

Oracle® Fusion Middleware

Technical Reference Guide for Site Studio

11g Release 1 (11.1.1)

E10615-01

May 2010

Oracle Fusion Middleware Technical Reference Guide for Site Studio, 11g Release 1 (11.1.1)

E10615-01

Copyright © 1996, 2010, Oracle and/or its affiliates. All rights reserved.

Primary Author: Sean Cearley

Contributor: Brian Cheyne

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

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

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

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

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

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

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

Contents

Preface	xv
Audience	xv
Documentation Accessibility	xv
Related Documents	xvi
Conventions	xvi
1 Introduction	
1.1 About the Site Studio Technical Reference	1-1
1.2 Scope of the Technical Reference	1-1
2 Site Asset Relationships and File Structure	
2.1 About Site Assets	2-1
2.2 Templates	2-2
2.2.1 Page Templates	2-2
2.2.2 Subtemplates	2-2
2.2.3 Region Templates	2-3
2.3 Definitions	2-3
2.3.1 Placeholder Definitions	2-3
2.3.2 Region Definitions	2-4
2.3.3 Element Definitions	2-6
2.4 Relationship Between Templates and Definitions	2-6
2.5 Serving Web Pages	2-6
2.6 Conversions Definitions	2-7
2.7 Contribution Mode	2-7
2.7.1 Key Command	2-7
2.7.2 Query String	2-8
2.7.3 Session Cookie	2-8
2.7.4 Process from Consumption Mode to Contribution Mode	2-8
3 Site Studio Metadata	
3.1 About Metadata	3-1
3.2 Metadata Fields	3-1
3.2.1 xWebsiteObjectType	3-1
3.2.2 xWebsiteSection	3-3
3.2.3 xWebsites	3-3

3.2.4	xDontShowInListsForWebsites	3-4
3.2.5	xRegionDefinition.....	3-4

4 Link Formats

4.1	About Link Formats.....	4-1
4.2	Using Path-Based Links	4-1
4.3	Using wcmUrl	4-2
4.4	Using Server-Side Script Links	4-3
4.4.1	ssLink.....	4-3
4.4.2	ssNodeLink.....	4-4
4.4.3	ssWebLayoutUrl	4-4
4.5	Using Token Links	4-5
4.5.1	ssLINK.....	4-5
4.5.2	ssNODELINK.....	4-6
4.5.3	Token Links as Returned Server-Side Links	4-6
4.6	Using JavaScript Links	4-7

5 Site Studio Runtime Generated Files

5.1	About Runtime Generated Files	5-1
5.2	Runtime Auto-Generated Files	5-1
5.2.1	sitenavigation.js.....	5-1
5.2.2	sitenavigationfunctions.js	5-3
5.2.3	sitenavigation.xml	5-3
5.2.4	sitenavigation.hda	5-4
5.2.5	sitenavigation_co.hda.....	5-4
5.3	wcm.toggle.js	5-4

6 Fragments

6.1	About Fragments	6-1
6.2	Fragment Libraries.....	6-2
6.3	Read-Only Fragment Libraries	6-3
6.4	Fragment Inclusion Using wcmFragment.....	6-3
6.5	Fragment Snippets and the ssIncludeXml().....	6-4
6.6	Fragments That Use Custom Section Properties	6-4
6.6.1	Client-Side JavaScript.....	6-4
6.6.2	Server-Side Idoc Script.....	6-5
6.7	The Fragment Definition File	6-5
6.7.1	<fragments>.....	6-6
6.7.2	<fragment>	6-6
6.7.3	<parameter>	6-8
6.7.3.1	<option>.....	6-9
6.7.3.2	<querytext>	6-9
6.7.3.3	<validate>	6-9
6.7.3.4	<convert>	6-10
6.7.3.5	<customgui>.....	6-11
6.7.4	<snippet>	6-12

6.7.5	<designview>	6-13
6.7.6	<element>	6-13
6.8	Fragment Instance Structure in the <ssinfo> XML Data Island	6-13

7 Native Documents and Conversion

7.1	About Native Documents	7-1
7.2	wcmDynamicConversion	7-1
7.3	Document Conversion in the Properties Pane.....	7-1
7.4	Common Errors Using Native Documentation.....	7-2

8 Custom Elements

8.1	About Custom Elements	8-1
8.2	Element API.....	8-1
8.2.1	Loading the Element API	8-1
8.2.2	Element API Dependent Scripts	8-2
8.3	Custom Elements within Contributor	8-3
8.3.1	Communication from a Contributor Form to a Custom Element	8-3
8.3.2	Communication from a Custom Element to a Contributor Form	8-4
8.4	Legacy Custom Element Compatibility.....	8-5
8.4.1	Detecting Legacy Custom Element Forms.....	8-5
8.4.2	Upgrading Legacy Custom Elements	8-6

9 Idoc Script Extensions

9.1	About Idoc Script Extensions	9-2
9.2	wcmPlaceholder	9-2
9.3	wcmElement	9-3
9.4	wcmListStart	9-4
9.5	wcmListEnd	9-4
9.6	wcmListElement.....	9-4
9.7	wcmListRowCount.....	9-5
9.8	wcmDynamicList	9-5
9.9	wcmIncludeElement.....	9-6
9.10	wcmDynamicConversion	9-6
9.11	wcmFragment.....	9-7
9.12	wcmUrl	9-7
9.13	ssIncludeXml	9-8
9.14	ssGetDocInfo.....	9-9
9.15	ssGetXmlNodeCount.....	9-9
9.16	ssIncDynamicConversion	9-10
9.17	ssIncDynamicConversionByRule	9-10
9.18	ssIncDynamicConversionByRulesEngine	9-11
9.19	ssIncInlineDynamicConversion.....	9-11
9.20	ssIsNativeDoc	9-11
9.21	ssRandom	9-12
9.22	ssGetNodeProperty	9-12
9.23	ssGetWebsiteNodeType.....	9-13

