Files\wwhelp` folder controls the message text used throughout WebWorks Help for multiple languages. You can override this file to provide customized messages for your specific needs.

## To modify the appearance of the search messages

1. **If you want to override the message appearance for all WebWorks Help targets**, complete the following steps:

- a. In your Stationery design project, on the **View** menu, click **Format Override Directory**.
- b. Create the following folder in your *projectname* folder, where *projectname* is the name of your ePublisher project:

```
Formats\WebWorks Help 5.0\Files\wwhelp\wwhimpl\js\scripts
```

2. **If you want to override the message appearance for one WebWorks Help target**, complete the following steps:

- a. In your Stationery design project, on the **View** menu, click **Target Override Directory**.
- b. Create the following folder in your *projectname* folder, where *projectname* is the name of your ePublisher project:

```
Targets\WebWorks Help 5.0\Files\wwhelp\wwhimpl\js\scripts
```

3. Copy the `search.js` file from the following folder to the `scripts` override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\WebWorks Help 5.0\ Files\wwhelp\wwhimpl\js\scripts
```

4. Open the `search.js` file you copied to your project override folder.
5. Find the following block of code:

```
// Display initializing message  
//  
  
HTML.fAppend("<h2>" +  
WWHFrame.WWHJavaScript.mMessage.mInitializingMessage + "</h2>"  
"\n");
```

6. Replace that block of code with the following code:

```
// Display searching message  
//  
  
HTML.fAppend("<h2 style=\"color:#FF00FF; text-  
decoration: underline; font-size: 12pt;\">" +
```

```
WWHFrame.WWHJavaScript.mMessages.mSearchSearchingMessage + "</h2>\n");
```

7. Find both instances of the following block of code:

```
// Display searching message
//

HTML.fAppend("<h2>" +
WWHFrame.WWHJavaScript.mMessages.mSearchSearchingMessage + "</h2>\n");
```

8. Replace each instance of that block of code with the following code:

```
// Display searching message
//

HTML.fAppend("<h2 style=\"color:#FF00FF; text-decoration: underline; font-size: 12pt;\">" +
WWHFrame.WWHJavaScript.mMessages.mSearchSearchingMessage + "</h2>\n");
```

9. Find the following block of code:

```
// Display search message and/or prepare results for display
//
if (this.mSavedSearchWords.length == 0)
{
HTML.fAppend("<h3>" +
WWHFrame.WWHJavaScript.mMessages.mSearchDefaultMessage + "</h3>\n");
}
```

10. Replace that block of code with the following code:

```
// Display search message and/or prepare results for display
//
if (this.mSavedSearchWords.length == 0)
{
HTML.fAppend("<h3 style=\" color:#FF00FF; text-decoration: none; font-size: 14pt;\">" +
WWHFrame.WWHJavaScript.mMessages.mSearchDefaultMessage + "</h3>\n");
}
```

11. Save and close the `search.js` file.

12. Regenerate your project to review the changes.

## Modifying the Search Ranking

The search results are displayed in the Search tab when a user types a word to search for and clicks **Go**. The search results are sorted by the relevancy ranking, which is calculated based on the scoring preference defined for the HTML tags in the `wwhelp_files.xml` file. By default, WebWorks Help assigns relevancy rankings based on where in a topic a particular item is found.

For example, if you set the scoring preference, or weight, for Heading 1 to 25 and you set the weight for Heading 2 to 15, then any search term found in a Heading 1 returns a higher relevancy ranking than if the search term is found in a Heading 2.

The following scenario illustrates how the relevancy ranking is calculated:

If you search for the word `popup`, and that word appears in both a Heading 1 and a Heading 2 on one page, and in a Heading 1 on another page, the search returns two results sorted by the relevancy ranking.

To get the relevancy ranking, WebWorks Help takes the score of each search result and divides it by the highest score found. To get the score of each search result, the scoring preference for each HTML tag is used.

For example, Heading 1 has a scoring preference of 25 and Heading 2 has a scoring preference of 15. If the word `popup` appears in a Heading 1 and a Heading 2 on a topic page, then the score for that word on that page is  $25+15=40$ . If the word `popup` appears in a Heading 1 on another topic page, then the score for that word on that page is 25. Therefore, if 40 is the highest score that a search for `popup` returns, the relevancy ranking of the first search result is  $100\%=40/40\times 100\%$ , and the relevancy ranking for the second search result is  $38\%=15/40\times 100\%$ .



## To modify the relevancy ranking for search results

1. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see "Stationery, Projects, and Overrides".
2. **If you want to override the message appearance for all WebWorks Help targets**, create the `Formats\WebWorks Help 5.0\Transforms` folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
3. **If you want to override the message appearance for one WebWorks Help target**, create the `Targets\WebWorks Help 5.0\Transforms` folder in your *projectname* folder, where *projectname* is the name of your ePublisher project.
4. Copy the `wwhelp_files.xml` file from the following folder to the `Transforms` override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\WebWorks Help 5.0\ Transforms
```

5. Open the `wwhelp_files.xml` file you copied to your project override folder.
6. Find the following block of code:

```
<ScoringPrefs>  
  <meta name="keywords" weight="100"/>  
  <meta name="description" weight="50"/>  
  <meta name="summary" weight="50"/>  
  <title weight="20"/>  
  <h1 weight="15"/>  
  <h2 weight="10"/>  
  <caption weight="10"/>  
  <h3 weight="7"/>  
  <th weight="5"/>  
  <h4 weight="5"/>  
  <h5 weight="4"/>  
  <h6 weight="3"/>  
  <h7 weight="2"/>  
</ScoringPrefs>
```

7. Modify the weight attributes for any tags, such as `h1` and `h2`, you want to change.
8. Save and close the `wwhelp_files.xml` file.
9. Regenerate your project to review the changes.

# Modifying the Search Highlighting

You can control search highlighting with an override:

```
Formats\WebWorks Help 5.0\Skins\[skin name]\wwhelp_settings.xml
```

For more information regarding overrides, See "Creating Format Overrides".

The following markup controls the behavior:

```
<Search enable="true">

<Results showrank="true" />

<Highlighting enable="true">

    <Colors foreground="#FFFFFF" background="#333399" />

</Highlighting>

</Search>
```

Using an HTML value for the color will help you control the search highlighting color for the text and for the background of the word that is being searched in the output page.

**Note:** This browser behavior is limited to Internet Explorer.

## Synonyms

By defining a list of synonyms, you can increase the effectiveness of a search by grouping similar words together as a single result. For example, you define movie, video, and avi as synonyms. If a user searches for one of those terms, a match will be scored if any of the three are found in a document.

To add these definitions, you'll need to perform an override on a file called synonyms.xml, located by default in the ePublisher Designer installation directory here:

```
\Formats\WebWorks Help 5.0\Files\wwhdata\common
```

Once the file is copied to the correct location in your project directory, open it in any text- or XML-editing application (e.g., NotePad), and add entries to the `<WebWorksSynonyms>` section. For example, your synonyms might look like this:

```

<WebWorksSynonyms>

  <Word value="movie">

    <Synonym value="avi" />

    <Synonym value="video" />

  </Word>

  <Word value="picture">

    <Synonym value="image" />

    <Synonym value="graphic" />

    <Synonym value="photo" />

  </Word>

</WebWorksSynonyms>

```

Please note that the Synonyms feature does not recognize multiple word values or synonyms. And, it ignores words (and synonyms) of 2 letters or less. Also, its search logic is quite literal. That is, it finds only what you type, and only if the exact form is in the text. For example, it does not find plurals or phrases, and it even includes punctuation.

## Minimum word length & common words

When performing a text search, the WebWorks Help 5.0 search engine follows a couple of guidelines designed to make the search more efficient and the results more helpful. By default, the minimum length of a word in the search results is 3 letters. ePublisher Designer also comes with a list of terms which appear commonly in many types of documents, and are therefore left out of search results. You can modify both the minimum length and the list of common words by performing a target override on a file called `locales.xml`, located by default in the ePublisher Designer installation directory, here:

```
2024.1\Formats\Shared\common\locale
```

Because this file is in the Shared formats directory, your project folder hierarchy should look like this:

```
MyProject\Targets\MyWWHelpTarget\Shared\common\locale
```

**To change the minimum number of letters in a WebWorks Help 5.0 search result:**

- 1.** Open locales.xml in an XML- or text-editing application (e.g., NotePad).
- 2.** Find the Locale used by your project (default English is "en").
- 3.** Under the <Search> section, change the MinimumWordLength value to the desired number of characters in the shortest word the search should return.

## To change the list of words which will not be returned in a WebWorks Help 5.0 search:

1. Open locales.xml in an XML- or text-editing application (e.g., NotePad).
2. Find the Locale used by your project (default English is "en").
3. Under the <Search> section, modify the <StopWords> values to reflect the list of words which should not appear in your project's search results.

## Using Context-Sensitive Help in WebWorks Help

Context-sensitive help links provide content based on the context of what the user is doing. In many cases, this help content is based on the window that is open and active. For example, the **Help** button on a window in a software product can open a specific help topic that provides important information about the window and links to related topics.

WebWorks Help allows you to use a TopicAlias marker to define an internal identifier for each topic. The benefit of using an internal identifier is that it allows file names to change without impacting the links from the product. The writer inserts this marker in a topic and specifies a unique value for that topic. ePublisher creates a mapping file to identify each topic associated with a unique value. Then, WebWorks Help uses this internal identifier and the mapping file to display the correct topic. Before you can reference topics in WebWorks Help using topic aliases, you must enable TopicAlias markers in your Stationery. For more information, see "Defining Filename Markers for Context-Sensitive Help Links".

## Mapping Files in WebWorks Help

WebWorks Help does not generate a mapping file. However, you can see a list of defined contexts and topics in the `wwdata/xml/files.xml` file. You can also use the Topics Report to verify that context-sensitive help topics have been created for each topic ID specified in your source document. The Topics Report lists the topic ID and the topic file created for each topic ID. Topic aliases report. For more information about the topics report, see "Topics Reports".

## Opening Context-Sensitive Help in WebWorks Help using Standard URLs

You can open WebWorks Help from the application using standard URLs or the WebWorks Help API. For more information about the API, see "Opening Context-Sensitive Help with the WebWorks Help API".

**Note:** If you have WebWorks Help installed on the local computer, Internet Explorer 7 ignores the query string attached to a URL when it reloads a page. For example, after displaying a security warning or other message, Internet Explorer reloads the page. Without the query string, the browser cannot open the specific topic, since the group name and topic alias are not available. To avoid this issue with Internet Explorer 7, you can use the WebWorks Help API to open a topic in locally installed WebWorks Help.

To open the complete WebWorks Help output to the default help topic, open the `index.htm` or `index.html` file in the root of the folder where the WebWorks Help is stored. This file defines the help frameset and loads the individual components, such as the table of contents, index, and topic content.

To open a specific topic in WebWorks Help, use the following URL:

```
helplocation/wwhelp/wwhimpl/api.htm?context=groupname&topic=alias
```

The variable parts of this URL are defined as follows:

### ***helplocation***

Specifies the location where the WebWorks Help is installed. If the help is on a Web server, specify the location using the `http` protocol and the Web site path to the root of the help, such as `http://www.webworks.com/help`. If the help is installed on the local computer, specify the location using the file protocol and the path to the root of the help, such as `file:///C:/Program Files/YourApplication/help`.

### ***groupname***

Specifies the group context value for the top-level group in which the topic resides in Document Manager. This group context is specified in merge settings for each top-level group.

### ***alias***

Specifies the value of the TopicAlias marker in the topic to open.

## **Opening Context-Sensitive Help with the WebWorks Help API**

WebWorks Help provides an API that you can use to call context-sensitive topics in WebWorks Help. Using the API instead of simple URLs gives you increased flexibility and enhanced control. For example, the following list highlights some of the benefits of using the API:

- The API automatically determines the default browser and opens it to display the correct help topic.

- The API reduces the amount of code developers must write to integrate context-sensitive help. Without the API, developers must code their own COM interface to communicate with the browser on the Windows platform. This code is provided as part of the WebWorks Help API.
- Using the API can significantly reduce the time required to load an individual topic if the help system is already open in the browser. If your application calls a topic using a URL instead of the API, the entire frameset, including the WebWorks Help applet, is loaded each time the user opens help. If the application calls a topic using the API, and the correct help is already loaded in the browser, neither the frameset nor the applet is reloaded. Instead, the currently open topic pane displays the correct help topic, which delivers a significant performance improvement for WebWorks Help users.
- The API allows you to avoid the issue that exists with standard URLs in Internet Explorer 7. In Internet Explorer 7, if you have WebWorks Help installed on the local computer, Internet Explorer 7 ignores the query string attached to a URL when it reloads a page. For example, after displaying a security warning or other message, Internet Explorer reloads the page. Without the query string, the browser cannot open the specific topic, since the group name and topic alias are not available.

You can use the C/C++ API, which is available as a `.dll` or a COM object. This API supports WebWorks Help 4.0 and 5.0. WebWorks Help 5.0, does not include the Java navigation. Therefore, do not use the Java navigation options with WebWorks Help 5.0. For more information about this API and the software development kit, see <http://wiki.webworks.com/DevCenter/Projects/WebWorksHelp/WebWorksHelpSDK>.

## Opening Context-Sensitive Help with the Javascript API

Similar to using the WebWorks Help API, you can use the Javascript API when working with web applications designed to run on websites using standard HTML and Javascript. For complete details of the WebWorks Help Javascript API, see <http://wiki.webworks.com/DevCenter/Projects/WebWorksHelp/JavascriptAPI>.

# Advanced Format and Target Customizations

Understanding Customized Processing  
Format and Target Overrides  
Customizing Page Templates (\*.asp)

By defining online navigation, setting options and preferences in your project, and embedding additional information in your source documents, you can define the behavior and appearance of your online content. The following sections provide some additional information about techniques you can use to further refine and customize your design.

**Note:** These techniques are supported in all formats unless otherwise noted.

## Understanding Customized Processing

ePublisher uses file and folder locations to provide a structure where you can create and store override files. These files customize how ePublisher transforms your content.

When you generate output for an output format, ePublisher uses the following process to identify the files to use to process your content and complete the task:

1. ePublisher checks the **Targets** folder hierarchy in your project for the files required to complete the task. ePublisher checks the folder hierarchy named for the target you are generating.
2. If the files are not found in the **Targets** folder hierarchy in your project, ePublisher checks the **Formats** folder hierarchy in your project for the files required to complete the task. ePublisher checks the folder hierarchy named for the output format of the target you are generating.
3. If the files are not found in your project folder hierarchy, ePublisher checks the installation folders.

This process allows you to override any default file in the installation folder hierarchy for one or more output formats or targets by placing a customized file with the same name at the correct location in the project folder hierarchy.

By recreating the file path in the **Targets** folder in your project, and storing a modified file in the correct location, you can change the processing for that specific target. This method allows you customize a specific target in a project that has multiple targets using the same output format.

By recreating the file path in the **Formats** folder in your project, and storing a modified file in the correct location, you can change the processing for all targets



that use the same output format. This method is also helpful for projects with one target.

**Note:** Do not modify files in the installation folders. Store customized files only in the project folder hierarchy. You need to create the `Targets` and `Formats` folder hierarchies in your project folder, as needed for the files you want to override.

For more information, see “Stationery, Projects, and Overrides” and “ePublisher Pipeline and Transforms”.

## Format and Target Overrides

ePublisher gives you complete control over the final output. However, some project modifications cannot be made through the ePublisher console. In these instances, you may need to override the default XSL files used by ePublisher to achieve the results you need.

When ePublisher generates output for the first time for a project, ePublisher reviews the `format.wwfmt` file in the folder for the appropriate format. This file tracks the actions required to generate the desired format. If there are no user customizations, all the files referenced by the `format.wwfmt` file are in the Format or Transforms folders in the installation folder hierarchy. By default, ePublisher first checks the project folder hierarchy for each required file before getting the files from the installation folder. With this process, you can override one or more default files by creating the same folder hierarchy in a project folder and storing a customized file with the same name at the correct location in the project folder hierarchy.

**Note:** You can override default files for all targets of a specific output format type in the project. You can also override default files for a specific target in a project without affecting other targets of the same output format type in that project. Overrides for a target override any customizations you specified in the override folders for a format.

## Creating Format Overrides

This section defines how to override a default file for all targets of a specific output format type in the project. Do not modify the files in the installation folder hierarchy. These files are the default files and should remain as is. These files are also overwritten when you install new ePublisher releases. Instead, store and incorporate your customized files with your Stationery so your projects based on the Stationery get those customizations.

**Note:** Changes to `.xsl` and Format Trait Info `.fti` files are considered advanced customizations and are not supported by the Quadralay Product Support department. If you are familiar with XSL, you may find that customizations to `.xsl` and `.fti` files are a powerful way to achieve very specific results in

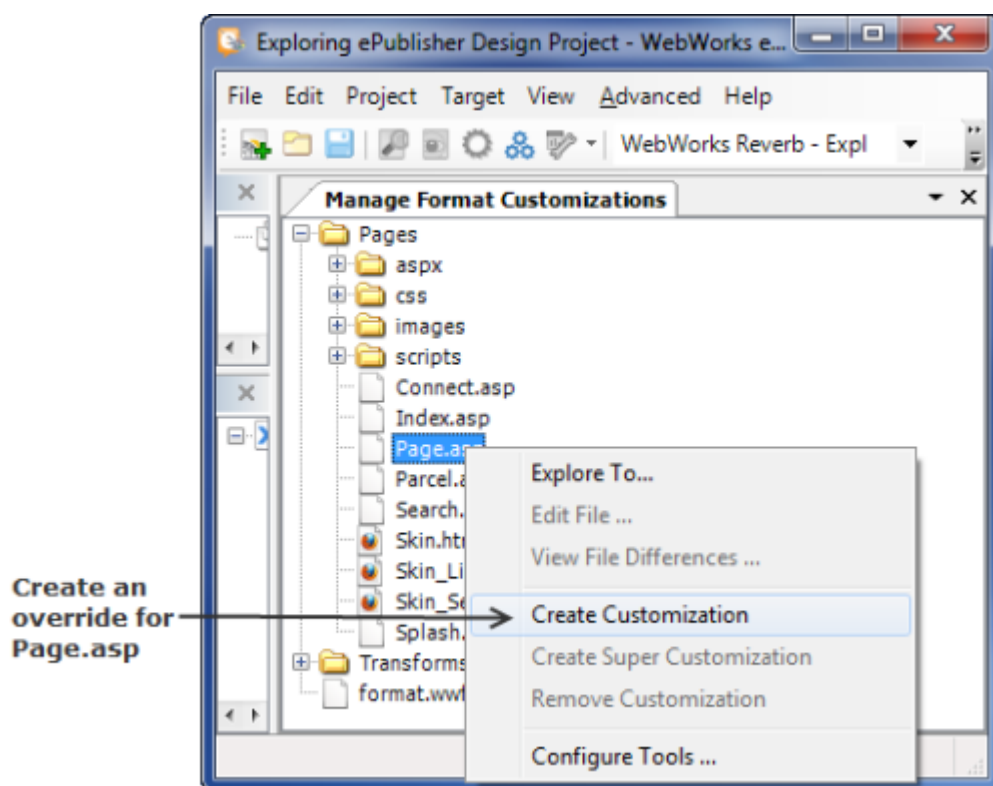
your generated output or output deployment. However, you are responsible for maintaining any `.xsl` or `.fti` file overrides that you implement. The Quadralay Product Support department does not provide support for these advanced customizations.

File names used in ePublisher are case sensitive. Make sure the file and folder names you create exactly match the default file and folder names in the installation folder hierarchy. Do not copy any files into the project folder hierarchy that you are not overriding.

The following task assumes that ePublisher is installed in the default location and your ePublisher projects are stored in the `My Documents` folder. If you installed ePublisher to a non-default location, or if you store your projects in a folder other than the `My Documents` folder, adjust the paths in the following task.

## To override a project format

1. In your Stationery design project, on the **Advanced** menu, click **Manage Format Customizations**.
2. In the displayed collection of folders and files, navigate to the file that you wish to override and highlight it.
3. To create an override for the selected file, right click and select the **Create Customization** menu. This will create an exact copy of the original file and place it into the appropriate location within the project directory. Now you can safely modify this file as desired as well as viewing the file differences between your override and the original.



4. Use either of the right-click mouse menu items: **Explore To...** or **Edit File ...** to access and modify your file override.

The next time you generate your project, the modified file automatically overrides the default file and the changes you made are incorporated into the output for all targets that produce that output format.

**Note:** When you save the ePublisher project as Stationery, the project format overrides you have created are saved with your Stationery. This Stationery can then be used to create future projects in ePublisher Express and ePublisher AutoMap.

# Creating Target Overrides

This section defines how to override default files for a specific target in a project without affecting other targets of the same output format type in that project. Do not modify the files in the installation folder hierarchy. These files are the default files and should remain as is. These files are also overwritten when you install new ePublisher releases. Instead, store and incorporate your customized files with your Stationery so your projects based on the Stationery get those customizations.

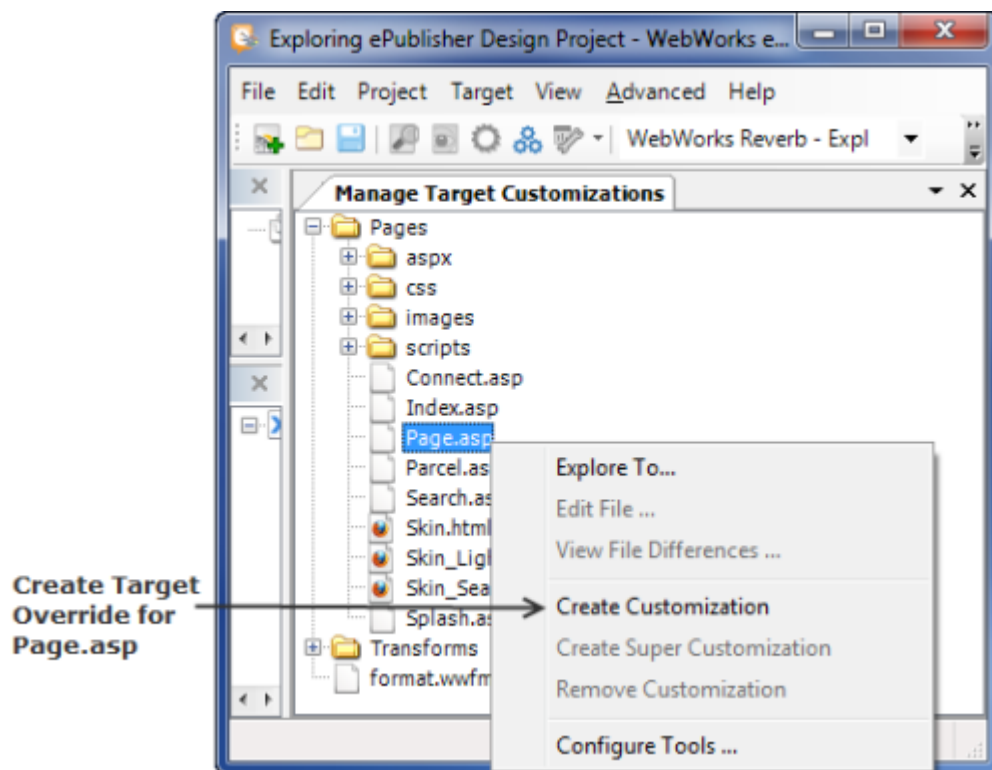
**Note:** Changes to `.xsl` and Format Trait Info `.fti` files are considered advanced customizations and are not supported by the Quadralay Product Support department. If you are familiar with XSL, you may find that customizations to `.xsl` and `.fti` files are a powerful way to achieve very specific results in your generated output or output deployment. However, you are responsible for maintaining any `.xsl` or `.fti` file overrides that you implement. The Quadralay Product Support department does not provide support for these advanced customizations.

Make sure the file and folder names you create exactly match the default file and folder names in the installation folder hierarchy. Do not copy any files into the project folder hierarchy that you are not overriding.

The following task assumes that ePublisher is installed in the default location and your ePublisher projects are stored in the `My Documents` folder. If you installed ePublisher to a non-default location, or if you store your projects in a folder other than the `My Documents` folder, adjust the paths in the following task.

## To override a project target

1. In your Stationery design project, on the **Advanced** menu, click **Manage Target Customizations**.
2. In the displayed collection of folders and files, navigate to the file that you wish to override and highlight it.
3. To create an override for the selected file, right click and select the **Create Customization** menu. This will create an exact copy of the original file and place it into the appropriate location within the project directory. Now you can safely modify this file as desired as well as viewing the file differences between your override and the original.



4. Use either of the right-click mouse menu items: **Explore To...** or **Edit File ...** to access and modify your file override.

The next time you generate your project, the modified file automatically overrides the default file and the changes you made are incorporated into the output for that target. These customizations also override any override files you saved in the project folder hierarchy for the associated output format.

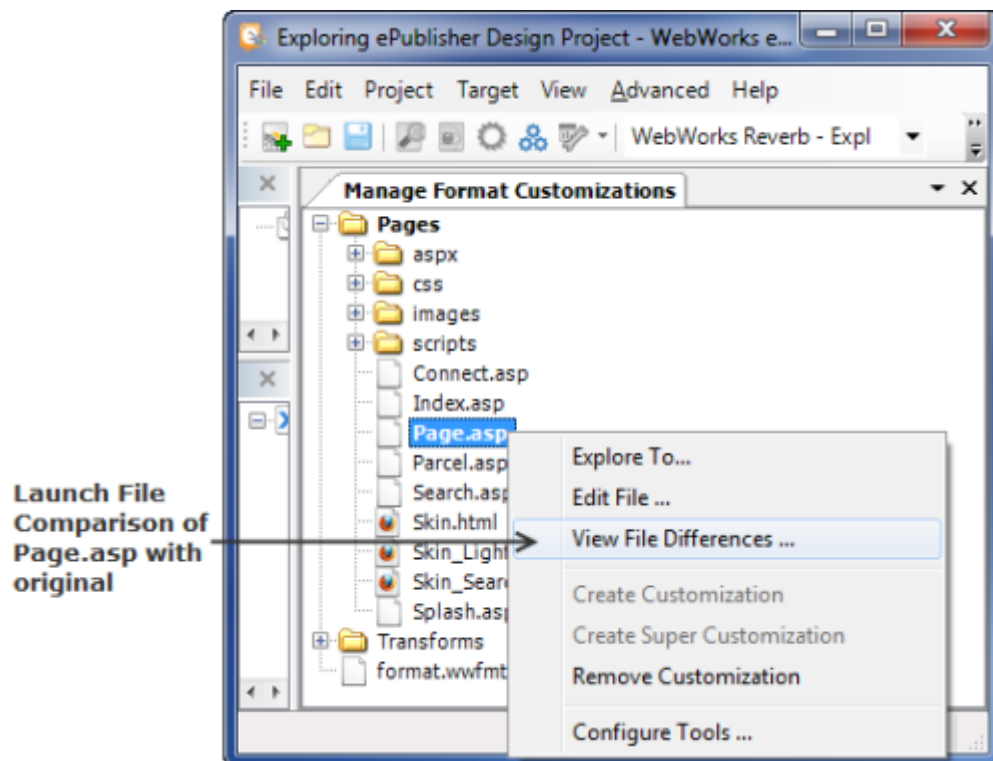
**Note:** When you save the ePublisher project as Stationery, the project target overrides you have created are saved with your Stationery. This Stationery can then be used to create future projects in ePublisher Express.