9.24	ssGetCoreMajorVersion	9-13
9.25	ssSplitString	9-13
9.26	ssGetWebsiteName	9-14
9.27	ssGetSiteProperty	9-14
9.28	ssGetFirstNodeId	9-15
9.29	ssGetRelativeNodeId	9-15
9.30	ssLoadSiteNavResultSet	9-15
9.31	ssGetServerRelativeUrl	9-16
9.32	ssGetServerRelativePath	9-16
9.33	ssGetUrlPageName	9-17
9.34	ssGetNodeLabel	9-17
9.35	ssGetNodeLabelPath	9-17
9.36	ssGetAllSites	9-18
9.37	ssLink	9-18
9.38	ssNodeLink	9-18
9.39	ssWeblayoutUrl	9-19

10 Idoc Script Variables

10.1	About Idoc Script Variables	10-1
10.2	HttpWebsitesRoot	10-1
10.3	HttpRelativeWebsitesRoot	10-1
10.4	HttpFragmentsRoot	10-2
10.5	HttpRelativeFragmentsRoot	10-2
10.6	SS_SERVER_NAME	10-2
10.7	HttpASPPath	10-2
10.8	ssServerRelativeSiteRoot	10-2

11 Site Studio Services

11.1	About Site Studio Services	11-1
11.2	Services Related to Contributor	11-1
11.3	Services Related to Designer	11-2
11.4	Services Related to Manager	11-3
11.5	Services Related to Switch Content	11-4
11.6	Services Related to Link Wizard	11-4
11.7	List of Services	11-4
11.7.1	SS_ADD_NODE	11-7
11.7.2	SS_ADD_WEBSITE_ID	11-7
11.7.3	SS_BATCH_DECODE_LINK	11-7
11.7.4	SS_CHECKIN_FRAGMENT_LIBRARY	11-8
11.7.5	SS_CHOOSE_WEBSITE_SECTION	11-8
11.7.6	SS_CHOOSE_WEBSITES	11-8
11.7.7	SS_CLEAR_PREVIEW	11-8
11.7.8	SS_CLEAR_REGION_ASSOCIATIONS	11-9
11.7.9	SS_CLEAR_WEBSITE_ID	11-9
11.7.10	SS_COMMIT_SITE_CHANGES	11-9
11.7.11	SS_CREATE_NEW_SITE_EX2	11-9
11.7.12	SS_CREATE_SITE_NAV_JS	11-10

11.7.13	SS_DECODE_LINK.....	11-10
11.7.14	SS_DELETE_NODE.....	11-11
11.7.15	SS_DOC_INFO_LATEST.....	11-11
11.7.16	SS_EDIT_NATIVE_DOCUMENT.....	11-11
11.7.17	SS_GET_ADMIN_PAGE.....	11-12
11.7.18	SS_GET_ALL_CUSTOM_NODE_PROP_DEFS.....	11-12
11.7.19	SS_GET_ALL_NODE_PROPERTIES.....	11-12
11.7.20	SS_GET_ALL_SITE_DOMAINS.....	11-12
11.7.21	SS_GET_ALL_SITE_PROPERTIES.....	11-13
11.7.22	SS_GET_ALL_SITES_EX2.....	11-13
11.7.23	SS_GET_CONFIG_INFO.....	11-13
11.7.24	SS_GET_CONTRIBUTOR_CONFIG.....	11-13
11.7.25	SS_GET_CONTRIBUTOR_STRINGS.....	11-14
11.7.26	SS_GET_DC_RULES.....	11-14
11.7.27	SS_GET_DOCUMENT_LABELS.....	11-14
11.7.28	SS_GET_DOCUMENT_USAGE.....	11-14
11.7.29	SS_GET_ENVIRONMENT_PROPERTY_NAMES.....	11-15
11.7.30	SS_GET_FIRST_NODE_ID.....	11-15
11.7.31	SS_GET_FRIENDLY_URL.....	11-15
11.7.32	SS_GET_LINK.....	11-16
11.7.33	SS_GET_LINK_MANAGEMENT_REPORT.....	11-16
11.7.34	SS_GET_LINK_WIZARD_CONFIG.....	11-16
11.7.35	SS_GET_LINK_WIZARD_CONFIG_WITH_SITE.....	11-17
11.7.36	SS_GET_NODE_LINK.....	11-17
11.7.37	SS_GET_NODE_PROPERTY.....	11-18
11.7.38	SS_GET_PAGE.....	11-18
11.7.39	SS_GET_PLACEHOLDER_SWITCH_CONTENT_CONFIG.....	11-21
11.7.40	SS_GET_REGION_ASSOCIATIONS.....	11-22
11.7.41	SS_GET_REGION_DEFINITION_ELEMENTS.....	11-22
11.7.42	SS_GET_RELATIVE_NODE_ID.....	11-22
11.7.43	SS_GET_SEARCH_RESULTS.....	11-22
11.7.44	SS_GET_SITE_AS_XML_EX2.....	11-23
11.7.45	SS_GET_SITE_ASSET_CATEGORIES.....	11-24
11.7.46	SS_GET_SITE_CHANGE_MONITOR_TOKEN.....	11-24
11.7.47	SS_GET_SITE_DEFINITION.....	11-24
11.7.48	SS_GET_SITE_DEFINITION_FOR_USER.....	11-24
11.7.49	SS_GET_SITE_DOMAINS.....	11-25
11.7.50	SS_GET_SITE_FRAGMENT_ASSET_REPORT.....	11-25
11.7.51	SS_GET_SITE_INFO.....	11-25
11.7.52	SS_GET_SITE_PROPERTY.....	11-25
11.7.53	SS_GET_SITE_PUBLISH_REPORT.....	11-26
11.7.54	SS_GET_SITE_REPORT.....	11-26
11.7.55	SS_GET_SWITCH_CONTENT_CONFIG.....	11-26
11.7.56	SS_GET_UNIQUE_NODE_SITE_ID.....	11-26
11.7.57	SS_GET_VERSION.....	11-27
11.7.58	SS_GET_WEBLAYOUT_URL.....	11-27
11.7.59	SS_IS_JS_NAV_OUT_OF_DATE.....	11-27

11.7.60	SS_MAP_FRIENDLY_NAME.....	11-27
11.7.61	SS_MOVE_NODE.....	11-28
11.7.62	SS_PARSE_FRIENDLY_URL.....	11-28
11.7.63	SS_PREPARE_PREVIEW.....	11-28
11.7.64	SS_PUBLISH_THIS_PAGE.....	11-29
11.7.65	SS_REMOVE_WEBSITE_ID.....	11-29
11.7.66	SS_SET_ALL_CUSTOM_NODE_PROP_DEFS.....	11-29
11.7.67	SS_SET_ELEMENT_DATA.....	11-29
11.7.68	SS_SET_ENVIRONMENT_PROPERTY_NAMES.....	11-30
11.7.69	SS_SET_NODE_PROPERTY.....	11-30
11.7.70	SS_SET_NODES_PROPERTIES.....	11-30
11.7.71	SS_SET_PREVIEW_ELEMENT_DATA.....	11-31
11.7.72	SS_SET_SITE_ASSET_CATEGORIES.....	11-31
11.7.73	SS_SET_SITE_DOMAINS.....	11-31
11.7.74	SS_SET_SITE_PROPERTIES.....	11-32
11.7.75	SS_SET_SITE_PROPERTY.....	11-32
11.7.76	SS_SWITCH_REGION_ASSOCIATION.....	11-32
11.7.77	SS_VALIDATE_WEBSITE_OBJECT.....	11-33
11.7.78	WCM_PLACEHOLDER.....	11-33
11.7.79	WCM_EDIT_DATA_FILE.....	11-34
11.7.80	WCM_BEGIN_EDIT_SESSION.....	11-34

12 Site Studio Configuration Flags

12.1	About Site Studio Flags.....	12-5
12.2	DisableSiteStudioContribution.....	12-5
12.3	ShowSiteStudioMissingDataFileErrors.....	12-5
12.4	SiteStudioValidateElementDefinitions.....	12-6
12.5	SiteStudioValidateRegionDefinitions.....	12-6
12.6	SiteStudioValidatePlaceholderDefinitions.....	12-6
12.7	SiteStudioValidateConversionsDefinitions.....	12-6
12.8	SiteStudioValidateDataFiles.....	12-7
12.9	SiteStudioValidateProjects.....	12-7
12.10	SSAccessDeniedHeader.....	12-7
12.11	SSAccessDeniedReplacementHeader.....	12-7
12.12	SSAccessDeniedUserAgentExceptions.....	12-8
12.13	SSAccommodateWelcomeFile.....	12-8
12.14	SSAdditionalNavResultSetFields.....	12-8
12.15	SSAddSecurityIDValues.....	12-8
12.16	SSAfterProjectLoadedProperties.....	12-9
12.17	SSAllowDynamicDefinitions.....	12-9
12.18	SSAllowEmptyUrlPageName.....	12-9
12.19	SSAllowNotModifiedHeader.....	12-9
12.20	SSAltTagFieldName.....	12-10
12.21	SSAlwaysRecordServerConfig.....	12-10
12.22	SSAssumeXmlIsUtf8.....	12-10
12.23	SSAutoCheckinBusyTimeout.....	12-10
12.24	SSBackupCollectionName.....	12-11