# Managing Overrides

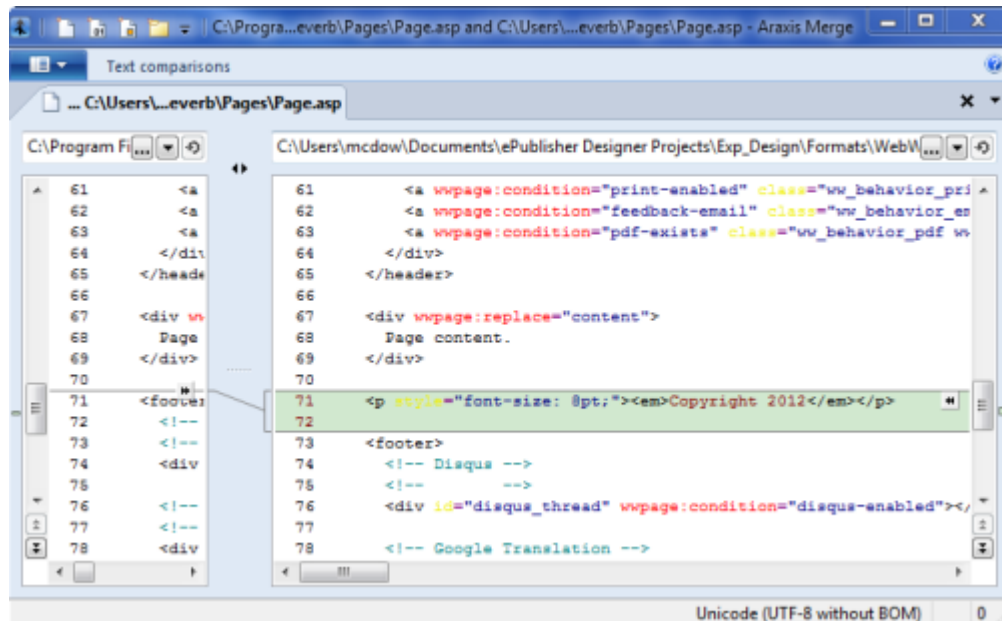
ePublisher assists you in the management of your project's overrides by graphically highlighting files that have been overridden as well as highlighting folders that contain files that have been overridden. In addition, you can also have ePublisher launch a 3rd-party *Diff* (also called a line-comparison) tool that will intuitively display what modifications have been made in your overridden file.

## To View File Differences

1. In your Stationery design project, on the **Advanced** menu, click **Manage Format Customizations** (or Target for target overrides).
2. In the displayed collection of folders and files, navigate to the overridden file that you wish to compare and highlight it.
3. To compare the differences of the selected override, right click and select the **View File Differences...** menu. This will launch your configured *Diff* program and automatically display the differences between your file and the original.



4. You can examine how your override is different from the original. In addition, you can modify your overridden file directly from within the *Diff* tool.



## Customizing Page Templates (\*.asp)

You can customize the appearance of your content page to meet your specific needs. ePublisher provides several customization options, such as where to display company information, browse buttons, and breadcrumbs. These options allow you to achieve a professional result in your generated output.

You may want to further customize your content page design. ePublisher provides this flexibility by allowing you to override the `Page.asp` file, which defines the appearance and behavior of your content page. However, if you customize this file, some customization options in the ePublisher console may no longer affect your content page design.

For example, you can override the `Page.asp` file to add a graphic bar across the top of the page with the product logo and name. You can also add a graphic bar across the bottom of the page with a copyright and a link to your company Web site and customer support area. You can also modify the formatting and layout of the default page design options, such as the company information options.



## To override the Page.asp file

1. In your Stationery design project, on the **View** menu, click **Project Directory**. For more information about override files and locations, see "Stationery, Projects, and Overrides".
2. **If you want to override the processing for an output format**, create the `Formats\formattype\Pages` folder in your project folder, where *formattype* is the name of the output format you want to override, such as `Dynamic HTML`.
3. **If you want to override the processing for a target**, create the `Targets\targetname\Pages` folder in your project folder, where *targetname* is the name of the target you want to override.
4. Copy the `Page.asp` file from the following folder to the override folder you created within your project folder:

```
Program Files\WebWorks\ePublisher Designer\Formats  
\formattype\Pages
```

5. Open the `Page.asp` file you copied to your project override folder in a text editor.
6. Modify the `Page.asp` file as needed.
7. Save and close the `Page.asp` file.

## Page Templates Reference

Page templates are files that form the skeleton structure of generated output files. Often users will create overrides for these files so that they can precisely control the look and function of generated output files. The most common type of page templates to customize are the `Page.asp` and `Splash.asp` files, which are used in most all of the output format types.

## Namespace and Attributes

ePublisher defines the namespace: `wwpage` for page templates as follows:

```
xmlns:wwpage="urn:WebWorks-Page-Template-Schema"
```

The following shows the page template attributes that can be used along with their purpose.

Attribute	Purpose
<code>wwpage:condition</code>	Controls when elements should be preserved in the generated output.
<code>wwpage:content</code>	Use this attribute to replace the <code>innerHTML</code> or <code>innerHTMLXML</code> of an element.
<code>wwpage:replace</code>	Use this attribute to replace the <code>innerHTML/innerHTMLXML</code> as well as the <code>outerHTML/outerXML</code> .
<code>wwpage:attribute- &lt;name&gt;</code>	Use this attribute to emit a generated attribute value with name: <code>name</code> . In addition, you can specify a value of either: <code>relative-to-output</code> or <code>copy-relative-to-output</code> or <code>wwsetting:&lt;target-setting-name&gt;</code> . Specifying the value using <code>relative-to-output</code> is useful for file type attributes that require a pathname relative to the parent file. If the <code>copy</code> version is used, it additionally copies the file into the correct location within the generated output directory.
<code>wwpage:NoBreak</code>	Use this attribute to collapse all white space between two elements into nothing.
<code>wwpage:Import</code>	Use this attribute to indicate an that an import operation will take place using one of the <code>*-from-*</code> <code>wwpage</code> attributes.
<code>wwpage:content-from-file</code>	Use this attribute to import the contents of the specified file as the <code>innerHTML</code> of an element.
<code>wwpage:replace-from-file</code>	Use this attribute to import the contents of the specified file as the <code>innerHTML</code> as well as the <code>outerHTML</code> of an element.
<code>wwpage:content-from-lookup</code>	Use this attribute to import the contents of the specified item name generated by ePublisher as the <code>innerHTML</code> of an element.

Attribute	Purpose
wwpage:replace-from-lookup	Use this attribute to import the contents of the specified item name generated by ePublisher as the <code>innerHTML</code> as well as the <code>outerHTML</code> of an element.

**Note:** If you are using `wwpage:attribute-<name>` to express attribute that uses a namespace of its own, you can use a "-" character in place of the ":" character. For example: `wwpage:attribute-xml-lang` is used to generate the `xml:lang` attribute in the output.

The following shows a typical usage of `wwpage` attributes within a `Page.asp` or `Splash.asp` page template.

Attribute	Example usage in Page.asp page template
wwpage:condition	<code>&lt;hr wwpage:condition="header-exists" align="left" /&gt;</code>
wwpage:content	<code>&lt;title wwpage:content="title"&gt;Title&lt;/title&gt;</code>
wwpage:replace	<code>&lt;div wwpage:replace="content"&gt;Page content.&lt;/div&gt;</code>
wwpage:attribute- <name>	<code>&lt;link rel="StyleSheet" href="css/print.css" wwpage:attribute-href="copy-relative-to-output" type="text/css" media="print" /&gt;</code>
wwpage:NoBreak	<code>&lt;a href="#"&gt; &lt;wwpage:NoBreak /&gt; &lt;img src="myimage.png"&gt; &lt;/a&gt;</code>
wwpage:Import	<code>&lt;wwpage:Import wwpage:replace-from-file="scripts/common.js" /&gt;</code>
wwpage:content-from-file	<code>&lt;div wwpage:Import wwpage:content-from-file="scripts/common.js"&gt;&lt;/div&gt;</code>
wwpage:replace-from-file	<code>&lt;wwpage:Import wwpage:replace-from-file="css/webworks.css" /&gt;</code>
wwpage:content-from-lookup	<code>&lt;div wwpage:Import wwpage:content-from-lookup="catalog-css"&gt;&lt;/div&gt;</code>
wwpage:replace-from-lookup	<code>&lt;wwpage:Import wwpage:replace-from-lookup="catalog-css" /&gt;</code>

## Logical AND/OR/NOT in condition attributes

The `wwpage:condition` attribute supports both logical `AND` as well as `OR` to provide for multiple conditions in a single attribute expression. For a logical

`AND`, use white space as a separator between conditions. For a logical `OR`, use a single comma as a separator between conditions. The following shows a few simple examples of using multiple conditions:

Logic Type	Example
AND (use space)	<code>&lt;hr wwpage:condition="header-exists company-email-exists" /&gt;</code>
OR (use ",")	<code>&lt;hr wwpage:condition="header-exists,footer-exists" /&gt;</code>
NOT (use "!")	<code>&lt;hr wwpage:condition="!header-exists" /&gt;</code>
AND with OR and NOT	<code>&lt;hr wwpage:condition="header-exists company-email-exists,footer-exists !company-email-exists" /&gt;</code>

## Working with URL and File Paths in Page Templates

The `wwpage:attribute-<name>` attribute can be used to specify several behaviors when working with URL and File paths. The following shows the list of available modifiers for this `wwpage:attribute`.

Attribute Modifier	Example
relative-to-output	<code>&lt;script type="text/javascript" src="scripts/common.js" wwpage:attribute-src="relative-to-output"&gt;&lt;/script&gt;</code>
copy-relative-to-output	<code>&lt;script type="text/javascript" src="scripts/common.js" wwpage:attribute-src="copy-relative-to-output"&gt;&lt;/script&gt;</code>
relative-to-output-root	<code>&lt;script type="text/javascript" src="scripts/common.js" wwpage:attribute-src="relative-to-output-root"&gt;&lt;/script&gt;</code>
copy-relative-to-output-root	<code>&lt;script type="text/javascript" src="scripts/common.js" wwpage:attribute-src="copy-relative-to-output-root"&gt;&lt;/script&gt;</code>
absolute-to-output	<code>&lt;external-graphic content-height="scale-to-fit" width="1in" height="1in" src="url('{wwformat:Pages/images/Logo.png}')" wwpage:attribute-src="absolute-to-output" /&gt;</code>
resolved-uri	<code>&lt;external-graphic src="url('{wwformat:Pages/images/rms_doc_logo.png}')" wwpage:attribute-src="resolved-uri" content-width="100px" content-height="100px" left="0pt" top="0pt" /&gt;</code>
resolved-path	<code>&lt;external-graphic src="{wwformat:Pages/images/rms_doc_logo.png}" wwpage:attribute-src="resolved-path" content-width="100px" content-height="100px" left="0pt" top="0pt" /&gt;</code>

## Using Replacement Types ({}) in Page Templates

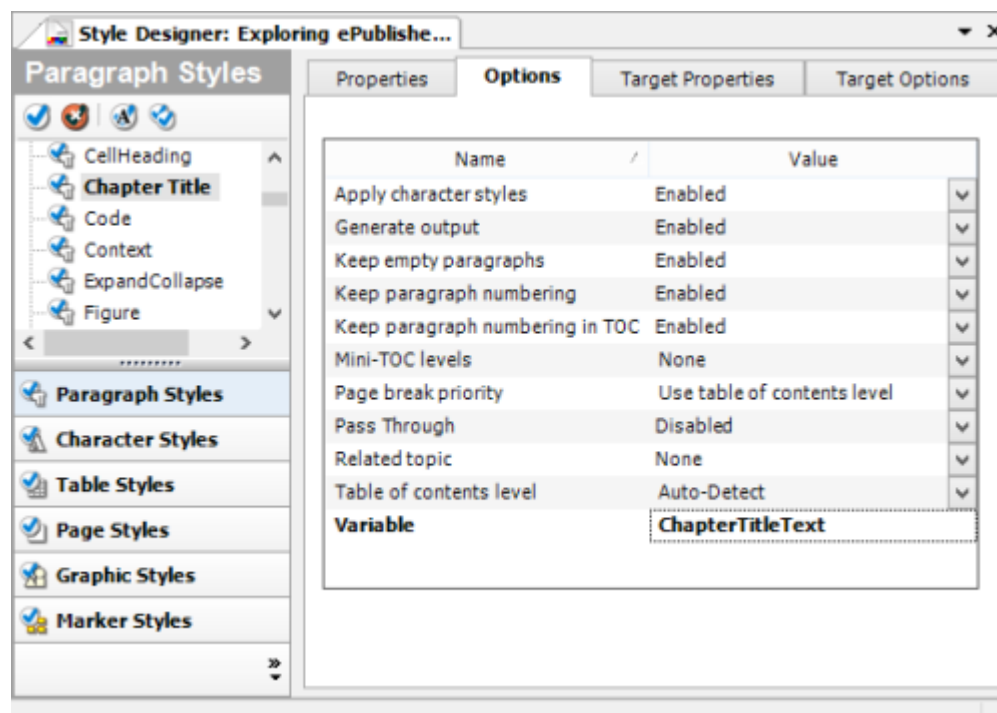
In page templates, you can use the "{}" characters to indicate a parameter to use in an attribute replacement. The "{}" characters are used to embed a path or parameter within an attribute value. For example, the following is a typical replacement for an HTML `img` tag, demonstrating how "{}" characters can be used to improve the result of the replacement.

```
.
```

## Using ePubublisher Style Variables in Page Templates

If you have meta data (text) in your source content, you can use ePubublisher Styles to capture that text and assign it to a named variable that can be accessed using the `wwvars:<Variable_Name>` in your page template files such as `Page.asp` or `Body.asp`.

To capture text from your source documents, you first need to configure the **Variable** style option within your ePubublisher Designer project. The **Variable** style option is available for Paragraph, Character, and Marker styles.



In the ePubublisher Style Designer, select the Variable option and assign it a value that will be used in your page template.

Once you have assigned a variable name to the style, any time ePubublisher encounters that style it will set the value of that variable to the text of that style in the source content.

To access the variable in your page template, you would use the `wwvars:<Variable_Name>` attribute, similar to the following.



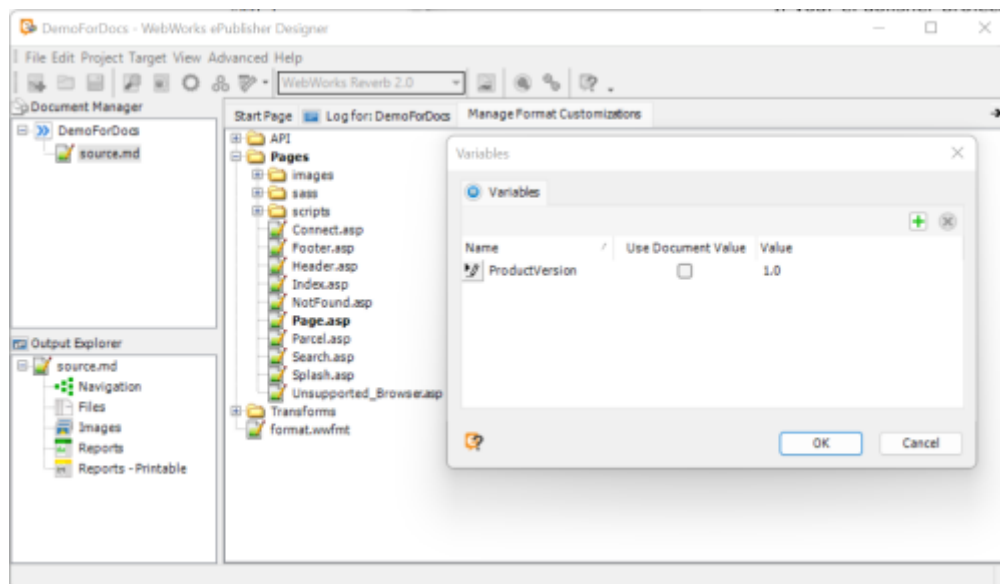
```
<div wwpage:content="wwvars:ChapterTitleText"
  wwpage:condition="wwvars:ChapterTitleText">
```

Chapter Title Appears Here

```
</div>
```

## Using ePublisher Project Variables in Page Templates

In ePublisher you can set up Project Variables that can later be modified using ePublisher Express or AutoMap at generation time. Project variables inject content into your source content during the generation of output. This feature is useful for maintaining commonly changing values such as your product's version number or copyright date or even the name of the manual.



These variables can then be used within your page template files (i.e. `Page.asp`, `Body.asp`). For example, if you have a Project Variable called: `ProductVersion`, you can access its value by using the `projvars:<Variable_Name>` in your page template file.

To access the project variable in your page template, you would use the `sprojvars:<Variable_Name>` attribute, similar to the following.

```
<div wwpage:content="projvars:ProductVersion"
  wwpage:condition="projvars:ProductVersion">
```

This text replaced with ProductVersion value.

```
</div>
```

# Using Markers in Page Templates

Another powerful feature for capturing text within your source content is through the use of markers. Within your authoring environment you can insert markers that have a name and value. Then in your page templates, you can access the value of these markers by their name. The page template will use the last encountered marker within the scope of the generated page of your output.

To implement this functionality in your page template file, use the syntax:

`wwmarker:<Marker_Name>.`

For example, if you used a marker called: Reviewer the following is syntax that could be used in the page template file to capture the value.

```
<div wwpage:condition="wwmarker:Reviewer"
  wwpage:content="wwmarker:Reviewer">
```

Reviewer name appears here, if marker is available.

```
</div>
```

# ePublisher Pipeline and Transforms

Terminology

Processing Workflow

Transformation Process

Stationery, Projects, and Overrides

ePublisher provides the complete framework needed to solve an unlimited number of publishing issues. ePublisher provides a flexible processing model that allows you and third-party developers to customize the ePublisher workflow as needed. This workflow uses the open, standards-based XSL transformation approach to give you the flexibility and extensibility you need without locking your data in a proprietary format.

## Terminology

To understand how ePublisher works, you should first understand some terminology. The following list defines several important product terms:

### source document input formats

The types of source documents ePublisher processes, such as Markdown++, Adobe FrameMaker, Microsoft Word, and DITA-compliant XML files.

### output formats

The types of output ePublisher can generate from your source documents. Each output format is a set of individual XSL transforms that process source documents to create the output. Quadralay provides many default output formats, such as WebWorks Reverb (1 & 2), WebWorks Help, HTML Help, Eclipse Help, Oracle Help, Sun JavaHelp, XML+XSL, Dynamic HTML, and many others. You can also create custom output formats. An output format is broken down into pipelines and stages.

### format files

The files that define the output format and transforms. These files are stored in the `Formats` folder in the installation folder. You can override files to customize the transformation process and these override files are stored in a matching location within the project folder. When you save a Stationery, the override files are stored with the Stationery.

### Stationery

A complete set of processing rules and styles that define all aspects of the output. The Stationery can define multiple targets, and each target can be the same or different output format. Writers use the Stationery created by the Stationery designer when they create projects and generate output. Writers

can also override some default settings in their projects, such as variable values and conditions.

### **project**

Identifies the Stationery to use, the source documents to process, and the output format and target settings to override when generating output. A project can include multiple targets.

### **target**

Defines a deliverable for the project, based on an output format. A project can have multiple targets and each target can be the same or different output format. All targets in a project share the Style Designer properties and options, but each target has its own target settings, such as variable values and company information.

### **stage**

The smallest discrete action possible in an output format, such as an XSL transform. ePublisher transforms source documents to a target by breaking the process into a series of steps, also known as stages. A stage is part of the transformation process that performs a specific action in the process. Stages are grouped into pipelines of related stages. In the future, non-XSL actions may also be possible. All stages define an output file type and zero or more input file types.

### **pipeline**

A set of stages that are run in sequence. Pipelines can have dependencies on other pipelines, which can ensure that required input files are created before a pipeline runs.

## **Processing Workflow**

A **Stationery designer** creates Stationery that identifies the supported output formats and defines the appearance and behavior of the output generated with projects based on that Stationery. A **writer** creates a project based on the Stationery and identifies the source documents to include in the project. Then, the writer uses the project to generate output for each target defined in the project. A **developer** can create custom behaviors and formats for Stationery designers to enable and use in the ePublisher Designer console.

ePublisher uses stages, pipelines, and the `format.wwfmt` file for each output format to structure and manage the processing workflow. A stage in a pipeline runs only once. The stage itself must determine the number of files to process and the number of files to emit. A stage can pass any message to another stage as long as it is emitted to a file first, and then the type and path of the file are emitted in the XML result.

ePublisher processes files as follows:

1. Apply conditions, cross-reference formats, and variables to source documents.
2. Export source documents to WIF.
3. Determine the execution order for format pipelines.
4. Select the next pipeline available for processing and execute all the stages defined in that pipeline.

To fill gaps left by XSL, ePublisher provides several XSL extension objects to support specific actions, such as processing image files, modifying the file system, and writing multiple documents from a single XSL transform.

## Transformation Process

The first pipeline in the process prepares the source documents to be transformed by XSL. ePublisher uses XSL to transform documents to output formats. ePublisher also provides extensions to XSL to support image processing and other source document operations that otherwise would require additional technologies, such as FrameScript or VBA. ePublisher requires native access to source documents, a standard starting point to start XSL transformation, and a standard method for coordinating XSL transforms on files. This first pipeline applies conditions and variables, extracts native drawings and images, and exports the source documents to the WebWorks Intermediate Format (WIF), a WebWorks XML language that enables XSL to process the source documents in later stages.

When XSL processing begins, ePublisher processes files based on type rather than by name. Each stage defines an `.xsl` file that creates a portion of the output target. When every stage in every pipeline is finished, the transformation process is complete.

Using the transformation to WIF gives you the ultimate flexibility with multiple input and output format support. Each input format, such as Microsoft Word and Adobe FrameMaker, can be transformed to any and all output formats, such as Microsoft HTML Help and WebWorks Help. As new output formats become available, all input formats are automatically supported. In addition, a new input format automatically supports all output formats.

## Adapters Transform Source Documents to WIF

To extract content from source documents, ePublisher defines an adapter interface. This adapter interface allows ePublisher to transparently support a variety of source documents, such as Microsoft Word, Adobe FrameMaker, and DITA.

When processing content, ePublisher is not aware of the type of adapter in use. ePublisher simply asks for an adapter that can process a source document, and then it sends requests to the adapter associated with that document type.

Adapters handle the following tasks:

1. Reporting book files.
2. Scanning for styles, conditions, cross-references, and variables.
3. Applying specified conditions, cross-reference formats, and variables to source documents.
4. Extracting native drawings and images.
5. Exporting source documents to WIF.

This final step is where the source document is effectively brought into the world of XML/XSL processing.

## WebWorks Intermediate Format (WIF)

WIF is a Quadralay Corporation standard used as an intermediate format. Quadralay chose to establish WIF rather than use another XML format because other XML formats do not provide all the functionality needed to solve conversion-related issues. Other XML standards are great for authoring structured content, but they deliberately exclude formatting information and they are not designed to define source that lacks structure. To address many conversion-related issues, such as unstructured Microsoft Word source documents with one style (Normal) used throughout the documents and customized as needed, Quadralay needed a more flexible and powerful intermediate format.

The most flexible way to resolve these issues was to create WIF, which provides a stable XML schema that can grow and change as needed for future input and output format requirements. WIF is designed to address the following goals:

- Represent source documents in a standard schema for XSL processing.
- Preserve individual word processor features with high fidelity.
- Strongly favor XSL processing over ease of authoring.

## Processing Files by Type

Once source documents are transformed into WIF, XSL processing can begin. The next question is about how to organize the XSL processing. ePublisher gives you the flexibility you need to define a workflow without the drudgery. While some processes and standards require intimate knowledge of an XSL transform's input

and output *file names*, ePublisher uses knowledge of an XSL transform's input and output *file types*. This distinction is subtle but powerful.

Consider processing XSL with exact file names:

- `xslt doctoc.xml alpha.xml > alpha_toc.xml`
- `xslt doctoc.xml delta.xml > delta_toc.xml`
- `xslt grouptoc.xml alpha_toc.xml delta_toc.xml > group_toc.xml`

Now consider using file types in place of file names:

- `xslt doctoc.xml Source > Doc_TOC`
- `xslt grouptoc.xml Doc_TOC > Group_TOC`

Using file types in place of file names simplifies the specification, where the file types are defined as follows:

File Type	File Names
<code>Source</code>	<code>alpha.xml</code> <code>delta.xml</code>
<code>Doc_TOC</code>	<code>alpha_toc.xml</code> <code>delta_toc.xml</code>
<code>Group_TOC</code>	<code>group_toc.xml</code>

With the file type approach, you need to specify `xslt doctoc.xsl` `Source` > `Doc_TOC` only once. This specification runs for every `Source` type document found and generates the corresponding `Doc_TOC` output files.

## Identifying Files to Process

Processing files based on type rather than individual file names allows developers to define workflows for any number of input and output files. To identify the files each XSL transform should process or create, ePublisher defines an XML schema to store file information. A simplified form of this schema looks like the following example:

```
<Files>
  <File name="alpha.xml" type="Source" />
  <File name="delta.xml" type="Source" />
</Files>
```

This XML document is fed as input to each XSL transform, and every XSL transform emits the list of generated files using this same schema. Therefore, running `doctoc.xsl` with the previous example input file list returns the following output:

```
<Files>
  <File name="alpha_toc.xml" type="Doc_TOC" />
  <File name="delta_toc.xml" type="Doc_TOC" />
</Files>
```

In this example, the final transform, `grouptoc.xsl`, runs with the following input file list:

```
<Files>
  <File name="alpha.xml" type="Source" />
  <File name="delta.xml" type="Source" />
  <File name="alpha_toc.xml" type="Doc_TOC" />
  <File name="delta_toc.xml" type="Doc_TOC" />
```



```
</Files>
```

This final transform returns the following output:

```
<Files>
  <File name="group_toc.xml" type="Group_TOC" />
</Files>
```

The *Source* documents were fed into the `grouptoc.xsl` transform because all previous output files are available to all downstream XSL transforms. ePublisher allows developers to define as many XSL stages as needed, and each stage can process, reprocess, examine, and manage, any preceding file generated.

## TOC Processing Example

The previous sections describe a TOC processing example to illustrate how ePublisher processes files. The following example shows the corresponding `format.wwfmt` file for this TOC processing example:

```
<Format>
<Pipeline name="TOC">
  <Depends pipeline="Locale" />
  <Stage type="xsl" action="doc_toc.xsl">
    <Parameter name="ParameterDependsType" value="Source" />
    <Parameter name="ParameterType" value="Doc_TOC" />
  </Stage>
  <Stage type="xsl" action="group_toc.xsl">
    <Parameter name="ParameterDependsType" value="Doc_TOC" />
    <Parameter name="ParameterType" value="Group_TOC" />
  </Stage>
</Pipeline>
</Format>
```

This representation defines the XSL transforms and file types to process.

## Stationery, Projects, and Overrides

The following sections provide an overview of the ePublisher files used in the transform process. You can customize these files and store them in an override location to customize how ePublisher transforms your content. For more information about terms used in the following sections, see “Terminology”.

### File Locations

ePublisher Designer allows you to create Stationery that writers can base their projects on. The Stationery stores the style settings and customized files used to define how ePublisher transforms your content. When writers create projects based on Stationery, their projects copy the customized files and settings from the Stationery and use those customized files when processing the content included

in their projects. In this way, customized files in the projects override the default ePublisher files. The Stationery also allows you to quickly roll out changes to your customized processes and settings. Once you update your Stationery, writers are notified when they open a project based on the Stationery to synchronize with the Stationery, which incorporates your latest changes.

ePublisher uses file and folder locations to provide a structure where you can create and store override files. An **override file** is a customized file stored in a parallel location in a project or Stationery that is used to process the source documents instead of the default ePublisher file stored in the installation folders. These files customize how ePublisher transforms your content. ePublisher uses the following locations to store its files:

## Your project folders

By default, each project is saved in the `My Documents\componentname\Projects\projectname` folder, where *componentname* is the name of the ePublisher component used to create the project, such as `ePublisher Designer` or `ePublisher Express`, and *projectname* is the name of the project itself.

## ePublisher installation folder

By default, ePublisher components are installed in the `Program Files\WebWorks` folder. The default transformation files are stored with ePublisher Designer in the `Program Files\WebWorks\ePublisher\ePublisher Designer` folder.