12.25	SSCacheControlOverride.....	12-11
12.26	SSCanGenerateUniqueDataFiles	12-11
12.27	SSChangeAccessDeniedHeaders	12-11
12.28	SSCheckAssignedContentAccess	12-12
12.29	SSCheckBrowserForSiteRoot.....	12-12
12.30	SSCheckWebsiteObjectSecurity	12-12
12.31	SSClearDefinitionArchiveWebsites.....	12-12
12.32	SSCompressorArguments.....	12-13
12.33	SSCompressorCommand.....	12-13
12.34	SSCompressorDir	12-13
12.35	SSCompressorJar	12-13
12.36	SSCompressorMainClass	12-14
12.37	SSCompressorTimeout.....	12-14
12.38	SSCompressorTimerInterval	12-14
12.39	SSCompressorWaitForever.....	12-14
12.40	SSContributorSourceDir	12-15
12.41	SSCustomNodePropertyDefsPermissions	12-15
12.42	SSDefaultDocumentsFields	12-15
12.43	SSDefaultEditor	12-15
12.44	SSDefaultExternalDocNamePrefix	12-16
12.45	SSDefaultExternalDocNameSuffix	12-16
12.46	SSDefaultExternalServerRelativeSiteRoot.....	12-16
12.47	SSDefaultExternalUrlPrefix	12-16
12.48	SSDefaultExternalUrlSuffix	12-17
12.49	SSDefaultPlaceholderDefinition	12-17
12.50	SSDefaultRegionTemplate	12-17
12.51	SSDefaultUrlPageName	12-17
12.52	SSDetectIncludeFileEncoding	12-18
12.53	SSDICPlaceholderDefinition	12-18
12.54	SSDirectDeliveryExtensions	12-18
12.55	SSDirectDeliveryOverrideProperty	12-18
12.56	SSDirectDeliveryProperty.....	12-19
12.57	SSDirectDeliveryRequiredExtensions.....	12-19
12.58	SSDisableDeferredNodeExpansion.....	12-19
12.59	SSDisableIncludeXmlCache	12-19
12.60	SSDisableLinkResolutionSiteLocking	12-20
12.61	SSDisableProjectDeferredNodeExpansion.....	12-20
12.62	SSDomCacheDefaultFileSizeFactor.....	12-20
12.63	SSDomCacheFileSizeFactors	12-20
12.64	SSDomCacheLowerBound	12-21
12.65	SSDomCacheMultiplier.....	12-21
12.66	SSDomCacheNodeMultiplier.....	12-21
12.67	SSDomCacheStringMultiplier	12-22
12.68	SSDomCacheStringOverhead	12-22
12.69	SSDomCacheUseDOM.....	12-22
12.70	SSDomCacheUseFileSize	12-22
12.71	SSEditorDebugLevel.....	12-22

12.72	SSEnableASPSupport	12-23
12.73	SSEnableDirectDelivery	12-23
12.74	SSEnableExtranetLookCompatibility.....	12-23
12.75	SSEnableFolioEditing	12-23
12.76	SSEnableFormEditing.....	12-24
12.77	SSEnableJavaScriptCompressor.....	12-24
12.78	SSEnableUpperCaseColumnsCheck	12-24
12.79	SSGenerateUniqueNodeIds.....	12-24
12.80	SSHidePrimaryFileInContributor.....	12-25
12.81	SSHttpAbsoluteHelpRoot.....	12-25
12.82	SSHttpLayerManager	12-25
12.83	SSIdocMarker	12-25
12.84	SSIgnoreMaxAgeNodeProperties.....	12-25
12.85	SSIgnoreNoProjectDefaultMetadataMessage.....	12-26
12.86	SSIgnoreReadyToReplicate	12-26
12.87	SSImportOnlyLatestRevs.....	12-26
12.88	SSIncludeInactiveNodesInNavResultSet	12-26
12.89	SSIncludeInactiveNodesInNavXML	12-27
12.90	SSIncludeRegionTemplatesInDefinitionBundles	12-27
12.91	SSIncludeXmlTransformFormat	12-27
12.92	SSIncludeXmlTransformIndent	12-27
12.93	SSJavaExecutablePath	12-27
12.94	SSJSONContentType	12-28
12.95	SSLoadCustomElementsWithOnDemandEditors.....	12-28
12.96	SSLoadProjectsAtStartup.....	12-28
12.97	SSLoadUncompressedFckSource	12-29
12.98	SSManuallyValidateNodeIdUniqueness.....	12-29
12.99	SSMaxNodeIdLength	12-29
12.100	SSMaxSiteIdLength.....	12-29
12.101	SSMaxSitesMenuItems	12-30
12.102	SSMaxTemplateEvaluationStack	12-30
12.103	SSMigrationCollectionName.....	12-30
12.104	SSOmitFragmentLibrariesInArchiverQueries	12-30
12.105	SSOnDemandEditorsThresholdCount.....	12-31
12.106	SSPrefillUrlDirNamesDuringUpgrade	12-31
12.107	SSProjectAutoCheckinInterval.....	12-31
12.108	SSProjectLoadFailureTracingSection	12-31
12.109	SSProjectReleaseSleepTime	12-32
12.110	SSProjectReleaseWaitTime	12-32
12.111	SSQuickDiffDefaultRegionTemplate	12-32
12.112	SSShowAssignmentTooltips	12-32
12.113	SSSQLUseContains	12-32
12.114	SSStoppedSiteResponsePageDocName.....	12-33
12.115	SSSuppressAddToWebsite	12-33
12.116	SSSuppressLargeCssOptimization	12-33
12.117	SSTempProjectLifetime	12-34
12.118	SSTitleTagFieldName.....	12-34

12.119	SSTrackContentAccess	12-34
12.120	SSTrackFragmentAccess	12-34
12.121	SSUrlFieldName.....	12-35
12.122	SSUrlFixupExceptions.....	12-35
12.123	SSUrlPageNames	12-35
12.124	SSUseAbsoluteRedirects	12-35
12.125	SSUseCallbackTrackingForASP	12-36
12.126	SSUseDefaultDocNamePrefix	12-36
12.127	SSUseDefaultServerRelativeSiteRoot.....	12-36
12.128	SSUseDefaultUrlPrefix	12-36
12.129	SSUseMissingLinkTargetFallback	12-36
12.130	SSUseOnDemandContributionModeMenus	12-37
12.131	SSUseUrlSegmentSessionInfo	12-37
12.132	SSValidateCustomElements	12-37
12.133	SSWebFilterIgnoreList.....	12-38
12.134	SSWebLayoutUrlUsesDocNames	12-38
12.135	SSWelcomeFile	12-38
12.136	SSWelcomeFileReplacement	12-38