When you generate output for an output format, ePublisher first checks the project folders for the files required to complete the task. If the files are not found in the project folders, ePublisher checks the installation folders. This process allows you to override any default file in the installation folder hierarchy for one or more output formats by placing a customized file with the same name at the correct location in the project folder hierarchy.

**Note:** Do not modify files in the installation folders. Store customized files only in the project folder hierarchy.

The files used to transform a source document are located in several main folders in the ePublisher installation folder hierarchy:

### Formats

Contains format-specific XSL files. The default files that you can customize and store in your project folder hierarchy are stored in this folder. The `Formats\Shared` folder contains XSL files used by multiple formats.

### Helpers

Contains command-line programs to perform specific actions, such as compiling HTML Help `.chm` files or generating WebWorks Help search indices.

## File Processing

ePublisher divides the transform process into a series of pipelines, or groups of similar stages. Each stage defines an `.xsl` file to run that creates a portion of the target. ePublisher uses a combination of the format pipelines, the `files.info` GlobalFiles record file, XSLT parameters, and the root match template in a given XSL file to perform the action identified in a stage. These stages are defined in the `format.wwfmt` file located in each format folder. The `format.wwfmt` file defines all the pipelines and stages necessary to create a specific output format, and the relationships that each of these pipelines and stages have to one another. There is no preconceived order in which pipelines run. ePublisher calculates at run time the order in which each pipeline runs, based on pipeline dependencies.

ePublisher processes files based on type rather than by name. Each stage defines an `.xsl` file to run that creates a portion of the target. All ePublisher XSL style sheets define the following global parameters by default:

Parameter	Description
GlobalFiles	Provides the <code>files.info</code> file for the project, which includes all previously generated files in the files XML schema.
GlobalProject	Provides the <code>.wep</code> project file, which defines the project as XML.
GlobalPipelineName	Specifies the name of the pipeline in which the current stage is defined.
GlobalInput	Identifies the files selected in Document Manager. This parameter provides all previously generated files derived from the <code>Generate Selected</code> command in the files XML schema.

In addition, ePublisher passes the XSL style sheets all parameters defined for the current stage in the `format.wwfmt` file. Parameters passed to each XSLT file are usually used to load the various XML node sets needed for the current stage. For example, a stage may need to take information recorded in the **Behaviors** pipeline and merge it with information in the **Links** pipeline. The location of information generated in each stage is stored in the `files.info` file. Future stages can use the `files.info` file to learn the location of previously generated information needed to complete that particular stage. When that stage finishes, it notifies the `files.info` file that the information it created is available for future stages to use, if necessary.

Processing files based on type rather than by individual names allows you to define workflows for any number of input and output files. This flexibility allows you to define as many XSL stages as you need, and each stage can use any preceding files generated. You can create or delete pipelines, add or delete stages, or insert elements that dictate when to run a stage or pipeline.

## ePublisher File Types

This section provides specific information about the files most commonly used to customize ePublisher and its transformation processes.

### Format Trait Info (\*.fti) Files

Format Trait Info files have the `.fti` file extension. Format Trait Info files are located in both the ePublisher `Formats\Shared` folders, and the `Format` folders of each target.

In every project, the user has a series of format and style options to customize the appearance of his output. In the ePublisher consoles, these options are accessed and manipulated through target settings and Style Designer. For ePublisher, these customization options are defined in the Format Trait Info files. Manipulating these files allows you to customize the location of options, create or delete options and parameters, adjust their valid values, or choose new default values.

For ePublisher to process a Format Trait Info file, the `.fti` file must have the same file name, not including the `.fti` file extension, as an XSL file in the current format folder hierarchy. For example, ePublisher processes the table of contents using the `toc.fti` and `toc.xsl` files in the same folder. The information displayed in ePublisher represents all Format Trait Info files associated with a specific format. Two or more Format Trait Info files may define the same `<Setting />` element, but the ePublisher console displays that setting only once.

When you generate output, ePublisher reviews the settings specified in target settings and Style Designer, stores the settings in the `.wep` project file, and incorporates the settings in the generated output.

## format.wwfmt Files

A `format.wwfmt` file is stored in each of the specific format folders. The `format.wwfmt` file defines all the stages required to create output, and the relationships between each of these stages. These stages are grouped in pipelines of related stages. When every stage of every pipeline has finished, ePublisher is done generating the output. There is no preconceived order in which pipelines run. ePublisher calculates at run time the order in which each pipeline runs. You can create or delete pipelines, add or delete stages, or insert a `<Depends />` element that dictates when a stage or pipeline runs.

## files.info Files

ePublisher creates and stores the `files.info` file for each target with the project files. Each stage completes a small portion of the transformation. A completed stage creates new information that contributes to the appearance or functionality of the transformed content. When a stage finishes, ePublisher writes the location of the processed information in the `files.info` file for use by other stages. Each stage can use information in the `files.info` file and record information in the `files.info` file for use by another stage.

## Stationery Design Project .wep File

The Stationery design project `.wep` file contains all the configuration information required to define the Stationery and generate output. This file includes the

sample source document names, settings, options, and customizations. You can use ePublisher Designer to modify a Stationery design project, and to save it as Stationery.

## Project .wrp File

The project `.wrp` file contains all the information required to generate output. This file includes all the source document names, settings, options, and customizations.

## Stationery .wxsp File

A Stationery `.wxsp` file is based on settings and options defined in a Stationery design project and saved as Stationery. By saving a project as Stationery, all the settings and customizations that you captured in your project, including project format overrides, are automatically implemented in any new projects based on the Stationery.

## XSL Match Templates

Style sheets are made of a number of templates, each of which defines what the XSLT processor should do when it matches a particular node in the XML source document. The XSLT processor populates the result document by instantiating a sequence of templates. Instantiation of a template means that the XSLT processor performs the following tasks:

- Copies any literal data from the template to the target
- Executes the XSLT instructions in the template

Templates are defined using the `<xsl: template>` element. The `match` attribute in the `<xsl:template>` element indicates which parts of the source document should be processed with the particular template.

## Root Match Templates

Each XML document has a single root element. This element encloses all the following elements and is therefore the parent element to all the other elements. When the XSLT processor applies a style sheet to an XML document, it begins processing with the root element of the XML source document. To process the root element, the XSLT processor searches the style sheet for a template rule that matches the root element. A template rule matches the root element when the value of the template `match` attribute is `/` (slash).

If the user explicitly defines a template rule that matches the root element, the XSLT processor finds it and applies that template to the entire XML document. If the XSLT processor does not find an explicitly defined template rule that matches the root element, the processor implements the default template that matches the root element. Every style sheet includes this default template.

# Root Match Templates in ePublisher

The root match template has a number of responsibilities in ePublisher:

- Recording files and dependencies
- Loading node sets
- Progress recording for the extension object that lives in the `wwprogress` namespace.
- Up-to-date checking on the projects output files.
- Writing output

## Extension Objects

While extremely powerful, XSL has shortcomings when used to perform system-level scripting tasks. Some operations, such as working with files and advanced string operations, are not included in the standard XSL language.

XSL provides a generic extension mechanism to add new features to the base language. ePublisher provides a standard set of extension objects that enable XSL to generate all the required formats.

XSL extensions live in a specific namespace. You can define your own prefix to associate with a given namespace, but using a consistent naming convention makes life easier.

## Creating Super Overrides

An XSL super override is simply a way for the stationery designer to add an XSL override to a file in the Transforms directory without having to copy the entire template structure of the file.

**Note:** Support XSL overrides is only available through Study Hall and not Standard Support. For more information regarding Study Hall, please go to [http://www.webworks.com/eschool/study\\_hall/](http://www.webworks.com/eschool/study_hall/).

Here is an example from the WebWorks wiki of `XSL` that you can use as a starting point for learning about this feature:

```
<!-- Override the mode wwdoc:Table from the base content.xsl -->
<!--
<xsl:template match="wwdoc:Table" mode="wwmode:content">
  <xsl:param name="ParamTable" select="." />
  <!-- Parameters passed in to this processing context. We need to
  pass them along. -->
```

```

<!--
-->
<xsl:param name="ParamSplits" />
<xsl:param name="ParamCargo" />
<xsl:param name="ParamLinks" />
<xsl:param name="ParamTOCData" />
<xsl:param name="ParamSplit" />

<!-- Manually set all table formatting in this template. -->
<!--
-->
<html:table cellpadding="0" cellspacing="5" border="1"
style="margin-top: 0.6em; margin-bottom: 0.6em;">
  <xsl:for-each select="$ParamTable/wwdoc:Caption/wwdoc:Paragraph[1]
| $ParamTable/preceding-sibling::wwdoc:Paragraph[@stylename =
'Caption'] [1]">
    <html:caption>
      <html:p>
        <html:b>
          <xsl:for-each select="./wwdoc:TextRun/wwdoc:Text">
            <xsl:value-of select="@value" />
          </xsl:for-each>
        </html:b>
      </html:p>
    </html:caption>
  </xsl:for-each>

  <xsl:for-each select="$ParamTable/wwdoc:TableHead | $ParamTable/
wwdoc:TableBody | $ParamTable/wwdoc:TableFoot">
    <xsl:variable name="VarSection" select="." />

    <xsl:variable name="VarTagName">
      <xsl:choose>
        <xsl:when test="local-name($VarSection) = 'TableHead'">
          <xsl:text>thead</xsl:text>
        </xsl:when>
        <xsl:when test="local-name($VarSection) = 'TableBody'">
          <xsl:text>tbody</xsl:text>
        </xsl:when>
        <xsl:when test="local-name($VarSection) = 'TableFoot'">
          <xsl:text>tfoot</xsl:text>
        </xsl:when>
      </xsl:choose>
    </xsl:variable>

    <xsl:element name="{ $VarTagName}" namespace="'http://
www.w3.org/1999/xhtml'">
      <xsl:for-each select="$VarSection/wwdoc:TableRow">
        <xsl:variable name="VarRow" select="." />

```



```

<html:tr>
  <xsl:for-each select="$VarRow/wwdoc:TableCell">
    <xsl:variable name="VarCell" select="." />

    <html:td>
      <!-- Pass control back to the base processing flow. -->
      <!-- -->
      <xsl:apply-templates select="$VarCell/wwdoc:Paragraph"
mode="wwmode:content">
        <xsl:with-param name="ParamSplits" select="$ParamSplits" />
        <xsl:with-param name="ParamCargo" select="$ParamCargo" />
        <xsl:with-param name="ParamLinks" select="$ParamLinks" />
        <xsl:with-param name="ParamTOCData" select="$ParamTOCData" /
>
        <xsl:with-param name="ParamSplit" select="$ParamSplit" />
      </xsl:apply-templates>
    </html:td>
  </xsl:for-each>
</html:tr>
</xsl:for-each>
</xsl:element>
</xsl:for-each>
</html:table>
</xsl:template>

```

## To use the super override

1. In your Stationery design project, on the **Advanced** menu, click **Manage Format Customizations** (or Target for target overrides)
2. In the displayed collection of folders and files navigate to the **Transforms** directory, and click on the XSL file you wish to modify
3. Right click the XSL file, and click Create Super Customization
4. In the text editor, you will see the XSL that indicates:

```
<!-- Write templates here -->  
<!-- -->
```

5. Add the custom XSL, please note that the code will have to reference the original XSL file in order for ePubublisher to process it correctly. For more information, please refer to [this wiki page](#).
6. Save XSL override, save ePubublisher project, reopen and generate output for ePubublisher to process this override

# Introduction

[Audience](#)  
[Help](#)  
[Conventions](#)  
[Organization](#)  
[About XML and XSL](#)

Welcome to the Quadralay's Developer's Guide for WebWorks ePublisher Designer. This document serves as an overview of how ePublisher Designer works and provides details on how the user can make changes to files to customize both the process of converting a document, and the document's final appearance.

## Audience

This document is aimed at users of ePublisher Designer who want to make specific customizations to their projects. Users should have at least basic knowledge of XML, XSLT, and how the two interact. This guide presents a brief overview of XML and XSL. If the concepts there present confusion to the user, the references section provides resources for learning critical features of the languages.

## Help

User changes to ePublisher Designer files are not supported through Quadralay's Product Support department. Modification of files is only supported through Quadralay's Services department. Modifying files in the installation directories is not recommended. Files should be modified in a project itself or in user-created formats only.

## Conventions

This Reference Guide uses the following conventions.

## Formatting

### **Bold:**

File names are listed in bold

### **Code:**

```
<Classes>

<Class name="boolean">

  <Item value="true" stringid="boolean-true" />

  <Item value="false" stringid="boolean-false" />

</Class>

<Class name="color-name">

  <Item value="transparent" />

  <Item value="aqua" />

  <Item value="black" />

  <Item value="white" />

  <Item value="yellow" />

</Class>

</Classes>
```

Highlights markup text used in XML.

## Terminology

GUI:

A graphical user interface is a method for issuing commands to a computer through direct manipulation of graphical images and widgets in addition to text. The buttons and menus used in Microsoft Word are an example of a GUI.

Output Target:

The chosen format to which a source document will be transformed. There may be more than one output target.

Source Document:

The original document, created in Word or Framemaker, that will be transformed by ePublisher Designer to an output target.

Transform:

The process of converting a source document into a new chosen format.

## Organization

This book is separated into seven parts:

Introduction

Architecture Overview:

Details regarding how ePublisher Designer works

XSL Match Templates:

ePublisher Designer's unique approach to template rule matching

Extension Objects:

Details on extending the functionality of XSL with Microsoft and ePublisher Designer extension objects

File Reference:

Details on files the user may modify

Project Format Overrides:

How to get started

Appendix:

Extension Objects: General, Microsoft, and ePublisher Designer extensions

## About XML and XSL

XML is a meta-language. That is, it is a language used to create other markup languages. XML provides a basic structure and a set of rules to which any markup language must adhere. XML is based on three basic building blocks:

- elements
- attributes
- values

Elements describe or contain a piece of information and form the basis of all XML documents. Elements take the form of tags, as in HTML. Attributes are pieces of descriptive information that appear within an elements opening tag. An attribute

consists of an attribute name and a corresponding value, separated by an equal symbol (=). The values of an attribute appear to the right of the equal symbol and must appear within quotes.

Together, a group of elements, attributes, and values make up an XML document.

XML allows users to create elements, attributes and values. There are no fixed elements as in HTML. In HTML `<table>` means only information grouped together in rows and columns. In XML, an element with the name `<table>` could refer to an HTML type table, or a piece of furniture, or whatever the author would like. In order to bring order to these seemingly randomly-named elements, XML needs a document that explains what each of these building blocks mean. The document that defines these building blocks is called an XSL style sheet. A rough comparison can be made to Custom Style Sheets, used in HTML.

ePublisher uses two different aspects of XSL to generate an output:

XSLT:

XSLT describes how to transform a source document from one markup language to another.

XPath:

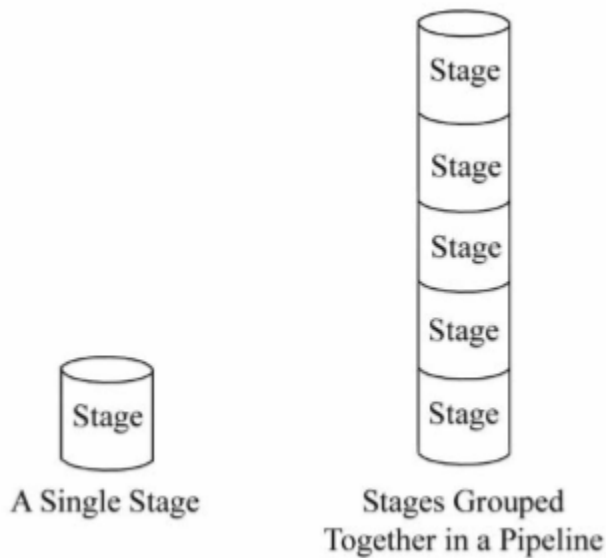
XPath is used by XSLT to select parts of XML to process and perform calculations.

An XSLT processor executes a stylesheet to give the user a particular result. In the transformation process, XPath defines parts of the source document for XSLT to match against one or more predefined templates. When a match is found, XSLT will transform the matching part of the source document into the result document.

## Architectural Overview

### Real World Example

ePublisher Designer converts a source document to an output target by breaking the process into a series of steps, or “stages.” Each stage performs a specific action in the process. These steps are grouped in “pipelines” of related stages. See Figure 1 for an illustration.



The first pipeline in the process prepares the source document to be converted by XSL. ePublisher Designer uses XSL to transform documents to target formats. XSL, however, cannot extract XML from Word or Framemaker documents, nor can it render images. This first stage applies conditions and variables, extracts native drawings and images, and exports the source document to WIF (WebWorks Intermediate Format), a WebWorks proprietary XML language that enables XSL to process the source document in later stages.

Once source documents are transformed into WIF, XSL processing can begin. ePublisher Designer processes files based on type rather than by name. Each stage defines an .xsl file to be executed which creates a portion of the output target. When every stage in every pipeline has been executed, the transform is complete.

## Real World Example

For the sake of demonstrating the ePublisher Designer transform process let us assume a fictional pizzeria in New York City. Further, we'll say that a bag of pizza ingredients in the refrigerator is our source document. While that bag of ingredients is perfectly nice, it isn't particularly useful to anyone. The chef needs to transform the ingredients into something useful to him, a cooked pizza. In ePublisher Designer, the ingredients are our source document, the pizza oven is XSL, and the cooked pizza is our output target.

Unfortunately, it isn't possible simply to throw the ingredients into the oven and then remove a pizza ten minutes later. The bag of ingredients must first be prepared so the oven can deal with the ingredients in a way that is useful to us. In ePublisher Designer, the process of rolling out the dough, and spreading the sauce and cheese is the first stage where a source document is prepared for XSL (the oven) to do it's job.

Experienced chefs understand that the cooking process is a series of chemical reactions of food to heat. The dough becomes crisp, the cheese melts, etc. The process creates what any reasonable person would define as a pizza. In ePublisher Designer, XSL (the oven) is applying a series of steps (cooking) that are changing our source document (uncooked pizza) into the output target (something the customer is willing to consume).

Just as the pizza chef can customize the pizza ingredients any number of ways, such as cooking the pizza longer or at a different temperature to get different results, the ePublisher Designer user may customize the XSL process to affect his source document.

## File Reference

File Locations  
File Processing  
What This Means For The User

This section provides basic information on the ePublisher Designer files used in the transform process. This is where the user can make changes to the process to customize his output. In our real life example, this is where the chef can make his crust extra crisp, or add pepperoni or anchovies.

## File Locations

There are two locations to keep in mind when planning to modify a file:

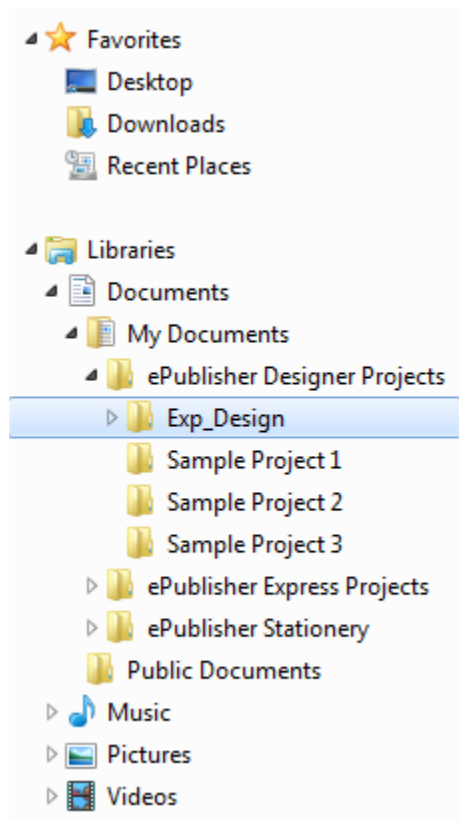
ePublisher Designer installation directory:

By default, ePublisher Designer is installed in C:\Program Files\WebWorks\ ePublisher Designer.

Project directory:

By default, a project is saved in "My Documents > ePublisher Designer Projects" in a directory with the same name as the project itself. See Figure 2.





When output is generated for a given output format, the ePublisher Designer engine first looks in the Project directory for the files required to complete its task. If the necessary files are not found in the Project directory, the engine will look in the installation directory. This means that any file contained in the installation directory hierarchy may be overridden in a given format by placing a file with the same name at the correct location in the Project directory. See the procedure detailed in “Creating Project Format Overrides” for more information.

Note: Modifying files in the installation directories is not recommended. Files should be modified in the project itself only.

The files used to transform a source document are located in three main folders in the ePublisher Designer installation directory:

Transforms:

This folder contains XSL documents which are used by all formats

Formats:

This folder contains format-specific XSL documents. Files to be modified by the user are located here.

Helpers:

This folder contains command line programs to perform some action (i.e., compiling .CHMs or generating WWHelp search indices)

## Real World Example

Let us revisit our pizzeria in New York City. For the sake of this example, let's pretend that once the chef makes a pizza, he completely forgets the process of how to do it. Luckily, the owner has placed the directions on the wall where the chef rolls out the dough. The chef quickly learns to check the wall to see what he is supposed to do next.

Now, sometimes, the customer wants thin crust. Or extra cheese. The instructions on the wall only tell the chef how to make one kind of crust, and only a certain amount of cheese. In order to make sure the chef makes the pizza the way the customer wants it, after receiving the order, the owner places instructions on the bag of ingredients the chef retrieves from the refrigerator. The owner tells the chef to always check the bag for instructions first. If there are no instructions on the bag, the chef should simply follow the directions posted on the wall.

In ePublisher Designer, the Project directory is the instructions posted on each bag. The pizza chef knows to check the bag (the project folder) first for instructions, then to check the instructions on the wall (the installation directory) if there are no instructions on the bag.

## File Processing

As mentioned previously, ePublisher Designer breaks up the transform process into a series of pipelines, or groups of similar stages. Each stage defines an **.xsl** file to be executed which creates a portion of the output target. ePublisher Designer uses a combination of the Format's Pipelines, the GlobalFiles record (**files.info**), XSLT parameters and the root match template in a given XSL file to perform the action to be executed in a stage. These stages are defined in the **format.wwfmt** file located in each Format directory. The **format.wwfmt** file defines all of the pipelines and steps necessary to create a given output, and the relationships that each of these have to one another. There is no preconceived order in which pipelines fire; the ePublisher Designer engine calculates at run time the order in which each pipeline will fire, based on pipeline dependencies.

ePublisher Designer processes files based on type rather than by name. Each stage defines an **.xsl** file to be executed which creates a portion of the output target. All ePublisher Designer XSL stylesheets define four parameters by default. These are as follows:

GlobalFiles –

A project's **files.info**

GlobalProject –

The project file (**\*.wep**)

GlobalPipelineName -

The pipeline of the current stage.

GlobalInput -

The files selected in the Document Manager.

In addition, XSL stylesheets are passed all parameters defined for the current stage in **format.wwfmt**. Parameters passed to each XSLT file are usually used to load the various XML node sets needed for the current stage. For example, a stage may need to take information recorded in the Behaviors pipeline and merge it with information in the Links pipeline. The location of information generated in each stage is stored in the **files.info** file. Future stages may consult **files.info** to learn the location of previously generated information needed to complete that particular stage. Similarly, when that stage is complete, it will send notice to the **files.info** file that the information it has created is available for future stages to use if necessary.

## What This Means For The User

Processing files based on type rather than by individual names allows developers to define workflows for any number of input-output files. This allows users to define as many XSL stages as they like and each stage may make use of any preceding file generated. Users may create or delete pipelines, add or delete stages, or insert elements that will dictate when a stage or pipeline is executed.

## File Types

Format Trait Info (\*.fti)  
format.wwfmt  
files.info  
Designer Project File (.wep)  
Stationery File (.wsxp)  
Express Project File (.wep)

The section provides specific information about the files most commonly used in customization of ePublisher Designer.

## Format Trait Info (\*.fti)

Format Trait Info files end in the .fti extension. Format Trait Info files are located in both the ePublisher Designer Transforms directory, and the format directories of each output target.

## Explanation

In every project, the user has a series of format and style options to customize the appearance of his output. In the GUI, these options can be accessed and manipulated through Target Settings and Style Designer on the ePublisher Designer toolbar. For the ePublisher Designer engine, these customization options reside in the Format Trait Info (\*.fti) files. Manipulating these files allows for customizing the location of options, creating or deleting options and parameters, or adjusting their values, or choosing a new default value.

Format Trait Info files must have the same base file name (that is, the file name without the file extension) as an XSL file in the current format in order to be processed by ePublisher Designer. The information displayed in the ePublisher GUI represents all Format Trait Info files associated with a given format. So, two or more Format Trait Info files may define a `<Setting />` but would only be displayed once in the GUI.

When ePublisher Designer is generating output, the settings specified in Target Settings and Style Designer are consulted, stored in the project file (.wep) and incorporated in the final generation of output.

## Components

There are four elements in an Format Trait Info (.fti) file.

### <Classes>

The only child element in `<Classes>` is `<Class>`. `<Class>` has a child element of `<Item>`. See Example 1 for sample code.

To facilitate organization, values for a given option may be grouped together as `<Items>` in a `<Class>`. The `<Class>` element can then be applied in the `<Settings>` and `<RuleTraitsSet>` elements to declare the options available for that specific option.

```

<Classes>

<Class name="boolean">

  <Item value="true" stringid="boolean-true" />

  <Item value="false" stringid="boolean-false" />

</Class>

<Class name="color-name">

  <Item value="transparent" />

  <Item value="aqua" />

  <Item value="black" />

  <Item value="white" />

  <Item value="yellow" />

</Class>

</Classes>

```

**Example 1: Sample Code Illustrating Child Elements,  
Attributes, and Values in the `<Classes>` Element**

## **<Groups>**

The only child element to `<Groups>` is `<Group>`. The `<Groups>` element defines items that can be used to group properties available in the Style Designer. Users can create groups and apply the group element tag in the appropriate location in the `<Settings>` and `<RuleTraitsSet>` elements of the Format Trait Info (**.fti**) file. This will assemble all `<Settings>` with a similar group together in the appropriate section of the GUI.

## **<Settings>**

The only child element of the `<Settings>` element is `<Setting>`. The `<Setting>` element defines the parameters for the GUI's Target Settings dialog. A user may customize the target settings by making changes in this portion of the Format Trait Info (**.fti**) file.

## **<RuleTraitsSet>**

The only child element of the `<RuleTraitsSet>` element is `<Options>`. `<Options>` contains the child element `<option>`. See Example 2 for sample code.

The `<option>` element defines the parameters for the GUI's Properties and Options Tabs in the Style Designer. A user may customize the Style Designer by making changes in this portion of the Format Trait Info (**.fti**) file.

```
<RuleTraitsSet>

  <RuleTraits category="Graphic">

    <Options>

      <Option name="file-extension" group="options" default=".jpg">

        <OptionClass name="file-extension" />

      </Option>

      <Option name="format" group="options" default="jpeg">

        <OptionClass name="image-format" />

      </Option>

      <Option name="color-depth" group="options" default="24">

        <OptionClass name="color-depth" />

      </Option>

    </Options>

  </RuleTraits>

</RuleTraitsSet>
```

**Example 2: Sample Code Illustrating Child Elements, Attributes, and Values in the `<RuleTraitsSet>` Element**

## Relationships

Format Trait Info appears in the ePublisher Designer user interface. When a user effects changes to these setting through the user interface, those changes are written to the project file (**\*.wep**). A format's XSL transformations then have access to these values by reading the project file.

# format.wwfmt

**format.wwfmt** files are located in the specific format directories.

## Explanation

The **format.wwfmt** file defines all of the steps necessary to create output, and the relationships that each of these steps has to one another. These steps, called “stages” are grouped in “pipelines” of related stages. When every stage of every pipeline has been executed, the ePublisher Designer engine output is complete. There is no preconceived order in which pipelines fire; the ePublisher Designer engine calculates at run time the order in which each pipeline will fire. Users may create or delete pipelines, add or delete stages, or insert a `<Depends />` element that will dictate when a stage or pipeline is executed.