13 Site Studio Performance Tuning

13.1	About Site Studio Performance.....	13-1
13.2	On-Demand Web Site Management	13-1
13.3	On-Demand Contributor Editors	13-1
13.3.1	About Configuration Flags.....	13-2
13.3.2	Configuration Flags used for On-Demand Editors	13-2
13.3.2.1	SSOnDemandEditorsThresholdCount	13-2
13.4	Optimizing Contributor Code.....	13-2
13.4.1	Optimization Requirements.....	13-3
13.4.2	The Build Process.....	13-3
13.4.3	Building the Optimized Code.....	13-4
13.4.4	Debugging the Build Script.....	13-4
13.4.5	Configuring Site Studio to Use Optimized Code.....	13-5
13.4.6	Customizations and the Build Process	13-5
13.5	Memory Usage	13-5
13.5.1	Flags for Memory Size in XML DOMs	13-5
13.5.2	Flags for Size of Items in the DOC_INFO Cache	13-5
13.5.3	Flags for Controlling the SSXPathCacheEntry Cache	13-6

14 JSON and Contributor

14.1	About JSON	14-1
14.2	Passing Configuration To and From Contributor.....	14-1

15 Contributor Console Window

15.1	About the Contributor Console Window	15-1
15.1.1	Logging Window	15-2
15.1.2	Command Window	15-2

15.1.3	Contributor Console Window User Interface.....	15-2
15.2	Installing the Contributor Console Window	15-3
15.3	Launching the Contributor Console Window	15-3
15.4	Using the Contributor Console Window.....	15-4
15.4.1	Contributor Console Window Context	15-4
15.4.2	Ensuring the Contributor Console Window is Root	15-4
15.4.3	Practical Use of the Contributor Console Window	15-5
15.4.4	Orphaned Console Windows.....	15-5
15.5	Logging Syntax.....	15-5
15.6	Time Profiling	15-6
15.7	Command Window Helper Functions	15-6
15.8	Keyboard Commands	15-7

16 Manager Settings File

16.1	About the Manager Settings File	16-1
16.2	<ssm:settings> Tag	16-1
16.3	<ssm:general> Tag.....	16-2
16.4	<ssm:addSection> Tag	16-2
16.5	<ssm:removeSection> Tag.....	16-3
16.6	<ssm:moveSection> Tag	16-3
16.7	<ssm:setErrorHandler> Tag	16-4
16.8	<ssm:editProperties> Tag	16-4
16.9	<ssm:editCustomProperties> Tag	16-5
16.10	<ssm:primaryLayout> Tag	16-5
16.11	<ssm:secondaryLayout> Tag	16-5
16.12	<ssm:sectionOverride> Tag.....	16-6
16.13	Example Manager Settings File.....	16-7

17 Content Tracker Integration

17.1	Tracked Data.....	17-1
17.2	Configuration Flags	17-2

A Upgrading Pre-7.5 Web Sites

A.1	Introduction	A-1
A.2	What the Automated Upgrade Does.....	A-2
A.3	Upgrading Your Content Servers.....	A-2
A.3.1	Upgrading Sites on a Single Content Server Instance.....	A-3
A.3.2	Upgrading Sites on Multiple Content Server Instances.....	A-3
A.3.3	Performing a Full Upgrade	A-4
A.3.4	Performing a Minimal Upgrade	A-5
A.4	Performing Additional Steps Manually	A-6
A.4.1	Updating the Site Navigation	A-6
A.4.2	Rebuilding the Content Server Index	A-6
A.4.3	Updating Your Custom Fragments.....	A-7
A.4.3.1	Modifying Links That Rely on the <base> Tag.....	A-7
A.4.3.2	Modifying Obsolete SS_GET_PAGE / JavaScript Links	A-7

A.4.3.3	Updating GET_SEARCH_RESULTS	A-7
A.4.4	Updating Your Custom Elements	A-10
A.4.5	Assigning a Web Site Section to Your Folders	A-10
A.4.6	Updating JSP Code	A-11

Index

Preface

The Oracle Fusion Middleware Technical Reference Guide for Site Studio contains information to assist developers and administrators responsible for the implementation of web sites managed by Site Studio.

Audience

This document is intended for those people identified in the organization who are responsible for developing and deploying Web sites managed by Oracle Site Studio.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

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

Related Documents

For more information, see the following documents in the Oracle Site Studio documentation set:

- *User's Guide for Site Studio Contributor*
- *User's Guide for Site Studio Designer*
- *User's Guide for Site Studio Publisher*
- *Administrator and Manager's Guide for Site Studio*

Conventions

The following text conventions are used in this document:

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

Introduction

This guide is built up as follows:

- ["About the Site Studio Technical Reference"](#) on page 1-1
- ["Scope of the Technical Reference"](#) on page 1-1

1.1 About the Site Studio Technical Reference

Welcome to the Technical Reference Guide for Site Studio. This guide provides a broad technical overview of Site Studio and how it creates web sites. It also provides technical information about the project file, markers, tags, service calls, and Idoc Script extensions used by Site Studio. To get the most out of this guide, you should have knowledge of HTML, JavaScript, and server-side Idoc Script; play the role of webmaster or web developer at your organization; and have coding experience.

Before reading the guide, you should have read the *User's Guide for Site Studio Designer* and *User's Guide for Site Studio Contributor*. You should also have built one or more web sites with Site Studio. The guide will illustrate the scripting syntax used by Site Studio, so that you can build upon the existing framework and customize the product to suit your needs.

1.2 Scope of the Technical Reference

The Technical Reference Guide for Site Studio describes the more technical aspects of site construction, maintenance, and consumption using Oracle Site Studio 11gR1, including information on how to customize Site Studio functionality. While Site Studio 11gR1 supports Web sites created with Site Studio 10gR3 and earlier, such Web sites are considered legacy Web sites. For technical information on legacy sites, see the Technical Reference Guide for Site Studio 10gR3.

Site Asset Relationships and File Structure

This section covers the following topics:

- ["About Site Assets"](#) on page 2-1
- ["Templates"](#) on page 2-2
- ["Definitions"](#) on page 2-3
- ["Relationship Between Templates and Definitions"](#) on page 2-6
- ["Serving Web Pages"](#) on page 2-6
- ["Conversions Definitions"](#) on page 2-7
- ["Contribution Mode"](#) on page 2-7

2.1 About Site Assets

The site assets in Site Studio allow for a very modular, customizable method of easily maintaining the content separate from the presentation. The relationship between the templates and the definitions, such as how they are connected across many different pages, may be necessary to know when you want to make specific changes to individual assets in the site.

Site assets are used to directly control the visual presentation of the site, and the actual content on the web pages (the "information"). In this way, the content and the presentation are separate, and can be maintained and modified without affecting each other.

The files that maintain the structure and presentation of the Web site are the templates: page templates, subtemplates, and region templates. Cascading style sheets can also be used to control structure and presentation and managed with Site Studio. The files that maintain the content are the definitions - placeholder definitions, region definitions, and element definitions. With these definitions, you control how the content is maintained.

The content itself is stored in content files - contributor data files, native documents, images, and any other related media (such as Flash) which you may use on your site. Contributor data files are XML formatted files that are generated by Site Studio. Contributor data files are edited using the Site Studio Contributor application. Native documents are files created using familiar third-party applications such as Microsoft Word. Native documents are converted to HTML format using Dynamic Converter, and they are edited using their associated application. Contributors are expected to be in charge of the content, and thus contributor data files are edited through Contributor. Native documents and other files can be edited through the associated third-party software (for instance, Microsoft Word or Adobe Photoshop) and then

added to the site by the contributor. The files that the contributor may add can be easily controlled by the designer or administrator.

In addition, there are several control and configuration files used to ensure that the site works as it should. These control files are described in other chapters of this guide.

2.2 Templates

Templates are used to arrange available site assets. They are all sections of HTML (or in the case of page templates, complete HTML pages) where the tags describing the related content are stored.

The three types of templates used are:

- [Page Templates](#)
- [Subtemplates](#)
- [Region Templates](#)

Templates allow the data to be placed in a certain manner. The definitions define which site assets are available to place on a template, as well as how the assets can display.

2.2.1 Page Templates

Page templates are the only templates that are complete HTML pages. Generally, the best use of Site Studio is to maximize reuse of assets, the page template should be looked at as a framework for the other templates; the subtemplates and region templates used to specifically align the content. Page templates and subtemplates additionally define the placement of the contribution regions.

Each section in the hierarchy can have - and typically will have - a page template assigned as the primary page for that node. The root section of the site hierarchy is where the home page of the Web site is located. Just as with the other sections in the hierarchy, the **Primary Page** and **Secondary Page** entries in section properties display the page templates used for the primary and secondary pages of the each second, including the root.

The data associating a page template with a primary or secondary page in a section is stored in the project page. Assigning a primary and secondary page is done in the properties pane.

Placeholders on a page template are ultimately placed using the `wcmPlaceholder` script extension. Placeholders themselves are not a site asset, placeholders are simply a defined area on a page template or subtemplate where you can use a placeholder definition to determine how content is reused in the specific placeholder.

2.2.2 Subtemplates

Subtemplates are, simply, page templates that do not have a `<HEAD>` section. They can contain a contribution region, and are most commonly used to break one contribution region on a page template into multiple parts.

Tags can be inserted on a subtemplate the same way they would be on a page template. However, any instance of reference to the `<HEAD>` tag on a subtemplate will generate unanticipated results. This is most common when using fragments on a subtemplate that have multiple snippets where one refers to the head.

2.2.3 Region Templates

Region templates help define where the elements and the associated content display. The region template is selected based on the region definition used.

Region templates are used to arrange the elements as they will display on the consumer page. The elements available to use are defined by the region definition. Not all elements available must be used, which allows for creating region templates that can have multiple layouts for the same named element definitions. Using the same element definitions by name, but a different data file, allows for the easy reuse of the site assets.

2.3 Definitions

Definitions are used to define which assets are available to use and how they can be used.

This section contains the following topics:

- ["Placeholder Definitions"](#) on page 2-3
- ["Region Definitions"](#) on page 2-4
- ["Element Definitions"](#) on page 2-6

2.3.1 Placeholder Definitions

The placeholder definition controls the allowed actions within the contribution region, including if contributors can modify data, if the metadata can be modified, if the associated data file can be switched or removed, and other actions.

The placeholder definition is associated to a placeholder in code (a tag), in a section property, the global mapping property, or as the default placeholder. When there is more than one listed association of a placeholder definition to a placeholder, then they are determined in that priority; that is, association in code takes precedence over the section property, which takes precedence over the global mapping property, which takes precedence over the default placeholder.

Example Code

The `<complexProperty name="flags">` is the collection of allowed (and disallowed) actions when the placeholder definition is used.

The `<mappings>` tag lists the allowed region definitions (using the `<regionDefinition>` tag) and their associated region templates (using the `<regionTemplate>` tag). Additionally, the default region template for a given region definition is coded within the `<regionTemplate>` tag. Similarly, the available subtemplates (if any) are listed under the `<subTemplates>` tag.