## Components

The root element of **format.wwfmt** is `<Format />`. The child elements are `<Pipelines>` and `<Capabilities>`. See example 3 for sample code.

### **`<Pipelines>`**

The `<Pipelines>` element is a container element for the `<Pipeline>` elements that define the steps needed to generate the format. The only child element of the `<Pipelines>` element is `<Pipeline>`.

`<Pipeline>`:

The `<Pipeline>` element is a container element for the `<Depends>` and `<Stage>` elements which comprise a given segment of output files needed to generate a format. This element requires a name attribute so that it can be identified by `<Depends>` elements within other `<Pipeline>` elements. The `<Pipeline>` element contains the following child elements:

`<Depends>`:

This element specifies which other `<Pipeline>` elements the current `<Pipeline>` requires to complete its task. Including a `<Depends>` element in a pipeline is the only way to ensure a Pipeline does not execute before another Pipeline on which it depends. The ePublisher Designer engine calculates at run time the order in which each Pipeline will run.

`<Stage>`:

This element identifies an action to perform and specifies a configuration for the action. The action is identified via the type and action attributes, usually an XSL stylesheet, and the configuration is defined with `<Parameter>` elements. These

`<parameter>` elements define what is to be worked on, what will be created, and what else should be done with the result.

Note:

With the exception of Global Files and GlobalProject, all parameters passed to XSL transforms are strings. Global Files and Global Project are node-sets which have already been loaded.

## **`<Capabilities>`**

The `<Capabilities>` element contains only the child element `<Capability>` which defines information regarding what types of technologies or features a format supports.



```

<Capabilities>

  <Capability name="merge-context" value="false" />

  ...

</Capabilities>

<Pipelines>

  <Pipeline name="CompanyInfo">

    <Stage type="xsl" action="wwtransform:common/companyinfo/
companyinfo.xsl">

      <!-- Pull in Company Info .fti file -->

      <!--          -->

    </Stage>

  </Pipeline>

  <Pipeline name="DocumentBehaviors">

    <Stage type="xsl" action="wwtransform:common/behaviors/document.xsl">

      <Parameter name="ParameterDropDowns" value="false" />

      <Parameter name="ParameterPopups" value="false" />

      ...

    </Stage>

    <Stage type="xsl" action="wwtransform:common/behaviors/pullup.xsl">

      <Parameter name="ParameterDropDowns" value="false" />

      <Parameter name="ParameterPopups" value="false" />

      ...

    </Stage>

  </Pipeline>

</Pipelines>

```

### Example 3: Sample Code Illustrating Child Elements, Attributes, and Values in the `format.wwfmt` file

## Relationships

The locations of files generated by the action elements in the stages of the **format.wwfmt** file are stored in **files.info**. The **format.wwfmt** draws the same information from **files.info** as needed.

## files.info

This file is created and stored with the project files.

## Explanation

Each stage completes a small portion of the conversion. A completed stage creates new information that will contribute to the appearance or functionality of the new transform. Upon completion of the stage, the ePublisher Designer engine will write the location of the processed information in the **files.info** file for use by other stages attempting to complete their pipeline. Each stage may draw information from **files.info** and deposit information for use by another stage.

## Components

The root element of **files.info** is `<Files />`. The child elements are `<File>` and `<Depends>`.

The `<File>` element holds the location of the data created by a stage.

The `<Depends>` element notes the location of data on which that file is dependent.

## Relationships

This file stores the dependencies and locations of files created by the stages defined in **format.wwfmt**. The ePublisher Designer engine will provide this list of locations in order to process other stages in other pipelines.

## Designer Project File (.wep)

The project file (**.wep**) file is a collection of all the necessary information required to create output. This includes all the source document, settings, options and customizations created by the user or through the GUI.

## Stationery File (.wsxp)

**.wsxp** is the extension used for stationery files. Stationery is a file based upon configurations that have been made to a previous ePublisher Designer project. By saving a project as stationery, all of the settings and customizations that you captured in your project, including project format overrides, are automatically implemented in any new projects based on the stationery.

## Express Project File (.wrp)

The project file (**.wrp**) file is a collection of all the necessary information required to create output and is ideally suited for repeated publishing tasks. Unlike the Designer Project (**.wep**), the Express project uses a Stationery to update (import) all of its settings, options, and other customizations.

## XSL Match Templates

### Root Match Templates

Stylesheets are made up of a number of templates, each of which defines what the XSLT processor should do when it matches a particular node in the XML source document. The XSLT processor populates the result document by instantiating a sequence of templates. Instantiation of a template means that the XSLT processor

- Copies any literal data from the template to the target
- Executes the XSLT instructions in the template

Templates are defined using the `<xsl:template>` element. The `match` attribute in the `<xsl:template>` element indicates which parts of the source document should be processed with the particular template.

## Root Match Templates

Each XML document has a single root element. This element encloses all following elements and is therefore the parent element to all the other elements.

When the XSLT processor applies a stylesheet to an XML document, it begins processing with the root element of the XML source document. To process the root element, the XSLT processor searches the stylesheet for a template rule that matches the root element. A template rule matches the root element when the value of the template's `match` attribute is `"/`.

If the user explicitly defined a template rule that matches the root element, the XSLT processor finds it and implements that template to the entire XML document. If the XSLT processor does not find an explicitly defined template rule that matches the root element, the processor implements the default template that matches the root element. Every stylesheet includes this default template.

# Root Match Templates in ePublisher Designer

The root match template has a number of responsibilities in ePublisher Designer. They are listed here:

- Recording Files and Dependencies
- Loading Node Sets
- Progress recording for the extension object that lives in the `wwprogress` namespace
- Up-to-Date checking on the projects output files
- Writing output

## Recording Files and Dependencies

All root match templates in ePublisher Designer XSL files begin and end with the `<wwfiles:Files></wwfiles:Files>` elements. Within each root match template, usually near the bottom are one or more `<wwfiles:Files>` elements which include one or more `<wwfiles:Depends />` elements. The attributes in these elements provide information about the file such as the path, and type. Upon completion of a stage, the ePublisher Designer engine will write the location of the processed information in the **files.info** file for use by other stages attempting to complete their pipeline. Each stage may draw information from **files.info** and deposit information for use by another stage.

## Loading Node Sets

Another responsibility of the ePublisher Designer XSL root match template is to load the node sets needed for the transformation. This is usually done by accessing one or more `<wwfiles:Files />` elements from `files.info`, and using the `path` attribute on each `<wwfiles:Files />` element together with the `document()` or `wwxsl doc:LoadXMLWithoutResolver` extension object, to load the node set for the transform. In most cases, the `<wwfiles:Files />` elements are selected using an XSL parameter from the **format.wwfmt** file, or Stage in the current Pipeline.

## wwprogress

The `wwprogress` extension object sends messages to the ePublisher Designer progress indicator. If a root match template is being applied over several documents in several groups, then the number of documents or groups to be processed is recorded using the `Start` method in the `wwprogress` namespace.

Notice that an operation registered with the `wwprogress` extension object has concluded is initiated by using the `End` method.

## Up-to-Date

The ePublisher Designer's XSL file's root match template also ensures whether files to be processed are up to date. This is accomplished by passing attributes accepted by the `wwfilesext:UpToDate` extension method.

## Writing Output

If a file is determined to be not up to date, then a XSL template is called, with the result then passed along with information regarding what type of output file is to be written, to the `wwexsldoc:Document()` method for processing. If the output is to contain XML or HTML elements, then it is necessary to use the `msxsl:node-set()` method to make the contents of the variable behave as a new XML fragment.

## Real Life Example

Let's return to our pizzeria. Our pizza chef, who forgets how to make a pizza whenever he sets out to do so, has a set of rules he follows posted on the wall. Sometimes, the owner pastes special instructions on the bag of ingredients. But there are some rules that are never altered. The chef may be instructed every time to wash his hands. Or not to smoke a cigarette while making a pizza. These rules that he must make sure to follow every single time during the entire pizza production process, are the equivalent of the root match template.

## Extension Objects

While extremely powerful, XSL has many shortcomings when used to perform system level scripting tasks. Operations such as working with files and advanced string operations are not included in the standard XSL language.

XSL does provide a generic extension mechanism to add new features to the base language. ePublisher Designer provides a standard set of extension objects which enable XSL to generate all required formats.

XSL extensions live in a specific namespace. Users can define their own prefix to associate with a given namespace, but in general sticking with a consistent naming convention makes life easier.

## Output Customizations

Presenting information in print form can involve very different presentation decisions than displaying information in a web-friendly form. Display can vary from one platform to another, or one format may have features unavailable in another. In a transform performed by ePublisher Designer, the user can make several customization choices through various GUI options. For example, Target Settings may allow a user to specify whether a certain piece of information is displayed,

or how or where it is located on a page. The Style Designer allows the user to customize any style in the document to appear a certain way. By creating specific rules and conditions for how the document should appear when the transform is complete the user can add functionality or information for specific audiences, or make use of specific features of a given format.

By using XSL, ePublisher Designer allows the user to customize the process even further. The user may make additions, deletions, or modifications to the XSL files in ePublisher Designer to further customize the output. The user may even add, remove, or modify options available in Target Settings and Style Designer.

## Transform Overrides

### Creating Transform Overrides

Users make decisions on a Project's final output through the ePublisher Designer GUI. Some project modifications, however, cannot be made through the GUI. In these instances, the user may access the XSL files used by the ePublisher Designer engine, make changes, and thereby generate the desired output. Users may even add, modify or delete options in the GUI via Format Trait Info (**.fti**) overrides.

When output is generated for the first time, ePublisher Designer consults the **format.wwfmt** file in the format directory that tracks which actions are to be executed to generate the desired format. If there are no user customizations, all the files consulted by **format.wwfmt** are in the Format or Applications Transforms directories. By default, WebWorks ePublisher Designer will check a project's local directory for files before seeking the files from the installation directory. Users can override files in the default format files by creating a mirror directory structure within a given Project's Format or Target Override directory. By placing a file of the same name at the correct location in the Project Format/Target Override directory, any file in a given format within the installation directory may be overridden.

## Creating Transform Overrides

This procedure details the steps necessary to override a specific file and ensure that all previous functionality remains intact.

Note: Changes to **.xsl** and Format Trait Info (**.fti**) files are not supported through Quadralay's Product Support department. Assistance with modifying these files may be purchased through WebWorks Services.

## Information about Overriding files

Modifying files in the installation directories is not recommended. Files should be modified in the project itself only. See the following procedure for the steps required to do this.

The file name nomenclature used in ePublisher Designer is case sensitive. Care should be taken to ensure directories and files created by the user match the original installation directory or file name exactly.

It is not necessary to copy any files into the project Formats directory that you are not explicitly overriding.

## Procedure

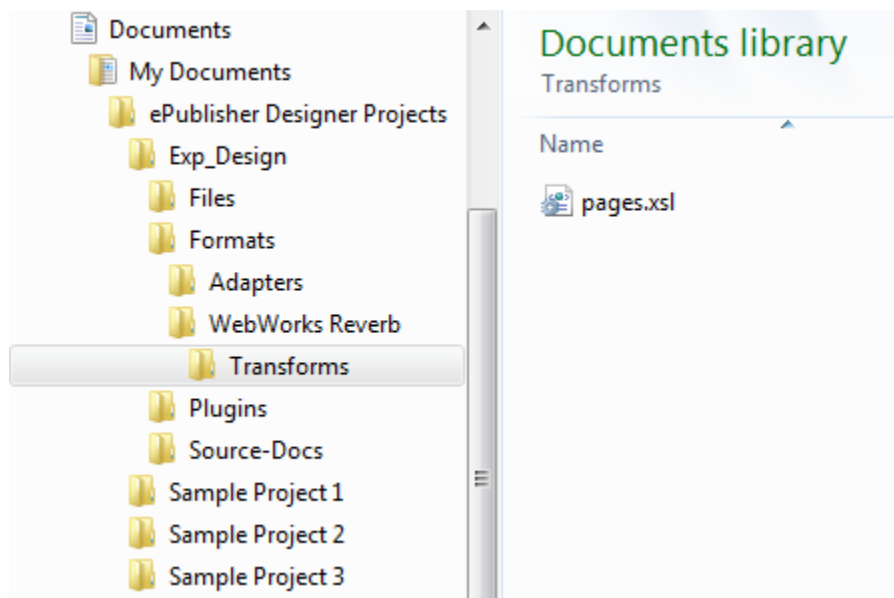
These steps may be taken with any of the files you need to modify inside the Formats directory. This procedure assumes that ePublisher Designer has been installed in the default directory, and that your ePublisher Designer projects are located in the "Documents" folder. If the user has installed ePublisher Designer somewhere else, or stores projects somewhere else, remember to substitute those locations when following the procedure.

Note: This is the manual way of doing an override, but you can always use the ePublisher Designer Advanced Menu (specially if you are overriding the Shared folder).

1. In Windows, open your project folder. Create a new folder named "Formats".
2. Open this new Formats directory.
3. Duplicate the folder hierarchy of the file you wish to modify.

**See Figure 2 for the folder hierarchy necessary to modify the pages.xsl file in WebWorks Reverb.**

Substitute the correct folder and file name inside the Formats directory.



**Figure 2: Folder Hierarchy Required To Modify pages.xml in WebWorks Reverb**

4. Navigate to the file you wish to modify in the ePublisher Designer installation directory.
5. Select and copy the file(s) you wish to override.
6. Navigate back to the Formats folder in the Projects directory created in steps 1-3.
7. Paste the file you copied from the installation directory into the folder created in step 3.
8. Modify files as needed.

After you have made the modifications you wish to make to the file in your project directory, save it, and the next time you click Generate in your ePublisher Designer project, this file will automatically override the default file and the changes you made will be incorporated into the output.

Note: When you save the ePublisher Designer project as Stationery, the project format overrides you have created will be saved with your Stationery. This stationery can then be used to create future projects.



# XSLT Reference

[XSLT Documentation](#)  
[Good to Know](#)  
[Using Extension Objects](#)

XSLT is the primary language used in transforming source XML into the various types of generated output. This section is an XSLT Reference for use with ePubliher.

## XSLT Documentation

Below is a collection of useful resources for better understanding XSLT:

### Useful Videos from WebWorks:

- Basics of XML, XSL, and ePubliher Conversions
- ePubliher Introduction to XSL and Extension Methods
- ePubliher XSLT, .Net, and Custom Formats
- ePubliher Document and WIF Processing
- Creating ePubliher Transform Overrides

### From the WebWorks Wiki:

- Debugging the XSL of an ePubliher Designer project
- XSL-FO Page template
- XML Transform Changes
- Locales
- Developer Documentation Topics

### Microsoft XSLT and XPath Reference

- XSLT Elements
- XSLT Syntax Patterns
- XSLT Functions
- XPath Handling and Special Characters
- XPath Node Collection Indexing, Finding, and Grouping

- XPath Logical Operators
- XPath Comparisons

## **Microsoft XPath Functions Reference**

- XPath Node-Set Functions
- XPath String Functions
- XPath Boolean Functions
- XPath Number Functions

## **XSLT Quick External Reference**

- Character Set used for HTML, XSLT, XML

## **Good to Know**

The below tools are useful for debugging and testing custom code in a controlled environment:

- For XSLT: XSLT fiddle
- For C#: .NET Fiddle
- For JavaScript/HTML/CSS: JSFiddle

Tool ePublisher uses to clean HTML documents: Tidy. If you need to customize Tidy, refer to the Tidy Quick Reference

## **Using Extension Objects**

The WebWorks ePublisher engine uses XML, XSL, and XPath as the foundation for all processing. While extremely powerful, XSL has many shortcomings when used to perform system level scripting tasks. Operations such as working with files and advanced string operations are not included in the standard XSL language.

XSL does provide a generic extension mechanism to add new features to the base language. WebWorks ePublisher provides a standard set of extension objects which enable XSL to generate all required formats.

## **General XSL Extensions**

XSL extensions live in a specific namespace. Users can define their own prefix to associate with a given namespace, but in general sticking with a consistent naming convention makes life easier.

For example, the XSL namespace is:

To define a prefix for it, one adds an XSL namespace declaration to your XSL stylesheet:

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
..
</xsl:stylesheet>
```

Now the desired namespace can be referenced just using a prefix rather than the actual namespace. Therefore, the following two lines are equivalent:

```
<xsl:variable name="VarNew" select="Hello" />

<variable xmlns="http://www.w3.org/1999/XSL/Transform" name="VarNew"
select="Hello" />
```

Microsoft's XSL transform implementation does not support extension elements at this time. Therefore, methods must be called from within "select" contexts.

```
<xsl:value-of select="wwexsl:Document($VarResult, $VarPath)" />

<xsl:variable name="VarDocumentWrite"
select="wwexsl:Document($VarResult, $VarPath)" />
```

## Microsoft Extensions

Extension objects that are defined and implemented by Microsoft as part of the .NET XSL transform runtime.

### Purpose

Microsoft implemented additional methods not part of the XSLT 1.1 standard. This work was done prior to the XSLT 2.0 and EXSLT specifications being finalized.

### Namespace

urn:schemas-microsoft-com:xslt

### Prefix

msxsl

Method

**msxsl:node-set(string textualXML):**

Description

Converts textual XML into a node set in a new XML document. Most often used to convert intermediate XML back into a working node set for additional processing.

Returns

A node set equivalent to the provided textual XML.

Note: The Microsoft XSL transform engine also supports custom extensions defined with script blocks.

## Using ePublisher XSLT Extensions

ePublisher implements a variety of extension objects to enable full processing of desired output within the language of XSL. In order to use them in XSL transforms they must be declared.

Here is an example of how to define namespace in you XSL file:

```
<?xml version="1.0" encoding="UTF-8"?>

<xsl:stylesheet version="1.0" xmlns="urn:WebWorks-Images-Schema"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:wwlog="urn:WebWorks-XSLT-Extension-Log"
                exclude-result-prefixes="xsl wwlog"
>

</xsl:stylesheet>
```

# ePublisher Platform XSLT Extensions

[Class Documentation](#)  
[Adapter](#)  
[AdapterConfiguration](#)  
[DateTimeUtilities](#)  
[Environment](#)  
[Exec](#)  
[ExecPython](#)  
[Sass](#)  
[ExslDocument](#)  
[Files](#)  
[FileSystem](#)  
[Fonts](#)  
[Imaging](#)  
[Log](#)  
[MultiSearchReplaceExtension](#)  
[NodeSet](#)  
[Progress](#)  
[Project](#)  
[StageInfo](#)  
[StringUtilities](#)  
[Units](#)  
[URI](#)  
[ZipExtension](#)

ePublisher implements a variety of extension objects to enable full processing of desired output within the language of XSL.

Namespace	Prefix	Class Name	Description
<b>urn:WebWorks-XSLT-Extension-Adapter</b>	wwadapter	<b><u>Adapter</u></b>	XSLT extension functions for use in ePublisher Adapter stylesheets.
<b>urn:WebWorks-Adapter-Configuration-Extension</b>	wwadapterconf	<b><u>AdapterConfiguration</u></b>	Provides access to the adapter configuration
<b>urn:WebWorks-XSLT-Extension-DateTimeUtilities</b>	wwdatetime	<b><u>DateTime</u></b>	Allows XSL pages access to various datetime methods
<b>urn:WebWorks-XSLT-Extension-Environment</b>	wwenv	<b><u>Environment</u></b>	Enable XSL transform to query the current system environment for the location and state of programs and variables.
<b>urn:WebWorks-XSLT-Extension-Execute</b>	wwexec	<b><u>Exec</u></b>	Allows XSL stylesheets to execute external programs and process results. Provides the return code, stdout, and stderr results from the running process.
<b>urn:WebWorks-XSLT-Extension-ExecPython</b>	wwpython	<b><u>ExecPython</u></b>	Allows XSL stylesheets to execute python programs and process results.
<b>urn:WebWorks-XSLT-Extension-Sass</b>	wwsass	<b><u>Sass</u></b>	Allows XSL stylesheets to manage SASS files
<b>urn:WebWorks-XSLT-Extension-Document</b>	wwxsldoc	<b><u>ExslDocument</u></b>	Allow multiple output files from a single XSL transform. Also provides routines to quickly load XML files

Namespace	Prefix	Class Name	Description
			without invoking XML validators as well as utility methods to enable correct output formatting.
<b>urn:WebWorks-XSLT-Extension-Files</b>	wwfilesext	<b><u>Files</u></b>	Enables incremental build support.
<b>urn:WebWorks-XSLT-Extension-FileSystem</b>	wwfilesystem	<b><u>FileSystem</u></b>	Allow XSL transforms to query and manipulate files and directories. Also handles system path parsing and processing.
<b>urn:WebWorks-XSLT-Extension-Fonts</b>	wwfonts	<b><u>Fonts</u></b>	Answer questions about fonts that might affect format output.
<b>urn:WebWorks-Imaging-Info</b>	wwimageinfo	<b><u>Imaging</u></b>	Enable processing of images within XSL transforms. Returns information about a particular image file, including width and height, image format, bit-depth, path on system, etc.
<b>urn:WebWorks-XSLT-Extension-Log</b>	wwlog	<b><u>Log</u></b>	Enables XSL transforms to report messages, warnings, and errors to the generation log.
<b>urn:WebWorks-XSLT-Extension-MultiSearchReplace</b>	wwmultisere	<b><u>MultiSearchReplaceExt</u></b>	Replaces multiple strings in a single operation.
<b>urn:WebWorks-XSLT-Extension-NodeSet</b>	wwnodeset	<b><u>NodeSet</u></b>	Miscellaneous node set functions.

Namespace	Prefix	Class Name	Description
<b>urn:WebWorks-XSLT-Extension-Progress</b>	wwprogress	<b><u>Progress</u></b>	Reports progress during long lived XSL transforms.
<b>urn:WebWorks-XSLT-Extension-Project</b>	wwprojext	<b><u>Project</u></b>	Query for information about the currently running project.
<b>urn:WebWorks-XSLT-Extension-StageInfo</b>	wwstageinfo	<b><u>StageInfo</u></b>	Allows XSLT processing to store and retrieve key/value pairs as needed to track state.
<b>urn:WebWorks-XSLT-Extension-StringUtilities</b>	wwstring	<b><u>StringUtilities</u></b>	Extend the available string processing methods to XSL to include message formatting, specialized text escaping, regular expression operations etc.
<b>urn:WebWorks-XSLT-Extension-Units</b>	wwunits	<b><u>Units</u></b>	Utility methods for extracting units and value from raw strings along with unit-to-unit conversion routines.
<b>urn:WebWorks-XSLT-Extension-URI</b>	wwuri	<b><u>URI</u></b>	Utility methods which convert to and from file paths and create absolute or relative URIs.
<b>urn:WebWorks-XSLT-Extension-Zip</b>	wwzip	<b><u>ZipExtension</u></b>	Allow XSL transforms to handle zip archives.

## Class Documentation

### Adapter

urn:WebWorks-XSLT-Extension-Adapter



## Functions

### **void AddToPDFPageNumberOffset (int addToPageNumberOffset)**

Adds to the PDF page number offset.

### **bool GeneratePDF (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string pdfJobSettings, string pdfFilePath)**

Generates a PDF.

### **bool GeneratePDFWithSaveAs (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string pdfJobSettings, string pdfFilePath)**

Generates a PDF using FrameMaker save as.

### **bool GeneratePostScriptForImage (object input, string postScriptPath)**

Generates postscript for image.

### **long GeneratePostScriptForPDF (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string postScriptFilePath)**

Generates a postscript for PDF.

### **void SetPDFPageNumberOffset (int pageNumberOffset)**

Sets PDF page number offset.

### **bool TemporaryLicense (string toolAdapterName)**

Checks if user is running a temporary license for specified 'toolAdapterName'.

## **Detailed Description**

urn:WebWorks-XSLT-Extension-Adapter

XSL extension functions for use in ePublisher Adapter stylesheets.

### **void AddToPDFPageNumberOffset (int addToPageNumberOffset)**

Adds to the PDF page number offset.

**Parameters:**

addToPageNumberOffset	The add to page number offset.
-----------------------	--------------------------------

**bool GeneratePDF (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string pdfJobSettings, string pdfFilePath)**

Generates a PDF.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

originalDocumentPath	Full pathname of the original document file.
conversionPDFDocumentPath	Full pathname of the conversion PDF document file.
singleFile	True for single file.
tocStyleNodesIterator	The TOC style node set.
groupFileNodesIterator	The group file node set.
pdfJobSettings	The PDF job settings.
pdfFilePath	Full pathname of the PDF file.

### Returns:

True if it succeeds, false if it fails.

**bool GeneratePDFWithSaveAs (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string pdfJobSettings, string pdfFilePath)**

Generates a PDF using FrameMaker save as.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

originalDocumentPath	Full pathname of the original document file.
conversionPDFDocumentPath	Full pathname of the conversion PDF document file.
singleFile	True for single file.
tocStyleNodesIterator	The TOC style node set.
groupFileNodesIterator	The group file node set.
pdfJobSettings	The PDF job settings.
pdfFilePath	Full pathname of the PDF file.

### Returns:

True if it succeeds, false if it fails.

### **bool GeneratePostScriptForImage (object input, string postScriptPath)**

Generates postscript for image.

### Parameters:

input	Node set containing content to be converted to postscript.
postScriptPath	Full pathname of the postscript file to create.

### Returns:

True if it succeeds, false if it fails.

**long GeneratePostScriptForPDF (string originalDocumentPath, string conversionPDFDocumentPath, bool singleFile, XPathNodeIterator tocStyleNodesIterator, XPathNodeIterator groupFileNodesIterator, string postScriptFilePath)**

Generates a postscript for PDF.

### Exceptions:



OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

originalDocumentPath	Full pathname of the original document file.
conversionPDFDocumentPath	Full pathname of the conversion PDF document file.
singleFile	True for single file.
tocStyleNodesIterator	The TOC style node set.
groupFileNodesIterator	The group file node set.
postScriptFilePath	Full pathname of the postscript file.

### Returns:

Number of pages in the PDF.

### void SetPDFPageNumberOffset (int pageNumberOffset)

Sets PDF page number offset.

### Parameters:

pageNumberOffset	The page number offset.
------------------	-------------------------

### **bool TemporaryLicense (string toolAdapterName)**

Checks if user is running a temporary license for specified 'toolAdapterName'.

#### **Parameters:**

toolAdapterName	Name of the tool adapter.
-----------------	---------------------------

### Returns:

True if is temporary, false otherwise.

## AdapterConfiguration

urn:WebWorks-Adapter-Configuration-Extension

### Functions

#### **string GetValue (string name)**

Gets a value.

#### **string GetValue (string name, string defaultValue)**

Gets a value and uses 'defaultValue' when no value set.

### Detailed Description

urn:WebWorks-Adapter-Configuration-Extension

XSL transform functions for getting information about the input configuration for adapters.

#### **string GetValue (string name)**

Gets a value.

### Parameters:

name	The input configuration item name.
------	------------------------------------

**Returns:**

The value.

**string GetValue (string name, string defaultValue)**

Gets a value and uses 'defaultValue' when no value set.

**Parameters:**

name	The input configuration item name.
defaultValue	The default value.

### Returns:

The value.

## DateTimeUtilities

urn:WebWorks-XSLT-Extension-DateTimeUtilities

### Functions

#### string **GetNow** (string format)

Takes a format string and gets DateTime in the given format.

#### string **GetGenerateStart** (string format)

Gets mStartDate and converts to given format, returns as string.

#### string **GetFileCreated** (string filePath, string format)

Gets created date from filepath and converts to given format.

#### string **GetFileLastModified** (string filePath, string Format)

Gets last modified date from filepath and converts to given format.

#### string **GetFromDateTimeString** (string dateTime, string inputFormat, string outputFormat)

Gets a date and a specific format, converts the date to another format.

### Detailed Description

urn:WebWorks-XSLT-Extension-DateTimeUtilities

Allows XSL pages access to various datetime methods.

#### string **GetNow** (string format)

Takes a format string and gets DateTime in the given format.

**Parameters:**

format	The datetime format.
--------	----------------------

**Returns:**

A datetime string.

**string GetGenerateStart (string format)**

Gets mStartDate at generation start and converts to given format.

**Parameters:**



format	The datetime format.
--------	----------------------

**Returns:**

A datetime string.

**string GetFileCreated (string filepath, string format)**

Gets created date from filepath and converts to given format.

**Parameters:**

filepath	Full pathname of the original document file.
format	The datetime format.

**Returns:**

A datetime string.

**string GetFileLastModified (string filepath, string format)**

Gets last modified date from filepath and converts to given format.

**Parameters:**

filepath	Full pathname of the original document file.
format	The datetime format input.

**Returns:**

A datetime string.

**string GetFromDateTimeString (string dateTime, string inputFormat, string outputFormat)**

Gets a date and specific format, converts the date to another format.

**Parameters:**

dateTime	The datetime input.
inputFormat	The datetime format input.
outputFormat.	The datetime format output.

### Returns:

A datetime string.

## Environment

urn:WebWorks-XSLT-Extension-Environment

### Functions

#### **string ApplicationBaseHelpURI ()**

Application base help URI.

#### **string CurrentUILocale ()**

Current user interface locale.

#### **long GetTotalMemory ()**

Reports the total amount of memory used by the running process. Useful for optimizing XSL to reduce memory usage.

#### **long GetTotalMemory (bool forceFullCollection)**

Reports the total amount of memory used by the running process. Useful for optimizing XSL to reduce memory usage.

#### **string HTMLHelpWorkshopPath ()**

HTML help workshop path.

#### **string JavaBits ()**

Gets latest version of either JDK or JRE bits stored in registry.

#### **string JavaHome ()**

Gets latest version of either JDK or JRE home directory path.

**string JavaVersion ()**

Gets latest version of either the JDK or JRE version stored in registry.

**string JDKBits ()**

Gets Java Development Kit (JDK) bits stored in registry.

**string JDKHome ()**

Gets Java Development Kit (JDK) home directory path.

**string JDKVersion ()**

Gets Java Development Kit (JDK) version stored in registry.

**string JREBits ()**

Gets Java Runtime Environment (JRE) bits stored in registry.

**string JREHome ()**

Gets Java Runtime Environment (JRE) home directory path.

**string JREVersion ()**

Gets Java Runtime Environment (JRE) version stored in registry.

**bool RequestedPipeline (string pipelineName)**

Determine if 'pipelineName' has been scheduled for processing.

**Detailed Description**

urn:WebWorks-XSLT-Extension-Environment

Enable XSL transforms to query the current system environment for the location and state of programs and variables.

**string ApplicationBaseHelpURI ()**

Application base help URI.

**Returns:**

A string.

### **string CurrentUILocale ()**

Current user interface locale.

#### **Returns:**

A string.

### **long GetTotalMemory ()**

Reports the total amount of memory used by the running process. Useful for optimizing XSL to reduce memory usage.

#### **Returns:**

The total memory in bytes.

### **long GetTotalMemory (bool forceFullCollection)**

Reports the total amount of memory used by the running process. Useful for optimizing XSL to reduce memory usage.

#### **Parameters:**

forceFullCollection	True to force full collection.
---------------------	--------------------------------

**Returns:**

The total memory in bytes.

**string HTMLHelpWorkshopPath ()**

HTML help workshop path.

**Returns:**

A string.

**string JavaBits ()**

Gets latest version of either JDK pr JRE bits stored in registry.

**Returns:**

The JDK/JRE bits string stored in registry.

**string JavaHome ()**

Gets latest version of either JDK or JRE home directory path.

**Returns:**

JDK/JRE home directory path string.

**string JavaVersion ()**

Gets latest version of either the JDK or JRE version stored in registry.

**Returns:**

JDK/JRE version string stored in registry.

### **string JDKBits ()**

Gets Java Development Kit (JDK) bits stored in registry.

#### **Returns:**

JDK bits string stored in registry.

### **string JDKHome ()**

Gets Java Development Kit (JDK) home directory path.

#### **Returns:**

JDK home directory path string.

Determine the path to the current JDK and jar some files.