These are handled in the Designer UI through the Placeholder Definition dialog. The source code for definitions can be viewed in Designer by selecting the Source tab.

```
<?xml version="1.0" encoding="UTF-8"?>
<placeholderDefinition
  xmlns="http://www.oracle.com/sitestudio/PlaceholderDefinition/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oracle.com/sitestudio/PlaceholderDefinition/
  http://www.oracle.com/sitestudio/ss_placeholder_definition.xsd">
  <property name="description" value="Default placeholder definition" />
  <complexProperty name="flags">
    <property name="update" value="true" />
  </complexProperty>
</placeholderDefinition>
```

```

<property name="preview" value="true" />
<property name="reset" value="true" />
<property name="modifyMetadata" value="true" />
<property name="approve" value="true" />
<property name="reject" value="true" />
<property name="docInfo" value="true" />
<property name="switchDataFile" value="true" />
<property name="viewUsageReport" value="true" />
<property name="viewTrackerReport" value="true" />
<property name="docInfoUpdate" value="true" />
<property name="switchRegionTemplate" value="true" />
<property name="removeAssociation" value="false" />
</complexProperty>
< mappings>
  < regionDefinition location="regiondefinition_recipe">
    < regionTemplate location="regiontemplate_recipe" default="true" />
  </ regionDefinition>
  < regionDefinition location="regiondef_nativedoc">
    < regionTemplate location="regiontemplate_nativedocument" default="true" />
    < regionTemplate location="regiontemplate_recipe" />
  </ regionDefinition>
  < regionDefinition location="regiondefinition_basic">
    < regionTemplate location="regiontemplate_default" default="true" />
  </ regionDefinition>
</ mappings>
< subTemplates>
  < subTemplate location="subtemplate_right_left" />
  < subTemplate location="subtemplate_new_basic" />
</ subTemplates>
</ placeholderDefinition>

```

You can see in the section of XML that each placeholder definition associates region templates with a region definition, and additionally marks which region template is the default template for that region definition within the scope of the placeholder definition. It follows that if you use a different placeholder definition, a different association could be in place.

2.3.2 Region Definitions

The region definition is used to map the content (any content, including contributor data files, native documents, and so forth) through the placeholder definition to get to the appropriate region template.

The region definition used in an instance can be defined in one of two ways. First, the region definition can be explicitly called in the `wcmPlaceholder` tag when written in a page template or a subtemplate.

More commonly, the region definition is loaded based on the value of the metadata field `xRegionDefinition` for the data file used in the placeholder.

Example Code

This particular example is a region definition with five element definitions referenced. Each `<elementReference>` tag has additional `<property>` tags defining the value for the label, which will also display in Contributor when the region is opened for editing, and the value for the description, which is the tooltip text displayed when the contributor hovers the mouse over the label. For more information about Contributor, see the *User's Guide for Site Studio Contributor*.

The `<dataProperty name="metadata">` tag is the location for any exceptions to enabling metadata modification. This is done in the Designer UI through the Enable Metadata Modification dialog (see the *User's Guide for Site Studio Designer*).

The `<complexproperty name="switchregioncontent">` tag is the location for the content the contributor is allowed to access via the Switch Region Content dialog. This is done in the Designer UI through the Region Content Options dialog (see the *User's Guide for Site Studio Designer*).

```
<?xml version="1.0" encoding="UTF-8"?>
<regionDefinition xmlns="http://www.oracle.com/sitestudio/Element/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oracle.com/sitestudio/Element/
  http://www.oracle.com/sitestudio/ss_element_definition.xsd">
  <elements>
    <elementReference name="Title_textonly" location="element_plaintext_full">
      <property name="label" value="Title_textonly" />
      <property name="description" value="Title of recipe, using text only
        element" />
    </elementReference>
    <elementReference name="Image" location="element_image_min">
      <property name="label" value="Recipe_image" />
      <property name="description" value="Image associated with the recipe" />
    </elementReference>
    <elementReference name="Ingredients" location="element_wysiwyg_full">
      <property name="label" value="Ingredients" />
      <property name="description" value="" />
    </elementReference>
    <elementReference name="Directions" location="element_plaintext_full">
      <property name="label" value="Directions" />
      <property name="description" value="" />
    </elementReference>
    <elementReference name="DynamicList" location="element_dynlist_full">
      <property name="label" value="Dynamic List" />
      <property name="description" value="Dynamic List for toolbar purposes" />
    </elementReference>
    <elementReference name="StatList" location="element_staticlist_full">
      <property name="label" value="Statlist" />
      <property name="description" value="Static List for the toolbar" />
    </elementReference>
  </elements>
  <property name="description" value="Recipe Region" />
  <dataProperty name="metadata">
    <![CDATA[]]>
  </dataProperty>
  <complexProperty name="switchregioncontent">
    <property name="createnewxml" value="true" />
    <property name="createnewnative" value="true" />
    <property name="choosemanaged" value="true" />
    <property name="chooselocal" value="true" />
    <property name="choosenone" value="false" />
    <valueList name="createnewnativetypes">
      <value>.pptx</value>
      <value>.psd</value>
    </valueList>
    <complexProperty name="choosemanagedquerytext">
      <property name="corecontentonly" value="false" />
      <dataProperty name="querytext">
        <![CDATA[xWebsiteObjectType <Matches> `Data File` <OR>
          xWebsiteObjectType <Matches> `Native Document`]]>
      </dataProperty>
    </complexProperty>
  </complexProperty>
</regionDefinition>
```

```
</complexProperty>
<dataProperty name="defaultmetadata">
  <![CDATA[]]>
</dataProperty>
</complexProperty>
</regionDefinition>
```

2.3.3 Element Definitions

Each element definition, just as with other site assets, is a simple XML file.