```
<xsl:variable name="VarJDKHome" select="wwenv:JDKHome()" />

<xsl:variable name = "VarJarPath"
  select="wwfilesystem:Combine($VarJDKHome, 'jar')" />

<xsl:variable name = "VarJarCommand"
  select="wwexec:ExecuteCommand($VarJarPath, 'cvf', 'output.jar',
    '.')" />
```

### **string JDKVersion ()**

Gets Java Development Kit (JDK) version stored in registry.

#### **Returns:**

JDK version string stored in registry.

### **string JREBits ()**

Gets Java Runtime Environment (JRE) bits stored in registry.

#### **Returns:**



The JRE bits string stored in registry

**string JREHome ()**

Gets Java Runtime Environment (JRE) home directory path.

**Returns:**

JRE home directory path string.

**string JREVersion ()**

Gets Java Runtime Environment (JRE) version stored in registry.

**Returns:**

JRE version string stored in registry.

**bool RequestedPipeline (string pipelineName)**

Determine if 'pipelineName' has been scheduled for processing.

**Parameters:**

pipelineName	Name of the pipeline.
--------------	-----------------------

## Returns:

True if scheduled for processing, false otherwise.

Generate a given report only if specifically enabled in a project or a user has specifically requested that report.

```
<xsl:variable name="VarGenerateReportSetting"
  select="wwproject:GetFormatSetting('report-filenames-generate',
  'true') = 'true'" /> <xsl:variable name = "VarRequestedPipeline"
  select="wwenv:RequestedPipeline($GlobalPipelineName)" />

<xsl:variable name = "VarGenerateReport"
  select="($VarGenerateReportSetting) or ($VarRequestedPipeline)" />
```

## Exec

urn:WebWorks-XSLT-Extension-Execute

## Functions

### XPathNodeIterator Execute (string commandLine)

Runs the command line.

**XPathNodeIterator ExecuteCommand (string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the current target's output directory as the working directory.

### XPathNodeIterator ExecuteCommandNoReturn (string command)

Runs identical to ExecuteCommand (string command) with the exception that the stdout and stderr streams are not returned in the node set. Instead these are written to the Log directory in a file with the same base filename as the command.

**XPathNodeIterator ExecuteCommandInDirectory (string directoryPath, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6,**

**string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the specified directory as the working directory.

**XPathNodeIterator ExecuteCommandInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the specified directory as the working directory.

**XPathNodeIterator ExecuteCommandWithTimeout (long timeoutInSeconds, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the current target's output directory as the working directory.

**XPathNodeIterator ExecuteInDirectory (string directoryPath, string commandLine)**

Runs the command line in the specified working directory.

**XPathNodeIterator ExecuteInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string commandLine)**

Runs the command line in the specified working directory.

**XPathNodeIterator ExecuteProgramWithArguments (string program, string arguments)**

Runs the specified program with arguments formatted as a string using the current target's output directory as the working directory.

### **XPathNodeIterator ExecuteProgramWithArgumentsInDirectory (string directoryPath, string program, string arguments)**

Runs the specified program with arguments formatted as a string in the specified working directory.

### **XPathNodeIterator**

#### **ExecuteProgramWithArgumentsInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string program, string arguments)**

Runs the specified program with arguments formatted as a string in the specified working directory.

### **XPathNodeIterator ExecuteProgramWithArgumentsWithTimeout (long timeoutInSeconds, string program, string arguments)**

Runs the specified program with arguments formatted as a string using the current target's output directory as the working directory.

### **XPathNodeIterator ExecuteWithTimeout (long timeoutInSeconds, string commandLine)**

Runs the command line.

### **Detailed Description**

urn:WebWorks-XSLT-Extension-Execute

Allows XSL stylesheets to execute external programs and process results. Provides the return code, stdout, and stderr results from the running process.

```
<wwexec:Result version="1.0" retcode="-1">
```

```
  <wwexec:Stream name = "Output" >
```

```
    Standard output will show up here, aka stdout.
```

```
  </wwexec:Stream>
```

```
  <wwexec:Stream name = "Error" >
```

```
    Standard error will show up here, aka stderr.
```

```
  </wwexec:Stream>
```

```
</wwexec:Result>
```

## **XPathNodeIterator Execute (string commandLine)**

Runs the command line.

### **Parameters:**

commandLine	The command line to be executed.
-------------	----------------------------------

### Returns:

A node set.

Execute prettycool.exe -stdout "Isn't this grand?" -stderr nothing.

```
<xsl:variable name="VarExecResult"
  select="wwexec:Execute('prettycool.exe -stdout &quot;Isn\'t this
grand?&quot; --stderr nothing') " />
```

**XPathNodeIterator ExecuteCommand (string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the current target's output directory as the working directory.

### Parameters:

command	The command.
argument1 – argument20	Optional arguments.

### **Returns:**

A node set.

### **XPathNodeIterator ExecuteCommandNoReturn (string command)**

Runs identical to ExecuteCommand (string command) with the exception that the stdout and stderr streams are not returned in the node set. Instead these are written to the Log directory in a file with the same base filename as the command parameter.

### **Parameters:**

command	The command.
---------	--------------

### Returns:

A node set.

**XPathNodeIterator ExecuteCommandInDirectory (string directoryPath, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the specified directory as the working directory.

### Parameters:



directoryPath	Full pathname of the directory file.
command	The command.
argument1 – argument20	Optional arguments.

### Returns:

A node set.

**XPathNodeIterator ExecuteCommandInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the specified directory as the working directory.

Will stop operation if timeout in seconds elapses.

### Parameters:

timeoutInSeconds	The timeout in seconds.
directoryPath	Full pathname of the directory file.
command	The command.
argument1 – argument20	Optional arguments.

### Returns:

A node set.

**XPathNodeIterator ExecuteCommandWithTimeout (long timeoutInSeconds, string command [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19, string argument20])**

Runs the specified command formatted as a string with optional argument(s) using the current target's output directory as the working directory.

Will stop operation if timeout in seconds elapses.

### Parameters:

timeoutInSeconds	The timeout in seconds.
command	The command.
argument1 – argument20	Optional arguments.

**Returns:**

A node set.

**XPathNodeIterator ExecuteInDirectory (string directoryPath, string commandLine)**

Runs the command line in the specified working directory.

**Parameters:**

directoryPath	Full pathname of the directory.
commandLine	The command line to be executed.

### Returns:

A node set.

Execute prettycool.exe -stdout "Isn't this grand?" -stderr nothing in the directory C:\workingdir.

```
<xsl:variable name="VarExecResult"
  select="wwexec:ExecuteInDirectory('C:\\workingdir', 'prettycool.exe
  --stdout &quot;Isn\'t this grand?&quot; --stderr nothing')" />
```

### XPathNodeIterator ExecuteInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string commandLine)

Runs the command line in the specified working directory.

Will stop operation after specified elapsed time is exceeded.

### Parameters:

timeoutInSeconds	The timeout in seconds.
directoryPath	Full pathname of the directory.
commandLine	The command line to be executed.

### **Returns:**

A node set.

### **XPathNodeIterator ExecuteProgramWithArguments (string program, string arguments)**

Runs the specified program with arguments formatted as a string using the current target's output directory as the working directory.

### **Parameters:**

program	The program.
arguments	The arguments.

### Returns:

A node set.

Runs the specified program with arguments formatted as a string using the current target's output directory as the working directory.

```
<xsl:variable name="VarExecResult"
  select="wwexec:ExecuteProgramWithArguments('prettycool.exe', '--
stdout &quot;Isn\'t this grand?&quot; --stderr nothing')" />
```

### XPathNodeIterator ExecuteProgramWithArgumentsInDirectory (string directoryPath, string program, string arguments)

Runs the specified program with arguments formatted as a string in the specified working directory.

### Parameters:

directoryPath	Full pathname of the directory.
program	The program.
arguments	The arguments.

### Returns:

A node set.

Execute prettycool.exe -stdout "Isn't this grand?" -stderr nothing in the directory C:\workingdir.

```
<xsl:variable name="VarExecResult"
  select="wwexec:ExecuteProgramWithArgumentsInDirectory('C:\
\workingdir', 'prettycool.exe', '--stdout &quot;Isn\'t this grand?
&quot; --stderr nothing')" />
```

### **XPathNodeIterator ExecuteProgramWithArgumentsInDirectoryWithTimeout (long timeoutInSeconds, string directoryPath, string program, string arguments)**

Runs the specified program with arguments formatted as a string in the specified working directory.

Will stop operation after specified elapsed time is exceeded.

### Parameters:

timeoutInSeconds	The timeout in seconds.
directoryPath	Full pathname of the directory.
program	The program.
arguments	The arguments.

### Returns:

A node set.

### **XPathNodeIterator ExecuteProgramWithArgumentsWithTimeout (long timeoutInSeconds, string program, string arguments)**

Runs the specified program with arguments formatted as a string using the current target's output directory as the working directory.

Will stop operation after specified elapsed time is exceeded.

### Parameters:



timeoutInSeconds	The timeout in seconds.
program	The program.
arguments	The arguments.

### **Returns:**

A node set.

**XPathNodeIterator ExecuteWithTimeout (long timeoutInSeconds, string commandLine)**

Runs the command line.

Will stop operation after specified elapsed time is exceeded.

### **Parameters:**

timeoutInSeconds	The timeout in seconds.
commandLine	The command line to be executed.

### Returns:

A node set.

## ExecPython

urn:WebWorks-XSLT-Extension-ExecPython

### Functions

#### **XPathNodeIterator ExecutePyScriptInCommandLine (string commandLine)**

Runs python with the specified commandLine.

#### **XPathNodeIterator ExecPyScript (string pyScriptPath [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19])**

Runs python with the specified script with zero or more arguments using the current target's output directory as the working directory.

#### **XPathNodeIterator ExecutePyScriptInDirectoryInCommandLine (string directoryPath, string commandLine)**

Runs python with the specified script with zero or more arguments using the specified directory as the working directory.

#### **XPathNodeIterator ExecPyScriptInDirectory (string directoryPath, string pyScriptPath [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19])**

Runs python with the specified script with zero or more arguments using the specified directory as the working directory.

### Detailed Description

urn:WebWorks-XSLT-Extension-ExecPython

Allows XSL stylesheets to execute python programs and process results. Provides the return code, stdout, and stderr results from the running process.

```
<wwexec:Result version="1.0" retcode="-1">
```

```
  <wwexec:Stream name = "Output" >
```

```
    Standard output will show up here, aka stdout.
```

```
  </wwexec:Stream>
```

```
  <wwexec:Stream name = "Error" >
```

```
    Standard error will show up here, aka stderr.
```

```
  </wwexec:Stream>
```

```
</wwexec:Result>
```

### XPathNodeIterator ExecutePyScriptInCommandLine (string commandLine)

Runs python with the specified command line.

### Parameters:

commandLine	The command line in python to be executed.
-------------	--

### Returns:

A node set.

```
<xsl:variable name="VarExecResult"
  select="wwpython:ExecutePyScriptInCommandLine('script.py
  some_arg') " />
```

**XPathNodeIterator ExecPyScript (string pyScriptPath [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19])**

Runs python with the specified script with zero or more arguments using the current target's output directory as the working directory.

### Parameters:

pyScriptPath	Path to the script file.
argument1 – argument19	Optional arguments.

### Returns:

A node set.

```
<xsl:variable name="VarExecResult" select="
  wwpython:ExecPyScript('script.py', 'some_arg') " />
```

### **XPathNodeIterator ExecutePyScriptInDirectoryInCommandLine (string directoryPath, string commandLine)**

Runs python with the specified script with zero or more arguments using the specified directory as the working directory.

### Parameters:

directoryPath	The working directory path where the python command line is going to be executed.
commandLine	The command line in python to be executed.

### Returns:

A node set.

**XPathNodeIterator ExecPyScriptInDirectory (string directoryPath, string pyScriptPath [, string argument1, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10, string argument11, string argument12, string argument13, string argument14, string argument15, string argument16, string argument17, string argument18, string argument19])**

Runs python with the specified script with zero or more arguments using the specified directory as the working directory.

### Parameters:

directoryPath	The working directory path where the python command line is going to be executed.
pyScriptPath	Path to the script file.
argument1 – argument19	Optional arguments.

### Returns:

A node set.

## Sass

urn:WebWorks-XSLT-Extension-Sass

### Functions

#### **XPathNodeIterator SassToCss (string inputSassFilePath, string outputCssFilePath)**

Creates a CSS file out of a SASS file.

#### **void ReplaceAllVariablesInFile (string inputSassFilePath, object replacements)**

Replaces a collection of SASS variables in a file with those specified by the user.

### Detailed Description

urn:WebWorks-XSLT-Extension-Sass

Allows XSL stylesheets to manage SASS files.

#### **XPathNodeIterator SassToCss (string inputSassFilePath, string outputCssFilePath)**

Creates a CSS file out of a SASS file.

### Parameters:

inputSassFilePath	Path to the input SASS file.
outputCssFilePath	Path to the output CSS file.

**Returns:**

A node set.

```
<xsl:variable name="VarSassToCssResult"
  select="wvsass:SassToCss('input.sass', 'output.css')" />
```

**void ReplaceAllVariablesInFile (string inputSassFilePath, object replacements)**

Replaces a collection of SASS variables in a file with those specified by the user.

**Parameters:**



inputSassFilePath	Path to the input SASS file.
replacements	Object that represents an XML collection of SASS variables.

Replaces matching variables in the input file with variables specified inside the replacements object.

```
<xsl:variable name="VarSassVariableReplacementsAsXML">
  <wwsass:Variable name="footer_height" value="50px" />
  <wwsass:Variable name="menu_width" value="100px" />
</xsl:variable>
<xsl:variable name = "VarSassVariableReplacements" select="msxsl:node-set($VarSassVariableReplacementsAsXML)/*" />

<xsl:variable name="VarReplaceVariables"
  select="wwsass:ReplaceAllVariablesInFile('file.scss',
    $VarSassVariableReplacements)"/>
```

## ExslDocument

urn:WebWorks-XSLT-Extension-Document

### Functions

**void Document (object input, string path [, string encoding, string method, string version, string indent, string omit\_xml\_declaration, string standalone, string doctype\_public, string doctype\_system, string cdata\_section\_elements, string media\_type])**

Writes the specified node set to a file at the location defined by path. Optionally uses specified encoding for writing content. Optionally uses specified method (i.e. text, xhtml, xml...) and version. Optionally specify if file is to be formatted with indentation with 'yes' or 'no'. Optionally specify if file will have XML declaration with 'yes' or 'no'. Optionally specify if the file will be standalone with 'yes' or 'no'. Optionally specify public and system doctype modifiers.

**XPathNodeIterator LoadXMLWithoutResolver (string uriAsString [, bool preserveSpace])**

Loads an XML file without resolving internal paths and validation DTDs.

**XPathNodeIterator LoadXMLWithResolver (string uriAsString [, bool preserveSpace])**

Loads an XML file while also resolving internal paths and validating DTDs.

### **XPathNodeIterator MakeEmptyElement (object input)**

Converts a non-empty XML node to an empty node.

### **Detailed Description**

urn:WebWorks-XSLT-Extension-Document

Allow multiple output files from an single XSL transform. Also provides routines to quickly load XML files without invoking XML validators as well as utility methods to enable correct output formatting.

**void Document (object input, string path [, string encoding, string method, string version, string indent, string omit\_xml\_declaration, string standalone, string doctype\_public, string doctype\_system, string cdata\_section\_elements, string media\_type])**

Writes the specified node set to a file at the location defined by path. Optionally uses specified encoding for writing content. Optionally uses specified method (i.e. text, xhtml, xml...) and version. Optionally specify if file is to be formatted with indentation with 'yes' or 'no'. Optionally specify if file will have XML declaration with 'yes' or 'no'. Optionally specify if the file will be standalone with 'yes' or 'no'. Optionally specify public and system doctype modifiers.

### **Parameters:**

input	The input.
path	Full pathname of the file.
encoding	The encoding (optional).
method	The method (optional).
version	The version (optional).
indent	Indent 'yes' or 'no' (optional).
omit_xml_declaration	Omit XML declaration 'yes' or 'no' (optional).
standalone	Standalone 'yes' or 'no' (optional).
doctype_public	The doctype public (optional).
doctype_system	The doctype system (optional).
cdata_section_elements	The cdata section elements (optional).
media_type	Type of the media (optional).

Write a node set to a file.

```
<xsl:variable name="VarResultAsXML">

...

</xsl:variable>

<xsl:variable name = "VarResult" select="msxsl:node-
set($VarResultAsXML)/*" />

<xsl:variable name = "VarWriteDocument"
  select="wwexsldoc:Document($VarResult, 'C:\badplacefor.xml') " />
```

**XPathNodeIterator LoadXMLWithoutResolver (string uriAsString [, bool preserveSpace])**

Loads an XML file without resolving internal paths and validation DTDs.

## Parameters:

uriAsString	The URI as string.
preserveSpace	True to preserve space (optional).

### Returns:

A node set.

Load the page template without resolving attribute paths.

```
<xsl:variable name="VarPageTemplate"
  select="wwexsldoc:LoadXMLWithoutResolver($VarPathTemplatePath)" />
```

### **XPathNodeIterator LoadXMLWithResolver (string uriAsString [, bool preserveSpace])**

Loads an XML file while also resolving internal paths and validating DTDs.

### Parameters:

uriAsString	The URI as string.
preserveSpace	True to preserve space (optional).

**Returns:**

A node set.

**XPathNodeIterator MakeEmptyElement (object input)**

Converts a non-empty XML node to an empty node.

**Parameters:**

input	The input node set.
-------	---------------------

### Returns:

A node set containing a single empty node.

```

```

```
</img>
```

### becomes:

```

```

Insure <img> element is emitted as an empty XML node for proper display in HTML browsers.

```
<xsl:variable name="VarImageElementAsXML">

<xsl:element name = "img" >

  < xsl:attribute name = "src" >

    < xsl:value-of select = "'blue.jpg'" />

  </ xsl:attribute>

</xsl:element>

</xsl:variable>

<xsl:variable name = "VarImageElement" select="msxsl:node-
set ($VarImageElementAsXML)/*" />

<xsl:value-of select =
  "wwexsldoc:MakeEmeptyElement ($VarImageElement)" />
```

## Files

urn:WebWorks-XSLT-Extension-Files

### Functions

**bool UpToDate (string path, string projectChecksum, string groupID, string documentID, string actionChecksum)**

Compares the provided parameters for a given path against a previously recorded values. If the values match, the result is true(). Otherwise, the result is false().

### **Detailed Description**

urn:WebWorks-XSLT-Extension-Files

Enables incremental build support.

**bool UpToDate (string path, string projectChecksum, string groupID, string documentID, string actionChecksum)**

Compares the provided parameters for a given path against a previously recorded values. If the values match, the result is true(). Otherwise, the result is false().

### **Exceptions:**



OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
projectChecksum	The project checksum.
groupID	Identifier for the group.
documentID	Identifier for the document.
actionChecksum	The action checksum.

### Returns:

True if it succeeds, false if it fails.

## FileSystem

urn:WebWorks-XSLT-Extension-FileSystem

### Functions

#### **bool AppendFileWithFile (string targetPath, string sourcePath)**

Appends one file to another file.

#### **bool ChecksumUpToDate (string path, string checksum)**

Compares the provided checksum with the current checksum.

#### **string Combine (string path, string component1 [, string component2, string component3, string component4, string component5, string component6, string component7, string component8, string component9, string component10])**

Combines the given component(s) to the file path.

#### **XPathNodeIterator CopyDirectoryFiles (string sourceDirectoryPath, string destinationDirectoryPath)**

Copies all files from the source directory to the destination directory. Destination files are reported as <FILE> elements. Source files are reported as <Depends> elements for each destination file.

#### **XPathNodeIterator CopyFile (string sourcePath, string destinationPath)**

Copies the source file to the destination path. Destination files are reported as <FILE> elements. Source files are reported as <Depends> elements for each destination file.

### **bool CreateDirectory (string path)**

Creates a directory with the given path. If the directory already exists, the method will return false().

### **void DeleteDirectory (string path)**

Recursively deletes the directory with the given path.

### **void DeleteFile (string path)**

Deletes the file described by path.

### **bool DirectoryExists (string path)**

Determines if a directory exists at the given path. If a file exists with the given path, this method will return false().

### **bool Exists (string path)**

Determine if file or directory 'path' exists.

### **bool FileExists (string path)**

Determines if a file exists at the given path. If directory exists with the given path, this method will return false().

### **bool FilesEqual (string alphaPath, string betaPath)**

Compares the contents of two files to determine if they are equal.

### **string GetAbsolutePath (string relativePath, string referencePath)**

Determines an absolute path given a relative path and a reference path to create the absolute path from. Returns the relative path argument if it is absolute to begin with.

### **string GetBaseName (string path)**

Gets the base name of any path. It handles as separators: '\' and '/'. If the path ends with a separator or it's an empty string then it returns an empty string.

### **string GetChecksum (string path)**

Determines the checksum for the specified file.

### **string GetDirectoryName (string path)**

Determines the directory prefix of the given path.

**string GetExtension (string path)**

Determines the file extension for the given path.

**string GetFileName (string path)**

Determines the name of the file with the directory prefix removed.

**string GetFileNameWithoutExtension (string path)**

Determines the name of the file with the directory prefix and extension removed.

**XPathNodeIterator GetFiles (string path)**

Returns an XML node set containing absolute file paths to all files in the specified path. If the path specifies a file, a single file entry will be returned. If the path specifies a directory, all file paths in the directory and their children are returned.

**string GetLongPathName (string path)**

Gets long path name of a specified short path filename.

**XPathNodeIterator GetRelativeFiles (string path)**

Returns an XML node set containing RELATIVE file paths to all files in the specified path. If the path specifies a file, a single file entry will be returned. If the path specifies a directory, all file paths in the directory and their children are returned.

**string GetRelativeTo (string path, string anchorPath)**

Determines the relative path from the absolute anchor path to the absolute destination path. May return an absolute path if no relative path exists.

**string GetShortPathName (string path)**

Gets short path filename from a specified filename.

**string GetTempFileName ()**

Gets temporary unique filename path.

**string GetTempPath ()**

Gets path to the user's temporary files directory.

**string GetWithExtensionReplaced (string path, string extension)**

Replaces the current file extension with the provided one.

### **string MakeValidFileName (string fileNameSeed)**

Makes a valid filename by eliminating the invalid characters from the specified seed filename.

### **void TranslateFileToEncoding (string sourceFilePath, string sourceFileEncodingName, string destinationFilePath, string destinationFileEncodingName)**

Translate file to encoding.

### **Detailed Description**

urn:WebWorks-XSLT-Extension-FileSystem

Allow XSL transforms to query and manipulate files and directories. Also handles system path parsing and processing.

### **bool AppendFileWithFile (string targetPath, string sourcePath)**

Appends one file to another file.

### **Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

targetPath	Full pathname of the target file.
sourcePath	Full pathname of the source file.

### Returns:

True if it succeeds, false if it fails.

Append the contents of "C:\FileSampleDoc.txt" to "C:\OutputAllContent.txt".

```
<xsl:variable name="VarADoc">C:\\File\Sample\Doc.txt</xsl:variable>
```

```
<xsl:variable name = "VarAllDocs" > C:\\Output\All\Content.txt</xsl:variable>
```

```
<xsl:variable name = "ActionAppendingDocToAllContent"
  select="wwfilesystem:AppendFileWithFile($VarAllDocs, $VarADoc)" />
```

### bool ChecksumUpToDate (string path, string checksum)

Compares the provided checksum with the current checksum.

This is a convenience method for XSL developers.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**



path	Full pathname of the file.
checksum	The checksum.

### Returns:

True if the same, false if not the same.

**string Combine (string path, string component1 [, string component2, string component3, string component4, string component5, string component6, string component7, string component8, string component9, string component10])**

Combines the given component(s) to the file path.

### Parameters:

path	Full pathname of the file.
component1	The first component.
component2 – component10	Additional components (optional).

### Returns:

A string.

### **XPathNodeIterator CopyDirectoryFiles (string sourceDirectoryPath, string destinationDirectoryPath)**

Copies all files from the source directory to the destination directory. Destination files are reported as <FILE> elements. Source files are reported as <Depends> elements for each destination file.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

sourceDirectoryPath	Pathname of the source directory.
destinationDirectoryPath	Pathname of the destination directory.

**Returns:**

A <Files></Files> node set.

**XPathNodeIterator CopyFile (string sourcePath, string destinationPath)**

Copies the source file to the destination path. Destination files are reported as <FILE> elements. Source files are reported as <Depends> elements for each destination file.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

sourcePath	Full pathname of the source file.
destinationPath	Full pathname of the destination file.

### Returns:

A <Files> node set.</Files>

### **bool CreateDirectory (string path)**

Creates a directory with the given path. If the directory already exists, the method will return false().

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

### Returns:

True if it succeeds, false if it fails.

Create directory C: if it does not exist.

```
<xsl:if test="wwfilesystem:Exists('C:\\exists')">

  <xsl:variable name = "VarCreateDirectory"
    select="wwfilesystem:CreateDirectory('C:\\exists') " />

</xsl:if>
```

### void DeleteDirectory (string path)

Recursively deletes the directory with the given path.

### Exceptions:



OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the directory.
------	---------------------------------

Delete the directory C:.

```
<xsl:variable name="VarDeleteDirectory"
  select="wwfilesystem:DeleteDirectory('C:\\deleteme') " />
```

**void DeleteFile (string path)**

Deletes the file described by path.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

### **bool DirectoryExists (string path)**

Determines if a directory exists at the given path. If a file exists with the given path, this method will return false().

### **Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

### Returns:

True if it succeeds, false if it fails.

Log existence of directory C:.

```
<xsl:if test="wwfilesystem:DirectoryExists('C:\\direxists')">
  <xsl:variable name = "VarLog" select="wwlog:Message('Directory \',
    'C:\\direxists', '\\ exists!')" />
</xsl:if>
```

### bool Exists (string path)

Determine if file or directory 'path' exists.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

### Returns:

True if it succeeds, false if it fails.

Determine if a file or directory exists at the given path.

```
<xsl:if test="wwfilesystem:Exists('C:\\exists')">

  <xsl:variable name = "VarCreateDirectory"
    select="wwfilesystem:CreateDirectory('C:\\exists') " />

</xsl:if>
```

### bool FileExists (string path)

Determines if a file exists at the given path. If directory exists with the given path, this method will return false().

### Exceptions:



OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

### Returns:

True if it succeeds, false if it fails.

Log existence of file C:.

```
<xsl:if test="wwfilesystem:DirectoryExists('C:\\fileexists')">
  <xsl:variable name = "VarLog" select="wwlog:Message('Directory \'',
    'C:\\fileexists', '\\ exists!')" />
</xsl:if>
```

### bool FilesEqual (string alphaPath, string betaPath)

Compares the contents of two files to determine if they are equal.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

alphaPath	Full pathname of the first file.
betaPath	Full pathname of the second file.

### Returns:

True if equal, false if not.

### **string GetAbsoluteFrom (string relativePath, string referencePath)**

Determines an absolute path given a relative path and a reference path to create the absolute path from. Returns the relative path argument if it is absolute to begin with.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

relativePath	Pathname of the relative file.
referencePath	Full pathname of the reference file.

### Returns:

The absolute from.

### **string GetBaseName (string path)**

Gets the base name of any path. It handles as separators: '\' and '/'. If the path ends with a separator or it's an empty string then it returns an empty string.

### Parameters:

path	The relative or absolute path.
------	--------------------------------

**Returns:**

A base name string.

**string GetChecksum (string path)**

Determines the checksum for the specified file.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**



path	Full pathname of the file.
------	----------------------------

**Returns:**

The checksum as string.

**string GetDirectoryName (string path)**

Determines the directory prefix of the given path.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

**Returns:**

The directory name.

**string GetExtension (string path)**

Determines the file extension for the given path.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

**Returns:**

The filename extension.

**string GetFileName (string path)**

Determines the name of the file with the directory prefix removed.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

**Returns:**

The file basename.

**string GetFileNameWithoutExtension (string path)**

Determines the name of the file with the directory prefix and extension removed.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**



path	Full pathname of the file.
------	----------------------------

### **Returns:**

The file name without extension or directory prefix.

### **XPathNodeIterator GetFiles (string path)**

Returns an XML node set containing absolute file paths to all files in the specified path. If the path specifies a file, a single file entry will be returned. If the path specifies a directory, all file paths in the directory and their children are returned.

### **Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file or directory.
------	---

**Returns:**

The files in a node set.

**string GetLongPathName (string path)**

Gets long path name of a specified short path filename.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

### Returns:

The long path filename.

### **XPathNodeIterator GetRelativeFiles (string path)**

Returns an XML node set containing RELATIVE file paths to all files in the specified path. If the path specifies a file, a single file entry will be returned. If the path specifies a directory, all file paths in the directory and their children are returned.

Filename path(s) created will be relative to 'relativeToPath' (if non-zero in length).

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

### Returns:

The relative files in a node set.

### **string GetRelativeTo (string path, string anchorPath)**

Determines the relative path from the absolute anchor path to the absolute destination path. May return an absolute path if no relative path exists.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**



path	Full pathname of the file.
anchorPath	Full pathname of the anchor file.

### Returns:

The relative to path.

### **string GetShortPathName (string path)**

Gets short path filename from a specified filename.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
------	----------------------------

**Returns:**

The short path name.

**string GetTempFileName ()**

Gets temporary unique filename path.

**Returns:**

The temporary filename path.

**string GetTempPath ()**

Gets path to the user's temporary files directory.

**Returns:**

The directory path ending with backslash.

**string GetWithExtensionReplaced (string path, string extension)**

Replaces the current file extension with the provided one.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

path	Full pathname of the file.
extension	The extension.

### Returns:

The with extension replaced.

### **string MakeValidFileName (string fileNameSeed)**

Makes a valid filename by eliminating the invalid characters from the specified seed filename.

### Parameters:

fileNameSeed	The file name seed.
--------------	---------------------

**Returns:**

A filename string.

**void TranslateFileToEncoding (string sourceFilePath, string sourceFileEncodingName, string destinationFilePath, string destinationFileEncodingName)**

Translate file to encoding.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

sourceFilePath	Full pathname of the source file.
sourceFileEncodingName	Name of the source file encoding.
destinationFilePath	Full pathname of the destination file.
destinationFileEncodingName	Name of the destination file encoding.

## Fonts

urn:WebWorks-XSLT-Extension-Fonts

### Functions

#### **bool UnicodeFont (string fontFamily)**

Determines if the specified font family is a Unicode font family. Used to detect Symbol font families.

#### **Detailed Description**

urn:WebWorks-XSLT-Extension-Fonts

Answer questions about fonts that might affect format output.

#### **bool UnicodeFont (string fontFamily)**

Determines if the specified font family is a Unicode font family. Used to detect Symbol font families.

#### **Parameters:**



fontFamily	The font family.
------------	------------------

### Returns:

True if a unicode font, false if not.

Determine if Symbol is a unicode font family.

```
<xsl:variable name="VarIsUnicodeFont"
  select="wwfonts:UnicodeFont('Symbol')" />
```

## Imaging

urn:WebWorks-Imaging-Info

### Functions

#### **XPathNodeIterator GetInfo (string imagePath)**

Gets image information of file path: 'imageFilePath'.

#### **void MapPDFLinks (object fileTable, string fileToFix, string fileToWrite, string originalFilePath, string outputPath, bool useAbsPath)**

Map PDF links.

#### **bool MergePDFs (object sourceFileList, string targetFilePath)**

This merges a set of PDFs and/or PostScript files into one PDF.

#### **bool MergePDFs (object sourceFileList, object fileTable, string targetFilePath)**

This merges a set of PDFs and/or PostScript files into one PDF.

#### **bool PostScriptToPDF (string postScriptFilePath, string pdfJobSettings, string pdfFilePath)**

Convert the specified PostScript file to a PDF.

#### **XPathNodeIterator RasterizePostScript (string postScriptFilePath, int renderHorizontalDPI, int renderVerticalDPI, int renderWidth, int renderHeight, string targetImageFormatAsString, int targetImageColorDepth, bool targetImageGrayscale, bool**

**targetImageTransparent, bool targetImageInterlaced, int targetImageQuality, string targetFilePath)**

Render a PostScript file to a known image format such as BMP, JPEG, PNG, or GIF.

**XPathNodeIterator Transform (string inputImageFilePath, string outputImageFormat, int outputImageWidth, int outputImageHeight, string outputImageFilePath)**

Create a new version of an image with a different format or with different dimensions.

**XPathNodeIterator Transform (string inputImageFilePath, string outputImageFormatAsString, int Choice\_outputImageQuality\_outputImageWidth, int Choice\_outputImageWidth\_outputImageHeight, string Choice\_outputImageHeight\_outputImageFilePath, string Choice\_outputImageFilePath\_outputResolution)**

Create a new version of an image with a different format or with different dimensions.

**XPathNodeIterator Transform (string inputImageFilePath, string outputImageFormatAsString, int outputImageQuality, int outputImageWidth, int outputImageHeight, string outputImageFilePath, int outputResolution)**

Create a new version of an image with a different format or with different dimensions.

### Detailed Description

urn:WebWorks-Imaging-Info

Enable processing of images within XSL transforms.

Returns information about a particular image file, including width and height, image format, bit-depth, path on system, etc.

```
<wwimageinfo:ImageInfo format="jpeg" width="200" height="300"
  bitdepth="32" grayscale="false" path="C:\\image.jpg" />
```

**XPathNodeIterator GetInfo (string imageFilePath)**

Gets image information of file path: 'imageFilePath'.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

imageFilePath	Full pathname of the image file.
---------------	----------------------------------

### Returns:

Image info node set.

**void MapPDFLinks (object fileTable, string fileToFix, string fileToWrite, string originalFilePath, string outputFilePath, bool useAbsPath)**

Map PDF links.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

fileTable	Node set of links to update.
fileToFix	The file to fix.
fileToWrite	The file to write.
originalFilePath	Full pathname of the original file.
outputFilePath	Full pathname of the output file.
useAbsPath	True to use abs path.

**bool MergePDFs (object sourceFileList, string targetFilePath)**

This merges a set of PDFs and/or PostScript files into one PDF.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

sourceFileList	List of source files.
targetFilePath	Full pathname of the target file.

**Returns:**

True if it succeeds, false if it fails.

**bool MergePDFs (object sourceFileList, object fileTable, string targetFilePath)**

This merges a set of PDFs and/or PostScript files into one PDF.

**Exceptions:**



OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

sourceFileList	List of source files.
fileTable	Node set of links to update.
targetFilePath	Full pathname of the target file.

### Returns:

True if it succeeds, false if it fails.

**bool PostScriptToPDF (string postScriptFilePath, string pdfJobSettings, string pdfFilePath)**

Convert the specified PostScript file to a PDF.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

postScriptFilePath	Full pathname of the post script file.
pdfJobSettings	The PDF job settings.
pdfFilePath	Full pathname of the PDF file.

### Returns:

True if it succeeds, false if it fails.

**XPathNodeIterator RasterizePostScript (string postScriptFilePath, int renderHorizontalDPI, int renderVerticalDPI, int renderWidth, int renderHeight, string targetImageFormatAsString, int targetImageColorDepth, bool targetImageGrayscale, bool targetImageTransparent, bool targetImageInterlaced, int targetImageQuality, string targetFilePath)**

Render a PostScript file to a known image format such as BMP, JPEG, PNG, or GIF.

### Parameters:

postScriptFilePath	Full pathname of the post script file.
renderHorizontalDPI	The render horizontal DPI.
renderVerticalDPI	The render vertical DPI.
renderWidth	Width of the render.
renderHeight	Height of the render.
targetImageFormatAsString	Target image format as string.
targetImageColorDepth	Depth of the target image color.
targetImageGrayscale	True to target image grayscale.
targetImageTransparent	True to target image transparent.
targetImageInterlaced	True if target image interlaced.
targetImageQuality	Target image quality.
targetFilePath	Full pathname of the target file.

### Returns:

Image info node set.

**XPathNodeIterator Transform (string inputImagePath, string outputImageFormat, int outputImageWidth, int outputImageHeight, string outputImagePath)**

Create a new version of an image with a different format or with different dimensions.

### Parameters:

inputImagePath	Full pathname of the input image file.
outputImageFormat	The output image format.
outputImageWidth	Width of the output image.
outputImageHeight	Height of the output image.
outputImageFilePath	Full pathname of the output image file.

### Returns:

Image info node set.

**XPathNodeIterator Transform (string inputImagePath, string outputImageFormatAsString, int Choice\_outputImageQuality\_outputImageWidth, int Choice\_outputImageWidth\_outputImageHeight, string Choice\_outputImageHeight\_outputImageFilePath, string Choice\_outputImageFilePath\_outputResolution)**

Create a new version of an image with a different format or with different dimensions.

### Parameters:

inputImagePath	Full pathname of the input image file.
outputImageFormatAsString	The output image format as string.
Choice_outputImageQuality_outputImageWidth	Width of the choice output image quality output image.
Choice_outputImageWidth_outputImageHeight	Height of the choice output image width output image.
Choice_outputImageHeight_outputImageFilePath	File path of the choice output image height output image file.
Choice_outputImageFilePath_outputResolution	The choice output image file path output resolution.

### Returns:

Image info node set.

**XPathNodeIterator Transform (string inputImagePath, string outputImageFormatAsString, int outputImageQuality, int outputImageWidth, int outputImageHeight, string outputImageFilePath, int outputResolution)**

Create a new version of an image with a different format or with different dimensions.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**



inputImagePath	Full pathname of the input image file.
outputImageFormatAsString	The output image format as string.
outputImageQuality	The output image quality.
outputImageWidth	Width of the output image.
outputImageHeight	Height of the output image.
outputImagePath	Full pathname of the output image file.
outputResolution	The output resolution.

### Returns:

Image info node set.

### Log

urn:WebWorks-XSLT-Extension-Log

### Functions

**void Error (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10])**

Records error(s) to the generation log.

**void Message (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10])**

Records message(s) to the generation log.

**void Warning (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10])**

Records warning(s) to the generation log.

## Detailed Description

urn:WebWorks-XSLT-Extension-Log

Enables XSL transforms to report messages, warnings, and errors to the generation log.

**void Error (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10])**

Records error(s) to the generation log.

## Parameters:

message1	The first message.
message2 – message10	Additional messages (optional).

Express and Designer display a dialog informing the user errors were encountered during processing. AutoMap returns a non-zero return code.

```
<xsl:variable name="VarMessage" select="wwlog:Error('Put this in your log.')" />
```

**void Message (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10])**

Records message(s) to the generation log.

#### **Parameters:**

message1	The first message.
message2 – message10	Additional messages (optional).

**void Warning (string message1 [, string message2, string message3, string message4, string message5, string message6, string message7, string message8, string message9, string message10])**

Records warning(s) to the generation log.

**Parameters:**

message1	The first message.
message2 – message10	Additional messages (optional).

## MultiSearchReplaceExtension

urn:WebWorks-XSLT-Extension-MultiSearchReplace

### Functions

**void ReplaceAllInFile (string inputEncodingAsString, string inputFilePath, string outputFilePath, object replacements)**

Replaces strings in a text file with the specified input encoding and writes the result to the output path with the same encoding.

**void ReplaceAllInFile (string inputEncodingAsString, string inputFilePath, string outputEncodingAsString, string outputFilePath, object replacements)**

Replaces strings in a text file with the specified input encoding and writes the result to the output path with the specified output encoding.

**string ReplaceAllInString (string input, object replacements)**

Replaces strings in the given string and returns the result. Using this method is much faster than performing multiple search/replace calls in XSL substring methods.

### Detailed Description

urn:WebWorks-XSLT-Extension-MultiSearchReplace

Replaces multiple strings in a single operation.

**void ReplaceAllInFile (string inputEncodingAsString, string inputFilePath, string outputFilePath, object replacements)**

Replaces strings in a text file with the specified input encoding and writes the result to the output path with the same encoding.

Note: Not a replacement for using page templates when possible.

## Parameters:

inputEncodingAsString	The input encoding as string.
inputFilePath	Full pathname of the input file.
outputFilePath	Full pathname of the output file.
replacements	The replacements in a node set.

**void ReplaceAllInFile (string inputEncodingAsString, string inputFilePath, string outputEncodingAsString, string outputFilePath, object replacements)**

Replaces strings in a text file with the specified input encoding and writes the result to the output path with the specified output encoding.

Note: Not a replacement for using page templates when possible.

**Parameters:**

inputEncodingAsString	The input encoding as string.
inputFilePath	Full pathname of the input file.
outputEncodingAsString	The output encoding as string.
outputFilePath	Full pathname of the output file.
replacements	The replacements in a node set.

### **string ReplaceAllInString (string input, object replacements)**

Replaces strings in the given string and returns the result. Using this method is much faster than performing multiple search/replace calls in XSL substring methods.

#### **Parameters:**



input	The input.
replacements	The replacements in a node-set.

### Returns:

A string.

## NodeSet

urn:WebWorks-XSLT-Extension-NodeSet

### Functions

#### **XPathNodeIterator FirstUnique (object input, string attributeLocalName)**

First unique element with matching local name.

#### **XPathNodeIterator FirstUniqueWithNamespace (object input, string attributeLocalName, string attributeNamespaceURI)**

First unique element with matching namespace and matching local name.

#### **XPathNodeIterator LastUnique (object input, string attributeLocalName)**

Last unique element with matching local name.

#### **XPathNodeIterator LastUniqueWithNamespace (object input, string attributeLocalName, string attributeNamespaceURI)**

Last unique element with matching namespace and matching local name.

### Detailed Description

urn:WebWorks-XSLT-Extension-NodeSet

Miscellaneous node set functions.

#### **XPathNodeIterator FirstUnique (object input, string attributeLocalName)**

First unique element with matching local name.

## Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

input	The input node set.
attributeLocalName	Local name of the element.

### Returns:

A node set.

**XPathNodeIterator FirstUniqueWithNamespace (object input, string attributeLocalName, string attributeNamespaceURI)**

First unique element with matching namespace and matching local name.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

input	The input node set.
attributeLocalName	Local name of the element.
attributeNamespaceURI	Namespace URI.

**Returns:**

A node set.

**XPathNodeIterator LastUnique (object input, string attributeLocalName)**

Last unique element with matching local name.

**Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

input	The input node set.
attributeLocalName	Local name of the element.

### Returns:

A node set.

**XPathNodeIterator LastUniqueWithNamespace (object input, string attributeLocalName, string attributeNamespaceURI)**

Last unique element with matching namespace and matching local name.

### Exceptions:



OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

input	The input node set.
attributeLocalName	Local name of the element.
attributeNamespaceURI	Namespace URI.

### Returns:

A node set.

## Progress

urn:WebWorks-XSLT-Extension-Progress

### Functions

#### **bool Abort ()**

Checks to see if the user has requested to abort the current operation.

#### **void Cancel ()**

Forces an abort to occur with Cancel as reason.

#### **void End ()**

End the current progress step.

#### **void QueueAlert (string message)**

Queue an alert to display.

#### **void Retry ()**

Forces an abort to occur with Retry as reason.

#### **void SetStatus (string message)**

Set the status bar for the current progress step.

#### **void Start (int totalSubSteps)**

Create a new progress step with the given number of sub-steps.

### Detailed Description

urn:WebWorks-XSLT-Extension-Progress

Reports progress during long lived XSL transforms.

### **bool Abort ()**

Checks to see if the user has requested to abort the current operation.

#### **Returns:**

True if abort requested, false otherwise.

### **void Cancel ()**

Forces an abort to occur with Cancel as reason.

### **void End ()**

End the current progress step.

### **void QueueAlert (string message)**

Queue an alert to display.

#### **Parameters:**

message	Message to display.
---------	---------------------

### **void Retry ()**

Forces an abort to occur with Retry as reason.

### **void SetStatus (string message)**

Set the status bar for the current progress step.

### **Parameters:**

message	Message to display.
---------	---------------------

```

<xsl:variable name="VarParagraphCount"
  select="count($VarParagraphs)" />

<xsl:variable name = "VarProgressParagraphsStart"
  select="wwprogress:Start($VarParagraphCount)" />

<xsl:variable name = "VarProgressParagraphsStatus"
  select="wwprogress:SetStatus(concat('Processing ',
    $VarParagraphCount))" />

<xsl:for-each select = "$VarParagraphs" >

  < xsl:variable name = "VarParagraph" select="." />

  <xsl:variable name = "VarProgressParagraphStart"
    select="wwprogress:Start(1)" />

  <xsl:variable name = "VarProgressParagraphStatus"
    select="wwprogress:SetStatus('Processing paragraph ', position(), '
    of ', $VarParagraphCount, '.'))" />

  <xsl:variable name = "VarProgressParagraphEnd"
    select="wwprogress:End()" />

</xsl:for-each>

<xsl:variable name = "VarProgressParagraphsEnd"
  select="wwprogress:End()" />

```

### **void Start (int totalSubSteps)**

Create a new progress step with the given number of sub-steps.

#### **Parameters:**

totalSubSteps	The total sub steps.
---------------	----------------------

```

<xsl:variable name="VarProgressTotalStart"
  select="wwprogress:Start(2)" />

<xsl:variable name = "VarProgressStep1Start"
  select="wwprogress:Start(1)" />

<xsl:variable name = "VarProgressStep1End" select="wwprogress:End()" /
>

<xsl:variable name = "VarProgressStep2Start"
  select="wwprogress:Start(1)" />

<xsl:variable name = "VarProgressStep2End" select="wwprogress:End()" /
>

<xsl:variable name = "VarProgressTotalEnd" select="wwprogress:End()" /
>

```

## Project

urn:WebWorks-XSLT-Extension-Project

### Functions

#### **bool DocumentExtension (string extension)**

Determine if filename 'extension' has a configured input adapter.

#### **bool GetConditionIsPassThrough (string conditionName)**

Determines if condition called: 'conditionName' is pass through.

#### **string GetConfigurationChangeID ()**

Gets configuration change identifier.

#### **XPathNodeIterator GetContextRule (string ruleTypeAsString, string ruleName, string documentID, string uniqueID)**

Gets context rule.

**string GetDocumentDataDirectoryPath (string documentID)**

Gets document data directory path using 'documentID'.

**string GetDocumentGroupPath (string documentID)**

Gets document group path using 'documentID'.

**string GetDocumentID (string documentPath [, string groupID])**

Gets document identifier using 'documentPath' and optionally 'groupID'.

**string GetDocumentPath (string documentID)**

Gets document path using 'documentID'.

**string GetDocumentsToGenerateChecksum ()**

Get string representing all documents in project so checksum can be generated.

**string GetFormatID ()**

Gets format identifier.

**string GetFormatName ()**

Gets format name.

**string GetFormatSetting (string name)**

Gets 'name' format setting.

**string GetFormatSetting (string name, string defaultValue)**

Gets 'name' format setting but returns 'defaultValue' if no setting configured.

**string GetGroupDataDirectoryPath (string groupID)**

Gets group data directory path using 'groupID'.

**string GetGroupName (string groupID)**

Gets group name using 'groupID'.

**XPathNodeIterator GetOverrideRule (string ruleTypeAsString, string ruleName, string documentID, string uniqueID)**

Gets override rule.

**string GetProjectDataDirectoryPath ()**

Gets project data directory path.

**string GetProjectDirectoryPath ()**

Gets project directory path.

**long GetProjectDocumentsCount ()**

Gets project documents count.

**string GetProjectFilesDirectoryPath ()**

Gets project files directory path.

**string GetProjectFormatDirectoryPath ()**

Gets project format directory path.

**string GetProjectName ()**

Gets project name.

**string GetProjectReportsDirectoryPath ()**

Gets project reports directory path.

**string GetProjectTargetName ()**

Gets project target name currently being processed.

**string GetProjectTargetOverrideDirectoryPath ()**

Gets project target override directory path.

**XPathNodeIterator GetRule (string ruleTypeAsString, string ruleName)**

Gets a rule.

**string GetTargetDataDirectoryPath ()**

Gets target data directory path.

**string GetTargetFilesInfoPath (string targetIDAsString)**

Gets target files information path.

**string GetTargetOutputDirectoryPath ()**



Gets target output directory path.

### **string GetTargetReportsDirectoryPath ()**

Gets target reports directory path.

### **Detailed Description**

urn:WebWorks-XSLT-Extension-Project

Query for information about the currently running project.

### **bool DocumentExtension (string extension)**

Determine if filename 'extension' has a configured input adapter.

### **Parameters:**

extension	The extension.
-----------	----------------

**Returns:**

True if extension has a configured adapter, false otherwise.

**bool GetConditionIsPassThrough (string conditionName)**

Determines if condition called: 'conditionName' is pass through.

**Parameters:**

conditionName	Name of the condition.
---------------	------------------------

**Returns:**

True if pass through, false otherwise.

**string GetConfigurationChangeID ()**

Gets configuration change identifier.

**Returns:**

The configuration change identifier.

**XPathNodeIterator GetContextRule (string ruleTypeAsString, string ruleName, string documentID, string uniqueID)**

Gets context rule.

**Parameters:**

ruleTypeAsString	The rule type as string.
ruleName	Name of the rule.
documentID	Identifier for the document.
uniqueID	Unique identifier.

### **Returns:**

The context rule as a node set.

### **string GetDocumentDataDirectoryPath (string documentID)**

Gets document data directory path using 'documentID'.

### **Parameters:**

documentID	Identifier for the document.
------------	------------------------------

**Returns:**

The document data directory path.

**string GetDocumentGroupPath (string documentID)**

Gets document group path using 'documentID'.

**Parameters:**

documentID	Identifier for the document.
------------	------------------------------

### **Returns:**

The document group path.

**string GetDocumentID (string documentPath [, string groupID])**

Gets document identifier using 'documentPath' and optionally 'groupID'.

### **Parameters:**

documentPath	Full pathname of the document file.
groupID	Identifier for the group (optional).

**Returns:**

The document identifier as string.

**string GetDocumentPath (string documentID)**

Gets document path using 'documentID'.

**Parameters:**

documentID	Identifier for the document.
------------	------------------------------

**Returns:**

The document path.

**string GetDocumentsToGenerateChecksum ()**

Get string representing all documents in project so checksum can be generated.

**Returns:**

A string representing documents for generating checksum.

**string GetFormatID ()**

Gets format identifier.

**Returns:**

The format identifier as string.

**string GetFormatName ()**

Gets format name.

**Returns:**

The format name as string.

**string GetFormatSetting (string name)**

Gets 'name' format setting.

**Parameters:**



name	The name.
------	-----------

**Returns:**

The format setting as string.

**string GetFormatSetting (string name, string defaultValue)**

Gets 'name' format setting but returns 'defaultValue' if no setting configured.

**Parameters:**

name	The name.
defaultValue	The default value.

### **Returns:**

The format setting as string.

### **string GetGroupDataDirectoryPath (string groupID)**

Gets group data directory path using 'groupID'.

### **Parameters:**

groupID	Identifier for the group.
---------	---------------------------

**Returns:**

The group data directory path.

**string GetGroupName (string groupID)**

Gets group name using 'groupID'.

**Parameters:**

groupID	Identifier for the group.
---------	---------------------------

**Returns:**

The group name as string.

**XPathNodeIterator GetOverrideRule (string ruleTypeAsString, string ruleName, string documentID, string uniqueID)**

Gets override rule.

**Parameters:**

ruleTypeAsString	The rule type as string.
ruleName	Name of the rule.
documentID	Identifier for the document.
uniqueID	Unique identifier.

### **Returns:**

The override rule node set.

### **string GetProjectDataDirectoryPath ()**

Gets project data directory path.

### **Returns:**

The project data directory path.

### **string GetProjectDirectoryPath ()**

Gets project directory path.

### **Returns:**

The project directory path.

### **long GetProjectDocumentsCount ()**

Gets project documents count.

### **Returns:**

The project documents count.

### **string GetProjectFilesDirectoryPath ()**

Gets project files directory path.

**Returns:**

The project files directory path.

**string GetProjectFormatDirectoryPath ()**

Gets project format directory path.

**Returns:**

The project format directory path.

**string GetProjectName ()**

Gets project name.

**Returns:**

The project name.

**string GetProjectReportsDirectoryPath ()**

Gets project reports directory path.

**Returns:**

The project reports directory path.

**string GetProjectTargetName ()**

Gets project target name currently being processed.

**Returns:**

The project target name.

**string GetProjectTargetOverrideDirectoryPath ()**

Gets project target override directory path.

**Returns:**

The project target override directory path.

**XPathNodeIterator GetRule (string ruleTypeAsString, string ruleName)**

Gets a rule.

**Parameters:**

ruleTypeAsString	The rule type as string.
ruleName	Name of the rule.

**Returns:**

The rule as a node set.

**string GetTargetDataDirectoryPath ()**

Gets target data directory path.

**Returns:**

The target data directory path.

**string GetTargetFilesInfoPath (string targetIDAsString)**

Gets target files information path.

**Parameters:**



targetIDAsString	Target identifier as string.
------------------	------------------------------

### **Returns:**

The target files information path.

### **string GetTargetOutputDirectoryPath ()**

Gets target output directory path.

### **Returns:**

The target output directory path.

### **string GetTargetReportsDirectoryPath ()**

Gets target reports directory path.

### **Returns:**

The target reports directory path.

## **StageInfo**

urn:WebWorks-XSLT-Extension-StageInfo

### **Functions**

### **string Get (string param\_key)**

Gets the value of a given key for this stage.

### **void Set (string param\_key, string param\_value)**

Set key/value pair for this stage.

### **Detailed Description**

urn:WebWorks-XSLT-Extension-StageInfo

Allows XLST processing to store and retrieve key/value pairs as needed to track state.

**string Get (string param\_key)**

Gets the value of a given key for this stage.

**Parameters:**

param_key	The key to lookup.
-----------	--------------------

**Returns:**

The value as string.

**void Set (string param\_key, string param\_value)**

Set key/value pair for this stage.

**Parameters:**

param_key	The key to set.
param_value	The value to set.

## StringUtilities

urn:WebWorks-XSLT-Extension-StringUtilities

### Functions

#### **string CSSClassName (string styleName)**

Convert the given string into a valid CSS class name.

#### **string DecodeURI (string value)**

Decode an escaped URI 'value' back to an unescaped URI.

#### **string DecodeURIComponent (string value)**

Decode an escaped partial URI component, 'value', back to an unescaped URI component.

#### **string EclipseId (string identifier)**

Create a valid Eclipse ID from 'identifier'.

#### **string EncodeURI (string value)**

Encode 'value' string as an escaped URI.

#### **string EncodeURIComponent (string value)**

Encode 'value' string, a partial URI component, as an escaped URI.

#### **string EscapeForXMLAttribute (string value)**

Escape 'value' string so that it can be written as an XML attribute.

#### **string Format (string format, string argument1 [, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10])**

Format a message using the C# string formatter.

**string FromFile (string sourceFilePath, string sourceFileEncodingName)**

Import file contents using 'sourceFileEncodingName' as the file's assumed encoding.

**string JavaScriptEncoding (string value)**

Convert all non-ASCII characters to Unicode escape sequences. Also convert all ASCII characters less than 32 along with problematic escape characters, i.e. \, to Unicode escape sequences.

**bool MatchExpression (string input, string matchExpressionAsString)**

Return success of match for 'matchExpressionAsString' in 'input'.

**string MatchExpressionValue (string input, string matchExpressionAsString)**

Return value of match for 'matchExpressionAsString' in 'input'.

**string MD5Checksum (string value)**

Compute the MD5 checksum on the given 'value' string.

**string NCNAME (string identifier)**

Convert 'identifier' to a valid NCNAME as defined by:

**string NormalizeQuotes (string value)**

Convert the given string to a string where all left/right single/double quotes are normalized. Left single quotation mark = single quotation mark Right single quotation mark = single quotation mark Left double quotation mark = double quotation mark Right double quotation mark = double quotation mark

**string OEBClassName (string styleName)**

Create a valid Open eBook class name from 'styleName'.

**string Replace (string input, string search, string replacement)**

Replace all occurrences of 'search' in 'input' with 'replacement'.

**string ReplaceWithExpression (string input, string searchExpressionAsString, string replacement)**

Replace all occurrences of 'searchExpressionAsString' in 'input' with 'replacement'.

**string ReplaceWithExpressionForCount (string input, string searchExpressionAsString, string replacement, int count)**

Replace 'count' occurrences of 'searchExpressionAsString' in 'input' with 'replacement'.

**string SHA1Checksum (string value)**

Compute the SHA-1 (Secure Hash Algorithm 1) checksum on the given 'value' string.

**string ToLower (string value)**

Convert the given string to lowercase.

**string ToUpper (string value)**

Convert the given string to uppercase.

**string ToCamel (string value)**

Convert the given string to camel case.

**string ToPascal (string value)**

Convert the given string to pascal case.

**bool EndsWith (string input, string suffix)**

Return success of suffix being the suffix of input.

**string WebWorksHelpContextOrTopic (string key)**

Converts the given 'key' into a valid WebWorks Help/Reverb context or topic string.

**Detailed Description**

urn:WebWorks-XSLT-Extension-StringUtilities

Extend the available string processing methods to XSL to include message formatting, specialized text escaping, regular expression operations, etc.

**string CSSClassName (string styleName)**

Convert the given string into a valid CSS class name.

**Parameters:**

styleName	Name of the style.
-----------	--------------------

### Returns:

A string.

Convert Blue\_Moon.Detective;Agency into a valid CSS style name.

```
<xsl:value-of
  select="wwstring:CSSClassName('Blue_Moon.Detective;Agency')" />
```

### string DecodeURI (string value)

Decode an escaped URI 'value' back to an unescaped URI.

### Parameters:

value	The value.
-------	------------

**Returns:**

A string.

**string DecodeURIComponent (string value)**

Decode an escaped partial URI component, 'value', back to an unescaped URI component.

**Parameters:**



value	The value.
-------	------------

**Returns:**

A string.

**string EclipseId (string identifier)**

Create a valid Eclipse ID from 'identifier'.

**Parameters:**

identifier	The identifier.
------------	-----------------

**Returns:**

A string.

**string EncodeURI (string value)**

Encode 'value' string as an escaped URI.

**Parameters:**

value	The value.
-------	------------

**Returns:**

A string.

**string EncodeURIComponent (string value)**

Encode 'value' string, a partial URI component, as an escaped URI.

**Parameters:**

value	The value.
-------	------------

**Returns:**

A string.

**string EscapeForXMLAttribute (string value)**

Escape 'value' string so that it can be written as an XML attribute.

**Parameters:**

value	The value.
-------	------------

### Returns:

A string.

Write onClick handler for <div> tag.

```
<html:div onClick="{wwstring:EscapeForXmlAttribute('alert(\'Boo!\n\');')}">

    Click me!

</html:div>
```

**string Format (string format, string argument1 [, string argument2, string argument3, string argument4, string argument5, string argument6, string argument7, string argument8, string argument9, string argument10])**

Format a message using the C# string formatter.

### Parameters:

format	Describes the format to use.
argument1	The first argument.
argument2 – argument10	Additional arguments (optional).

### Returns:

The formatted value.

Create the message: '17 total'.

```
<xsl:value-of select="wwstring:Format('{0} total', 17)" />
```

Create the message: '17 total, 15 of 17'.

```
<xsl:value-of select="wwstring:Format('{0} total, {1} of {0}.', 17, 15)" />
```

### **string FromFile (string sourceFilePath, string sourceFileEncodingName)**

Import file contents using 'sourceFileEncodingName' as the file's assumed encoding.

### Parameters:

sourceFilePath	Full pathname of the source file.
sourceFileEncodingName	Name of the source file encoding.

### Returns:

A string.

Import the contents of file: 'C:.txt' which was encoded using UTF-8.

```
<xsl:value-of select="wwstring:FromFile('C:\myfile.txt', 'UTF-8')" />
```

### string JavaScriptEncoding (string value)

Convert all non-ASCII characters to Unicode escape sequences. Also convert all ASCII characters less than 32 along with problematic escape characters, i.e. \, to Unicode escape sequences.

### Parameters:

value	The value.
-------	------------

### Returns:

A string.

Convert Hello!

to JavaScript encoded text.

```
<xsl:value-of select="wwstring:JavaScriptEncoding('Hello\nworld!\n') " />
```

### **bool MatchExpression (string input, string matchExpressionAsString)**

Return success of match for 'matchExpressionAsString' in 'input'.

### Exceptions:



OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

input	The input.
matchExpressionAsString	The match expression as string.

### Returns:

True if match found, false otherwise.

Contains 3-4 "a"s?

```
<xsl:value-of select="wwstring:Replace('<letter>scar's <letter>nly
<letter>strich <letter>iled an <letter>range <letter>wl
t<letter>day.', '<letter>', 'o')" />
```

**string MatchExpressionValue (string input, string matchExpressionAsString)**

Return value of match for 'matchExpressionAsString' in 'input'.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

input	The input.
matchExpressionAsString	The match expression as string.

### Returns:

String.

Value of 3-4 "a"s.

```
<xsl:value-of select="wwstring:MatchExpressionValue('The end of aa
sentenceaaaaalways leaves me sad.', 'a{3-4}')" />
```

### string MD5Checksum (string value)

Compute the MD5 checksum on the given 'value' string.

### Parameters:

value	The value.
-------	------------

### Returns:

A string.

Determine the MD5 signature for A long time ago, way back in history, when all there was to drink, was nothing but cups 'o tea..

```
<xsl:value-of select="wwstring:MD5Checksum('A long time ago, way back
in history, when all there was to drink, was nothing but cups \'o
tea.\')" />
```

### string NCNAME (string identifier)

Convert 'identifier' to a valid NCNAME as defined by:

<http://www.w3.org/TR/1999/REC-xml-names-19990114/#NT-NCNameChar>.

### Parameters:

identifier	The identifier.
------------	-----------------

### Returns:

A string.

Create an NCName from 'id' attribute.

```
<xsl:value-of select="wwstring:NCNAME($VarTable/@id)" />
```

### string NormalizeQuotes (string value)

Convert the given string to a string where all left/right single/double quotes are normalized. Left single quotation mark = single quotation mark Right single quotation mark = single quotation mark Left double quotation mark = double quotation mark Right double quotation mark = double quotation mark

### Parameters:

value	The value.
-------	------------

### Returns:

String.

```
<xsl:value-of select="wwstring:NormalizeQuotes('string-with-left-right-quote-marks...')"/>

```

### string OEBClassName (string styleName)

Create a valid Open eBook class name from 'styleName'.

### Parameters:

styleName	Name of the style.
-----------	--------------------

**Returns:**

A string.

**string Replace (string input, string search, string replacement)**

Replace all occurrences of 'search' in 'input' with 'replacement'.

**Parameters:**



input	The input.
search	The search.
replacement	The replacement.

### Returns:

Result String.

Replace all instances of <letter> with O.

```
<xsl:value-of select="wwstring:Replace('<letter>scar's <letter>nly
<letter>strich <letter>iled an <letter>range <letter>wl
t<letter>day.', '<letter>', 'o')"/> />
```

**string ReplaceWithExpression (string input, string searchExpressionAsString, string replacement)**

Replace all occurrences of 'searchExpressionAsString' in 'input' with 'replacement'.

### Parameters:

input	The input.
searchExpressionAsString	The search expression as string.
replacement	The replacement.

### Returns:

String.

Replace runs of 3-4 "a"s with ". ".

```
<xsl:value-of select="wwstring:ReplaceWithExpression('The end of aa
sentenceaaaaalways leaves me sad.', 'a{3-4}', '. ')" />
```

**string ReplaceWithExpressionForCount (string input, string searchExpressionAsString, string replacement, int count)**

Replace 'count' occurrences of 'searchExpressionAsString' in 'input' with 'replacement'.

### Parameters:

input	The input.
searchExpressionAsString	The search expression as string.
replacement	The replacement.
count	Number of.

### Returns:

A string.

Replace first run of 3-4 "a"s with ". ".

```
<xsl:value-of select="wwstring:ReplaceWithExpression('The end of aa
sentenceaaaaalways leaves me sad.', 'a{3-4}', '. ', 1)" />
```

### string SHA1Checksum (string value)

Compute the SHA-1 (Secure Hash Algorithm 1) checksum on the given 'value' string.

### Parameters:

value	The value.
-------	------------

### Returns:

A string.

Determine the SHA1 signature for A long time ago, way back in history, when all there was to drink, was nothing but cups 'o tea..

```
<xsl:value-of select="wwstring:SHA1Checksum('A long time ago, way back
in history, when all there was to drink, was nothing but cups \'o
tea.\')" />
```

### string ToLower (string value)

Convert the given string to lowercase.

### Parameters:

value	The value.
-------	------------

### Returns:

String.

```
<xsl:value-of select="wwstring:ToLower('UpPERcAse')" />
```

### string ToUpper (string value)

Convert the given string to uppercase.

### Parameters:

value	The value.
-------	------------

### Returns:

String.

```
<xsl:value-of select="wwstring:ToUpper('lOwErCAsE')" />
```

### string ToCamel (string value)

Convert the given string to camel case.

### Parameters:

value	The value.
-------	------------

### Returns:

String.

```
<xsl:value-of select="wwstring:ToCamel('camel case')" />
```

### string ToPascal (string value)

Convert the given string to pascal case.

### Parameters:

value	The value.
-------	------------

**Returns:**

String.

```
<xsl:value-of select="wwstring:ToPascal('pascal case')" />
```

**bool EndsWith (string input, string suffix)**

Return success of suffix being the suffix of input.

**Parameters:**



input	The input.
suffix	The suffix.

### Returns:

Bool.

Check if ".scss" is suffix of "webworks.scss". This returns True.

```
<xsl:value-of select="wwstring:EndsWith('webworks.scss', '.scss')" />
```

### string WebWorksHelpContextOrTopic (string key)

Converts the given 'key' into a valid WebWorks Help/Reverb context or topic string.

Note: WebWorks Help/Reverb context and topic strings may only contain the characters A-Z, a-z, 0-9, and \_.

### Parameters:

key	The key.
-----	----------

### Returns:

A string.

Convert A long time... into a valid WebWorks Help context or topic name.

```
<xsl:value-of select="wwstring:WebWorksHelpContextOrTopic('A long time...')"/>

```

## Units

urn:WebWorks-XSLT-Extension-Units

### Functions

#### **double Convert (double sourceValue, string sourceUnits, string targetUnits)**

Convert a measurement from one set of units to another.

#### **string CSSRGBColor (string htmlColor)**

Convert the given HTML/CSS color to a hex encoded CSS color. Useful for converting named colors such as green to their hex equivalents.

#### **string EncodingFromCodePage (int codePage)**

Determine the HTML encoding for a page given a Windows code page value.

#### **string NumericPrefix (string value)**

Extract the numeric prefix of 'value'.

#### **string RTFColor (string htmlColor)**

Convert a standard HTML/CSS color to an RTF color.

#### **string UnitsSuffix (string value)**

Extract the units suffix of 'value'. Only returns a non-zero length string if there is also a numeric prefix.

### Detailed Description

urn:WebWorks-XSLT-Extension-Units

Utility methods for extracting units and value from raw strings along with unit-to-unit conversion routines.

**double Convert (double sourceValue, string sourceUnits, string targetUnits)**

Convert a measurement from one set of units to another.

**Parameters:**

sourceValue	Source value.
sourceUnits	Source units.
targetUnits	Target units.

### Returns:

A number.

Convert 2in to centimeters (cm).

```
<xsl:variable name="VarCentimeters"
  select="wwunits:Convert(wwunits:NumericPrefix('2in'),
    wwunits:UnitsSuffix('2in'), 'cm')"/>

```

### string CSSRGBColor (string htmlColor)

Convert the given HTML/CSS color to a hex encoded CSS color. Useful for converting named colors such as green to their hex equivalents.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

htmlColor	The HTML color.
-----------	-----------------

### Returns:

Hex encoded CSS color as string.

Convert green to the hex equivalent.

```
<xsl:variable name="VarGreenAsRGB"
  select="wwunits:CSSRGBColor('green')"/>

```

### string EncodingFromCodePage (int codePage)

Determine the HTML encoding for a page given a Windows code page value.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

codePage	The code page.
----------	----------------

### Returns:

A string.

Determine the HTML encoding for code page 23.

```
<xsl:variable name="VarEncoding"
  select="wwunits:EncodingFromCodePage(23)" />
```

### string NumericPrefix (string value)

Extract the numeric prefix of 'value'.

### Parameters:



value	The value.
-------	------------

### Returns:

A string.

Determine numeric prefix of 24px.

```
<xsl:variable name="VarNumber"
  select="wwunits:NumericPrefix('24px') " />
```

### string RTFColor (string htmlColor)

Convert a standard HTML/CSS color to an RTF color.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

htmlColor	The HTML color.
-----------	-----------------

### Returns:

RTF color as string.

Convert #FF3399 to an RTF color.

```
<xsl:variable name="VarRTFColor"
  select="wwunits:RTFColor('#FF3399')" />
```

### string UnitsSuffix (string value)

Extract the units suffix of 'value'. Only returns a non-zero length string if there is also a numeric prefix.

### Parameters:

value	The value.
-------	------------

### Returns:

A string.

Determine units of 24px.

```
<xsl:variable name="VarNumber" select="wwunits:UnitsSuffix('24px')" />
```

## URI

urn:WebWorks-XSLT-Extension-URI

### Functions

#### **string AsFilePath (string uriAsString)**

Converts an uriAsString to a file path.

#### **string AsURI (string filePathAsString)**

Converts a file path to an uriAsString.

#### **string EscapeData (string unescapedString)**

Convert unescaped string into an escaped URI data.

#### **string EscapeUri (string unescapedUri)**

Convert unescaped URI into an escaped URI path.

#### **string GetRelativeTo (string uriAsString, string anchorUriAsString)**

Convert an absolute URI into a relative URI.

#### **bool IsFile (string uriAsString)**

Determine if the supplied URI refers to the file system.

#### **string MakeAbsolute (string absoluteUriAsString, string uriAsString)**

Convert a relative URI into an absolute URI. If the second parameter is already an absolute URI, it will be returned unchanged.

### **XPathNodeIterator PossibleResolvedUris (string uriAsString)**

Convert file paths to URIs.

### **string Unescape (string escapedString)**

Convert URI escaped string into unescaped string (%20 to space, etc.).

### **Detailed Description**

urn:WebWorks-XSLT-Extension-URI

Utility methods which convert to and from file paths and create absolute or relative URIs.

### **string AsFilePath (string uriAsString)**

Converts an uriAsString to a file path.

### **Exceptions:**

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

uriAsString	The URI as string.
-------------	--------------------

### Returns:

File path as string.

Convert URIs (i.e. file:///network/filesystem/file) to file paths.

```
<xsl:variable name="VarFilePath1" select="wwuri:AsFilePath('\\\\
\network\\filesystem\\file')">
```

```
<xsl:variable name = "VarFilePath2" select="wwuri:AsFilePath('file:///
network/filesystem/file')">
```

### string AsURI (string filePathAsString)

Converts a filePathAsString to a URI.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**



filePathAsString	The file path as string.
------------------	--------------------------

### Returns:

URI as string.

Convert file paths (i.e. C:\file) to URI strings (i.e. file:///C:/file).

```
<xsl:variable name="VarURI" select="wwuri:AsURI('C:\\file') ">
```

### string EscapeData (string unescapedString)

Convert unescaped string into an escaped URI data.

### Parameters:

unescapeString	The unescaped string.
----------------	-----------------------

### Returns:

Escaped string.

Convert `http://www.webworks.com/name with spaces.html` to `http%3a%2f%2fwww.webworks.com%2fname%20with%20spaces.html`.

```
<xsl:variable name="VarEscapedData" select="wwuri:EscapeData('http://www.webworks.com/name with spaces.html')"/>
```

### **string EscapeUri (string unescapedUri)**

Convert unescaped URI into an escaped URI path.

### Parameters:

unescapeUri	URI of the unescaped.
-------------	-----------------------

### Returns:

Escaped URI as a string.

Convert `http://www.webworks.com/name with spaces.html` to `http://www.webworks.com/name%20with%20spaces.html`.

```
<xsl:variable name="VarEscapedURI" select="wwuri:EscapeUri('http://www.webworks.com/name with spaces.html')"/>
```

### **string GetRelativeTo (string uriAsString, string anchorUriAsString)**

Convert an absolute URI into a relative URI.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

uriAsString	The URI as string.
anchorUriAsString	The anchor URI as string.

### Returns:

The relative to.

Determine the relative path to <http://www.webworks.com/css/webworks.css> from <http://www.webworks.com/information/index.html>.

```
<xsl:variable name="VarRelativeURI"
  select="wwuri:GetRelativeTo('http://www.webworks.com/css/
webworks.css', 'http://www.webworks.com/information/index.html') " />
```

### bool IsFile (string uriAsString)

Determine if the supplied URI refers to the file system.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

uriAsString	The URI as string.
-------------	--------------------

### Returns:

True if file, false if not.

Determine if \network and file:///network/filesystem/file are file URIs.

```
<xsl:if test="wwuri:IsFile('\\\\network\\filesystem\\file')">
    <xsl:variable name = "VarLog" select="wwlog:Message('File!')" />
</xsl:if>

<xsl:if test="wwuri:IsFile('file:///network/filesystem/file')">
    <xsl:variable name = "VarLog" select="wwlog:Message('File!')" />
</xsl:if>
```

### string MakeAbsolute (string absoluteUriAsString, string uriAsString)

Convert a relative URI into an absolute URI. If the second parameter is already an absolute URI, it will be returned unchanged.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**



absoluteUriAsString	The absolute URI as string.
uriAsString	The URI as string.

### Returns:

URI as string.

Fully qualify the relative URI ../css/webworks.css against the absolute URI http://www.webworks.com/information/index.html.

```
<xsl:variable name="VarCSSURI" select="wwuri:MakeAbsolute('http://www.webworks.com/information/index.html', '../css/webworks.css')"/>
```

### XPathNodeIterator PossibleResolvedUris (string uriAsString)

Convert file paths to URIs.

### Exceptions:

OutOfMemoryException	Thrown when a low memory situation occurs.
----------------------	--

**Parameters:**

uriAsString	The URI as string.
-------------	--------------------

### Returns:

A node set of possible resolved URIs for the given virtual URI.

```
<wwuri:Uris>

  <wwuri:Uri value = "file:///C:/Users/user/Projects/Targets/Reverb/
Transforms/pages.xml" />

  < wwuri:Uri value = "file:///C:/Users/user/Projects/Formats/WebWorks
Reverb/Transforms/pages.xml" />

  < wwuri:Uri value = "file:///C:/Program Files (x86)/WebWorks/
ePublisher/2012.4/Formats/WebWorks Reverb/Transforms/pages.xml" />

</ wwuri:Uris>
```

Resolve virtual URI wwformat:Transforms/pages.xml.

```
<xsl:variable name="VarPossibleURIs"
  select="wwuri:PossibleResolvedUris('wwformat:Transforms/
pages.xml')"/>

<xsl:for-each select = "$VarPossibleURIs/wwuri:Uris/wwuri:Uri" >

  <xsl:variable name = "VarPossibleURI" select="." />

  <xsl:if test="wwuri:IsFile($VarPossibleURI)">

    <xsl:variable name = "VarPossibleFilePath"
    select="wwuri:AsFilePath($VarPossibleURI)" />

    <xsl:if test="wwfilesystem:FileExists($VarPossibleFilePath)">

      ...

    </xsl:if>

  </xsl:if>

</xsl:for-each>
```

**string Unescape (string escapedString)**

Convert URI escaped string into unescaped string (%20 to space, etc.).

**Parameters:**

escapedString	The escaped string.
---------------	---------------------

### Returns:

Unescaped string.

Convert name%20with%20spaces.html to name with spaces.html.

```
<xsl:variable name="VarUnescape" select="wwuri:Unescape('name%20with%20spaces.html') " />
```

## ZipExtension

urn:WebWorks-XSLT-Extension-Zip

### Functions

#### **void Zip (string zipFilePath, object nodes)**

Create a zip archive containing a list of files.

#### **void ZipAdd (string zipFilePath, object nodes)**

Add a list of files to a zip archive.

#### **void ZipAddWithoutCompression (string zipFilePath, object nodes)**

Add a list of files to a zip archive without compressing any files.

#### **void ZipDirectory (string zipFilePath, string directoryPath)**

Zip a directory and recursively include sub-directories without compressing any files.

#### **void ZipDirectoryWithoutCompression (string zipFilePath, string directoryPath)**

Zip a directory and recursively include sub-directories without compressing any files.

#### **void ZipExtract (string zipFilePath, string targetDirectory)**

Extract contents of a zip archive to a specified directory location.

#### **void ZipWithoutCompression (string zipFilePath, object nodes)**

Create a zip archive containing a list of files without compressing any files.

### **Detailed Description**

urn:WebWorks-XSLT-Extension-Zip

Allow XSL transforms to handle zip archives.

**void Zip (string zipFilePath, object nodes)**

Create a zip archive containing a list of files.

**Parameters:**

zipFilePath	Full pathname of the zip file.
nodes	The node set.

Creates a zip archive C:.zip with the contents defined in the node set.

```
<xsl:variable name="VarZipListAsXML">

  <wwzip:File source = "C:\some\directory\alpha.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\beta.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\gamma.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\last\gamma.txt" zip-
directory="directory\last" />

</xsl:variable>

<xsl:variable name = "VarZipList" select="wwzip:Zip('C:\some
\archive.zip', msxsl:node-set($VarZipListAsXML)/*" />
```

**void ZipAdd (string zipFilePath, object nodes)**

Add a list of files to a zip archive.

**Parameters:**

zipFilePath	Full pathname of the zip file.
nodes	The node set.

Append to zip archive C:.zip with the contents defined in the node set.

```
<xsl:variable name="VarZipListAsXML">

  <wwzip:File source = "C:\some\directory\alpha.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\beta.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\gamma.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\last\gamma.txt" zip-
directory="directory\last" />

</xsl:variable>

<xsl:variable name = "VarZipList" select="wwzip:ZipAdd('C:\some
\archive.zip', msxsl:node-set($VarZipListAsXML)/*" />
```

**void ZipAddWithoutCompression (string zipFilePath, object nodes)**

Add a list of files to a zip archive without compressing any files.

**Parameters:**



zipFilePath	Full pathname of the zip file.
nodes	The node set.

Append to zip archive C:.zip with the contents defined in the node set.

```
<xsl:variable name="VarZipListAsXML">

  <wwzip:File source = "C:\some\directory\alpha.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\beta.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\gamma.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\last\gamma.txt" zip-
directory="directory\last" />

</xsl:variable>

<xsl:variable name = "VarZipList"
  select="wwzip:ZipAddWithoutCompression('C:\some\archive.zip',
  msxsl:node-set($VarZipListAsXML)/*" />
```

**void ZipDirectory (string zipFilePath, string directoryPath)**

Zip a directory and recursively include sub-directories without compressing any files.

**Parameters:**

zipFilePath	Full pathname of the zip file.
directoryPath	Full pathname of the directory.

Creates zip archive C:.zip with the directory and and subdirectories of 'C:\ projects'.

```
<xsl:variable name="VarZipDirectory" select="wwzip:ZipDirectory('C:\some\archive.zip', 'C:\projects\myproject'" />
```

**void ZipDirectoryWithoutCompression (string zipFilePath, string directoryPath)**

Zip a directory and recursively include sub-directories without compressing any files.

**Parameters:**

zipFilePath	Full pathname of the zip file.
directoryPath	Full pathname of the directory file.

Creates zip archive C:.zip with the directory and and subdirectories of C:\ projects.

```
<xsl:variable name="VarZipDirectory"
  select="wwzip:ZipDirectoryWithoutCompression('C:\some\archive.zip',
  'C:\projects\myproject'" />
```

**void ZipExtract (string zipFilePath, string targetDirectory)**

Extract contents of a zip archive to a specified directory location.

**Parameters:**

zipFilePath	Full pathname of the zip file.
targetDirectory	Pathname of the target directory.

Extracts the contents of zip archive C:.zip to C:.

```
<xsl:variable name="VarZipExtract" select="wwzip:ZipExtract('C:\some
\archive.zip', 'C:\projects\myproject'" />
```

**void ZipWithoutCompression (string zipFilePath, object nodes)**

Create a zip archive containing a list of files without compressing any files.

**Parameters:**

zipFilePath	Full pathname of the zip file.
nodes	The nodes.

Creates a zip archive C:.zip with the contents defined in the node set.

```
<xsl:variable name="VarZipListAsXML">

  <wwzip:File source = "C:\some\directory\alpha.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\beta.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\gamma.txt" zip-
directory="directory" />

  <wwzip:File source = "C:\some\directory\last\gamma.txt" zip-
directory="directory\last" />

</xsl:variable>

<xsl:variable name = "VarZipList"
  select="wwzip:ZipWithoutCompression('C:\some\archive.zip',
  msxsl:node-set($VarZipListAsXML)/*" />
```

# Frequently Asked Questions

What changed in Font Awesome version 5.15.4 from 4.7.x that impacts Reverb 2.0?

How do I disable automatic Preview generation in Designer?

How do I capture source file meta data in the published HTML output?

How do I use a Project Variable's value in the published output?

How do I upgrade an existing WebWorks Reverb 2.0 project to a newer version?

How to add an image icon and stylize your DITA Hazard statement elements?

How to supplement or replace Reverb Toolbar Group Tabs

Invalid File Path Characters Filtered from Generated File Names

How to enable 'WebWorks Menu' (Transit) Add-in for Microsoft Word

How do I set image horizontal alignment in ePublisher?

Why is the PDF format skipping Markdown files?

How can a markdown file include another?

How can you insert an Index Marker into a Markdown file?

How to use alternate DITA-OT installation with ePublisher

How do I embed HTML within DITA source content?

How to use Context Links in Reverb 2.0

What happens if a Reverb link is no longer valid?

How do I know what baggage files are in my Reverb 2.0 output?

What user interactions can Reverb track in Google Analytics?

How can I modify the Reverb search result summary?

How to add Keywords meta data to generated HTML pages

Why does HTML content within my source file not publish to PDF?

## What changed in Font Awesome version 5.15.4 from 4.7.x that impacts Reverb 2.0?

The Font Awesome library included with ePublisher and used in the **WebWorks Reverb 2.0** format was upgraded the **2021.1** version of ePublisher.

Most of the icons and glyphs used by **WebWorks Reverb 2.0** look and function the same, however there are some differences to be aware of that may impact your own advanced customizations.

### Three new classes were introduced: `fas` `far` `fab`

- `fas`, which replaces `fa` is typically used to get the *solid* version of an icon. `fa` is backward compatible with `fas` and may not need to be changed in pre-existing implementations.
- `far` is typically used to get the *regular* version of an icon, and is thinner and lighter than the `fas` version.
- `fab` is used for icons that represent a brand, for example the *Twitter* icon.

# Font Awesome Cheatsheet

Free to use icons can be found at: <https://fontawesome.com/v5/cheatsheet/free>

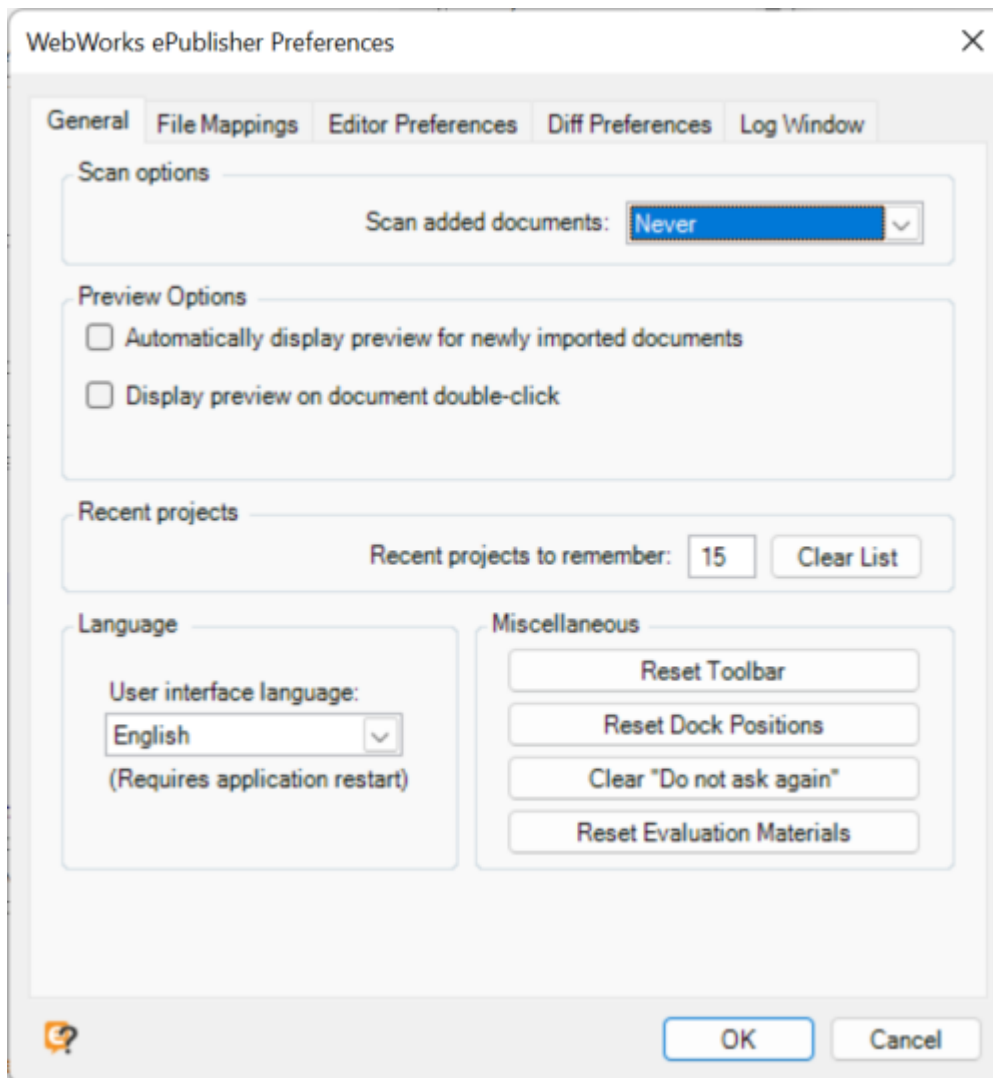
## How do I disable automatic Preview generation in Designer?

In ePublisher Designer's **Document Manager**, to disable the automatic generation of *Preview* tabs each time a document is added, do the following:

1. Open the **Preferences** dialog using menu: **Edit > Preferences**.
2. Uncheck **Automatically display preview for newly imported documents**.
3. Select **OK**

In addition, you can disable generation of **Preview** tabs when double-clicking documents in the **Document Manager** by also unchecking:

**Display preview on document double-click**



## How do I capture source file meta data in the published HTML output?

You can capture location specific meta data within your source content and then use it in your generated HTML pages.

The most common way to use this kind of data is through the `Page.asp` template file.

For example, a marker called `ReviewDate` in your source content could contain the date this section of the file was reviewed. Then using an advanced customization of `Page.asp`, the following would conditionally emit its value on the generated HTML page.



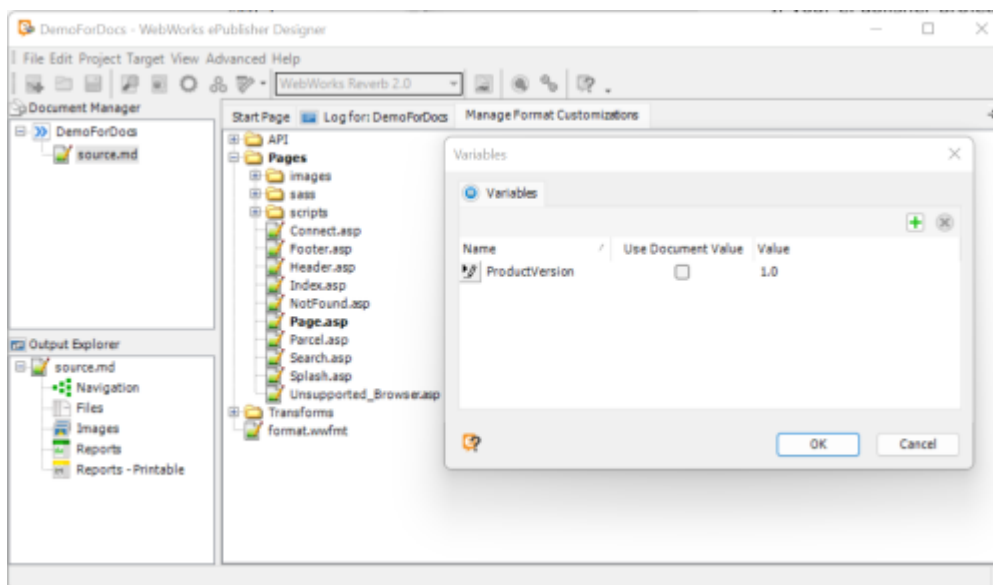
```
<div wwpage:content="wwmarker:ReviewDate"  
  wwpage:condition="wwmarker:ReviewDate">
```

October 20, 2021

```
</div>
```

## How do I use a Project Variable's value in the published output?

If your ePubublisher project configures a specific variable to a non-empty value, then that value can be captured in your generated output even if the variable is not used in the source content.



Project variables can be captured using attributes in the `Page.asp` template file.

For example, if a project variable called `ProductVersion` has been configured in ePubublisher to a non-empty value such as: `1.0`, then it can be published in your generated output as follows:

```
<div  
  wwpage:content="projvars:ProductVersion"  
  wwpage:condition="projvars:ProductVersion">  
  x.x  
</div>
```

## How do I upgrade an existing WebWorks Reverb 2.0 project to a newer version?

The following video walks through all of the steps that may be required to upgrade an existing **WebWorks Reverb 2.0** project.

Watch on youtube to access the video's *quick links*.

[See video on Reverb 2.0 Typical Customization Tasks.](#)

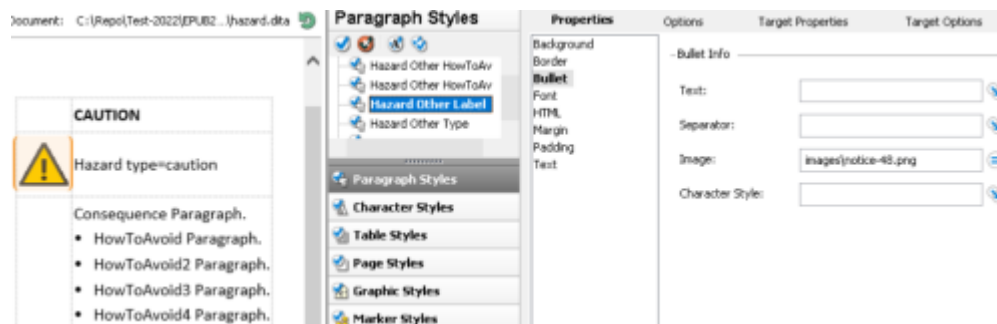
## How to add an image icon and stylize your DITA Hazard statement elements?

Specify an image file to use for your `Hazard Caution Label` paragraph style in the Style Designer.

It will show up to the left of your content's Hazard statement.

For example:

### ePublisher Designer *Display Preview*



For each type of Hazard statement, there are a set of Designer styles as follows:

- Paragraph
  - Hazard
  - Hazard  Consequence
  - Hazard  HowToAvoid
  - Hazard  HowToAvoid End
  - Hazard  Label
  - Hazard  Type
- Table
  - Hazard

**Note:**

There is a placeholder style used that is shared by all Hazard statements for the empty regions of the table. It does not need to be styled.

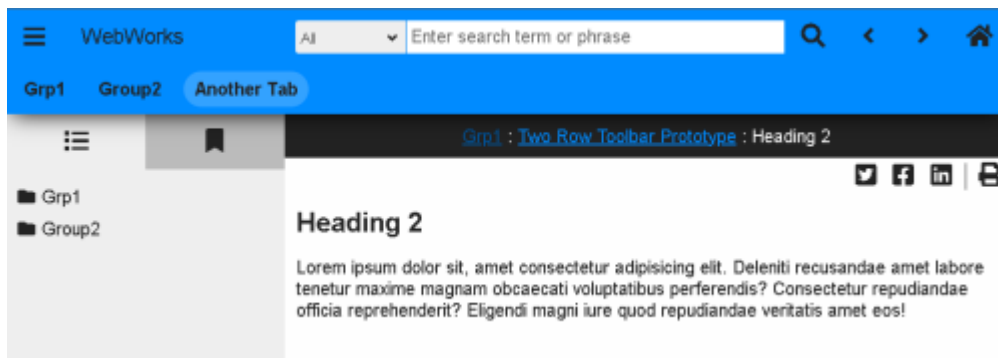
- Paragraph
  - Hazard Empty

## How to supplement or replace Reverb Toolbar Group Tabs

Create an **Advanced Customization** of the file `Connect.asp` to add to or replace the Reverb Toolbar group tabs.

For example:

### Reverb 2.0 Toolbar with Additional Tab



## Steps

1. Create **Advanced Target Customization** for `Connect.asp`.
2. Locate HTML code for tabs by searching for `toolbar-tabs` in `Connect.asp`.
3. Copy existing `<li wwpage:replace="toolbar-tabs">..</li>` element with sub-content to use as starting point for each new tab to create.
4. Remove attribute `wwpage:replace="toolbar-tabs"` from your newly pasted elements. This attribute is only used to inject a list of the ePublisher project's group tabs into the output.
  - **Note:** If you want to remove the group tabs from the toolbar, remove the `wwpage:replace="toolbar-tabs"` from all tab `<li>` elements.
5. Specify desired `href` destination, usually a context specifier with **Topic Alias ID**, such as: `#context/<TOPIC ALIAS ID>`.
6. Specify desired `title` attribute to produce *fly over* text for your tab.
7. Specify the text to appear on the tab itself.
8. Optionally, add a target attribute to open page in a new window, which is required for destinations outside of the help set.
  - For example: `<a target="_blank" href="#context/LandingPage2" title="Landing Page 2">`

## Example Reverb Toolbar Tab Code

```
<ul class="ww_skin_toolbar_tab_group">

  <li wwpagereplace="toolbar-tabs">

    <div class="ww_skin_toolbar_tab">

      <a href="connect/splash.html" title="Home">

        Home

      </a>

    </div>

  </li>

  <li>

    <div class="ww_skin_toolbar_tab">

      <a href="#context/LandingPage2" title="Another Tab">

        Another Tab

      </a>

    </div>

  </li>

</ul>
```

## Invalid File Path Characters Filtered from Generated File Names

When working with file systems and browsers, certain file path characters are either forbidden or problematic. With ePublisher, you can use target settings and/or `Filename` markers to influence the file names generated in your published output. However, there are some characters that are considered *invalid* and ePublisher will not use them when creating output file names.

You don't need to worry or track the invalid characters, because ePublisher will automatically eliminate them from the output. In addition, any time an invalid

character is attempted to be used to generate a file name, ePublisher will replace that character with a single '\_' (underscore) character as a placeholder.

For example, if you had a heading paragraph that is attempting to generate a file name such as: `Document (%25).html`. The actual generated file name will be: `Document (_25).html` because the % character is a problematic character.

*Aside:* In URLs the % is a special character that is used to encode other characters, making it problematic to use this character as itself in URL paths.

## Invalid File Path Characters

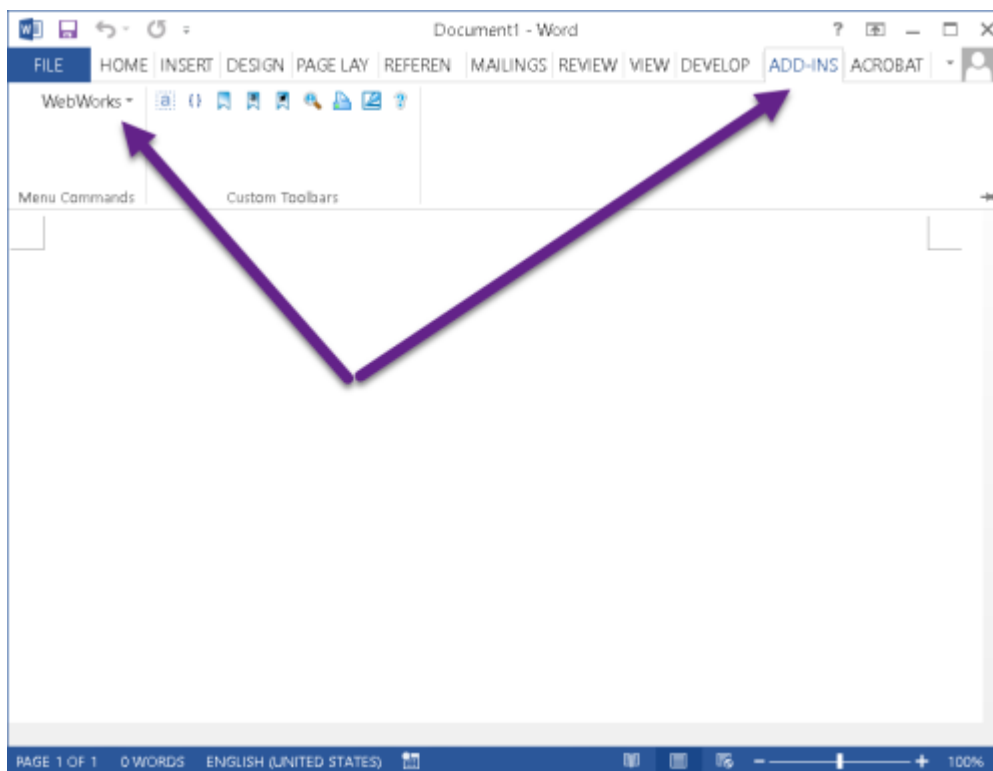
```
"#$%&*+, / : ; < = > ? [ \ ] |
```

Each occurrence of one of these characters will be replaced by a single \_ (underscore) character.

## How to enable 'WebWorks Menu' (Transit) Add-in for Microsoft Word

If you have **WebWorks ePublisher** component installed on your system and when running *Microsoft Word* you do not see a **WebWorks** menu, then this article will explain how to enable the Add-in called `Transit.dotm`.

### WebWorks Menu Add-in for Microsoft Word



## Steps to Enable

**Note:** There are several ways to make the `transit.dotm` add-in available in the Word toolbar (ribbon). The **ePublisher Express** installer will normally install the add-in file to the prescribed location for your version of Microsoft Word. However, sometimes the computer system will block the installer from copying the add-in to this location. Find the location below for your version of Word and make sure the file has been copied there. If it is not there, you can manually copy it from the ePublisher installation folder, usually at the following.

```
C:\Program Files (x86)\WebWorks\ePublisher\2022.1\Transit
```

### 1. Location of `transit.dotm` based on the version of Microsoft Word.

- Word 2016 (64-bit) or later versions, possible locations:

```
C:\Program Files\Microsoft Office\office16\Startup\
```

```
C:\Program Files\Microsoft Office\root\office16\Startup\
```

- Word 2013 (64-bit), possible locations:

```
C:\Program Files\Microsoft Office\office15\Startup\
```

```
C:\Program Files\Microsoft Office\root\office15\Startup\
```

- Word 2016 (32-bit) or later versions, possible locations:

```
C:\Program Files (x86)\Microsoft Office\office16\Startup\
```

```
C:\Program Files (x86)\Microsoft Office\root  
\office16\Startup\
```

- Word 2013 (32-bit), possible locations:

```
C:\Program Files\Microsoft Office\office15\Startup\
```

```
C:\Program Files\Microsoft Office\root\office15\Startup\
```

2. On Microsoft Word Ribbon, click **File > Options > Trust Center > Trusted Locations**.

- Click the **Add New Location** button
- Add the directory path noted above for `transit.dotm`.

## How do I set image horizontal alignment in ePublisher?

In ePublisher version 2022.1 and above, you can control the image alignment of a *Graphic Style* using the *Style Designer* to modify that style's property: **HTML > Text > (Alignment) Horizontal**.

### Possible Horizontal Property Values

- Left
- Center
- Right

**Note:** If your authoring environment supports assigning horizontal alignment, then that property will be inherited by the *Graphic Style*, which means the same alignment will be used in the generated output without having to explicitly set it in ePublisher's *Style Designer*.

### Floating Image to Left or Right of Paragraph

To float an image style to the left or right of the paragraph that contains it, use the *Style Designer* to modify the graphic style properties as follows.



Graphic Style Property	Value
HTML > Display	Block
HTML > Float	Left or Right

## Why is the PDF format skipping Markdown files?

When generating the Output Format **PDF** using Markdown source documents, this message will appear in the generation log:

```
[Warning] Skipping file ... PDF Format is only available for Word and
FrameMaker documents. Use PDF - XSL-FO Format instead.
```

To publish PDF output using Markdown files, use the format **PDF - XSL-FO**, which is available for all source document types.

The legacy format called: **PDF** was created only for publishing *Adobe FrameMaker* and *Microsoft Word* source documents, and thus is not available for publishing Markdown source content. If you do attempt to publish a markdown file using the **PDF** format, you will get a warning message in your ePublisher generation log.

## How can a markdown file include another?

Using Markdown++ you can include the contents of one markdown file in another, similar to the following:

```
<!--include:../Topics/MyTopic.md-->
```

In the above statement, a filename from the parent folder's *Topics* folder is being included in markdown file.

To include markdown files in other folders, be sure to use a path that is relative to the file that contains the *include* statement. Absolute paths will work as well, but might not be as portable if you move your content from one location to another.

## How can you insert an Index Marker into a Markdown file?

In your markdown file, you can insert an **IndexMarker** using the following syntax:

```
<!--markers:{"IndexMarker": "term1 term2"}-->
```

```
Paragraph to be used for the index terms: term1 and term2.
```

## How to use alternate DITA-OT installation with ePublisher

Set the `DITA_HOME` environment variable and ePublisher will use that installation of the DITA Open Toolkit instead of the toolkits that it maintains in its installed location.

ePublisher doesn't require anything unique for its use of the DITA-OT, however, you may have your own custom plugins, in which case, this may be an easier way to maintain and use the DITA-OT with ePublisher.

### Steps

1. Set the environment variable `DITA_HOME` to the installation location of the DITA-OT that you are using. For example: `C:\dita-ot-4.0`.
2. Make sure that your ePublisher project is configured to use the same version number (or closest) using the menu: **Project > Project Settings....** Locate the the setting: **Input Configurations > DITA Open Toolkit version**.

## How do I embed HTML within DITA source content?

It is often necessary to embed HTML within your DITA source content for supporting the display of media such as youtube videos.

The two most common ways to embed HTML are through the use of a `<foreign>` element or a paragraph with a custom `outputclass` assigned to it.

An example for each is as follows.

### Using the `<foreign>` element

**Note:** Embedding a link to a local file, still requires that file to exist at the specified link location. In other words, ePublisher will not handle it as a **Baggage File**.

```
<foreign audience="web">

  <![CDATA[

    <iframe width="800" height="500" src="_animation/
AC_41_Rackwinkel_1.html" frameborder="0" allowfullscreen="true">

    </iframe>

  ]]>

</foreign>
```

### Using `@outputclass` to enable *Pass Through* behavior

```
<p outputclass="PassThrough" audience="web">

  <![CDATA[

    <iframe

      width="660"

      height="375"

      src="https://www.youtube.com/embed/D4QoQWboM-U?rel=0"

      frameborder="0"

      allow="accelerometer; autoplay; encrypted-media; gyroscope;
picture-in-picture"

      allowfullscreen>

    </iframe>

  ]]>

</p>
```

## Steps in ePubliher Designer

By default ePubliher disables HTML code within the source content, so you will need to *enable* the appropriate paragraph style's **Pass Through** option in the *Style Designer*.

1. In ePubliher Designer, open the *Style Designer* and select the name of the paragraph style assigned with the `@outputclass` attribute or if you used `<foreign>`, then the style name will be `foreign`.
2. Select the **Options** tab and then set the option: **Pass Through** to **Enabled**.
3. The ePubliher project will now generate the embedded HTML correctly and you can deploy an updated Stationery for use in other Express projects.

## How to use Context Links in Reverb 2.0

If you are generating online help it may be more robust to use a context link to a topic alias identifier instead of a traditional hyperlink to a filename and anchor.

## Link Structure

```
#context/<TopicAlias Identifier>
```

```
#context/<Context>/<TopicAlias Identifier>
```

With a Reverb context link you are creating a link to a location. This is different from a traditional hyperlink which points to a specific filename.

## What happens if a Reverb link is no longer valid?

When end-users access an invalid link in your published Reverb 2.0 online help a standard page will be displayed in the same locale as the online help files. The default text for this page is as follows:

### 404

## File Not Found

## How do I know what baggage files are in my Reverb 2.0 output?

If you are linking to non-source files from within your source content, ePublisher will treat them as **Baggage Files** and make them available as part of your generated help set.

In Reverb 2.0, the generated file `url_maps.xml` (default filename) contains a complete list of all the baggage files in your help set. The baggage file entries will look similar to the following:

```
<BaggageMap>

  <Baggage

    basename="download_kit.zip"

    path="UserGuide\baggage\download_kit.zip"

    groupID="0jQCq1sTUkc"

  />

  .

  .

  .

</BaggageMap>
```

## What user interactions can Reverb track in Google Analytics?

When using the WebWorks Reverb 2.0 output format, you can enable the target setting for **Google Analytics** and capture end-user interactions with the online help content.

The users interactions that are tracked are as follows:

- Content page view
- Content page click
- Content page scroll
- Content page PDF button click
- Content page Print button click
- Content page "Back to top" button click
- Content page "Was this helpful - Yes" button click
- Content page "Was this helpful - No" button click
- Search query
- Search result page view
- Search result page "Was this helpful - Yes" button click
- Search result page "Was this helpful - No" button click
- Toolbar menu button click

- Toolbar search button click
- Toolbar previous button click
- Toolbar next button click
- Toolbar home button click
- Toolbar translate button click
- Menu TOC button click
- Menu Index button click
- "Context Sensitive Help" access

## How can I modify the Reverb search result summary?

The search result summaries use **Description** marker values when available in the source content.

The marker must occur somewhere within the content that makes up the generated output HTML page. Markers in other locations will not affect the search result summary.

If a **Description** marker is not found, then the summary will be calculated from the first paragraph(s) within the generated HTML page.

## How to add Keywords meta data to generated HTML pages

When generating HTML-based output such as the **WebWorks Reverb 2.0** or **Dynamic HTML** formats, you can use a **Keywords** marker to set the *Keywords* meta data for the generated HTML page.

In some search engines, setting the **Keywords** meta data can make sure that your audience knows that those keywords are important in the HTML file, even if those keywords are not actually present in the file.

In the **WebWorks Reverb 2.0" format**, the **Keywords** marker can make those words show up higher in priority in the search results when they are queried.

For example, in the following markdown source content, the keyword "WidgetXYZ" will cause generated HTML page to show up with high priority in the search results.

```
<!--markers:{"Keywords": "WidgetXYZ"}-->
```

```
# The Widget Company
```

```
We make all kinds of widgets.
```

## Why does HTML content within my source file not publish to PDF?

PDF output does not support pass-through HTML. This includes even HTML content within Markdown source files.

When ePublisher generates PDF output from content that contains HTML, the HTML content is ignored since PDF viewers do not understand HTML.