The static list element definition files will have more code in the XML because it must list each element definition used within the static list.

Custom elements will not be any more complex than other element definitions, because custom elements simply load a separate form in HTML.

Example Code

The code for an element definition can vary widely depending on the type of element. All elements will contain a `<complexProperty name="flags">` section to describe the flags and their state. The flags used by an element through the element definition depend on the element.

Because of the variations in code for the element definitions, none are represented here. To see the differences in the code, open the element definitions in Designer.

2.4 Relationship Between Templates and Definitions

The idea is to make this section a technical description of the interactions between the template and the definition to control page layout and reusability.

All data must be tagged with a region definition. That is how it is placed with the appropriate definition and template combination.

2.5 Serving Web Pages

Since each asset is stored individually, the page is combined on the server before it is served to the client. The assets used to construct are selected based on the references in each other asset, as previously described.

The general process of events in creating the page happens in this manner after the request is made:

1. The page template is loaded based on the requested URL.

A specific page request will load the page template associated with that page, and a request for a folder or the root will load the page template named in the project file. That template's name is editable through the properties pane in Designer.

2. All assets not listed in a placeholder are loaded.
3. As the assets outside of the placeholder are loaded, the Idoc script is executed and the placeholder is filled, starting with the evaluation of the placeholder definition. The specific placeholder definition for the page is loaded in this order:
 - The placeholder definition explicitly named in the `wcmPlaceholder` tag.
 - If no placeholder definition is specifically listed in the tag, the definition listed in the section properties based on the URL is used.

- If the section properties lists no placeholder definition, then the definition used in the global definition mappings is used.
 - If there is no placeholder definition listed in the global definition mappings, then the placeholder listed in the Web site properties in the properties pane is used.
4. The data file used is based on the value listed in the **Primary** (or **Secondary**) **Page Params** in the section properties. The exception is when the data file to use is listed explicitly in the `wcmPlaceholder` tag.
 5. The **xRegionDefinition** metadata field for the data file lists which region definition to use. Again, this is when a region definition is not explicitly listed in the `wcmPlaceholder` tag.
 6. The region template and element definitions used are determined from the settings in the region definition, unless a different region template is stated in the `wcmPlaceholder` tag.
 7. Once all elements are collected, the page is assembled and served.

For more information on using `wcmPlaceholder`, see "[wcmPlaceholder](#)" on page 9-2.

2.6 Conversions Definitions

The conversions definitions, like the templates and definitions, are simple XML files used to reference the conversion rules.

Here is an example of the XML used in a conversions definition with a rule that is the default:

```
<?xml version="1.0" encoding="UTF-8"?>
<conversionsDefinition
  xmlns="http://www.oracle.com/sitestudio/ConversionsDefinition/"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.oracle.com/sitestudio/ConversionsDefinition/
  http://www.oracle.com/sitestudio/ss_conversions_definition.xsd">

  <conversion name="default" type="simple" key="s~"/>
</conversionsDefinition>
```

2.7 Contribution Mode

Contribution mode is the state a page is in when the contributor has it open for editing. The rest of the time, the page is considered to be in consumption mode.

This section covers the following topics:

- "[Key Command](#)" on page 2-7
- "[Query String](#)" on page 2-8
- "[Session Cookie](#)" on page 2-8
- "[Process from Consumption Mode to Contribution Mode](#)" on page 2-8

2.7.1 Key Command

The key command is the keystroke combination used to enter and exit contributor mode. This is set in the JavaScript file `wcm.toggle.js`. See "[wcm.toggle.js](#)" on page 5-4 for more information.

2.7.2 Query String

Setting `wcm.contributor.mode` within the query string will enable contributor mode or consumption mode, depending on the setting. This is most commonly done in the URL, as in this example:

```
http://www.example.com?wcm.contributor.mode=true
```

Setting it to `true` will enable contributor mode; setting it to `false` will enable consumption mode.

2.7.3 Session Cookie

A session cookie is set to persist the contributor mode setting when navigating within the same domain. The cookie is set and removed based upon the key command action or query string value.

2.7.4 Process from Consumption Mode to Contribution Mode

When a contributor opens a page for editing, the page passes through these states:

1. The user presses the key command (as set in `wcm.toggle.js`; default is Ctrl + Shift + F5).
2. The contribution mode session cookie is set.
3. The contribution mode query string value is removed from the URL (if present).
4. The browser requests the same page from the server using the newly calculated URL.
5. Server authentication; the contributor mode cookie is removed in the event of an authentication failure.
6. The browser loads the requested page.
7. During page load, an `OnKeyDown` event handler is applied to the HTML document object to detect the next contributor mode keyboard command.
8. During page load, the JavaScript code detects the contributor mode cookie. (Assuming that the contributor mode cookie was not removed by the server.)
9. The contribution mode is drawn in the browser at the HTML window `OnLoad` event.

Site Studio Metadata

This section covers the following topics:

- ["About Metadata"](#) on page 3-1
- ["Metadata Fields"](#) on page 3-1

3.1 About Metadata

Each Site Studio asset has information associated with it called metadata. Metadata is used by the Oracle Content Server to help you manage the multiple resources and site assets.

The metadata associated with the Site Studio assets you will create, edit, and use to construct your Web site are necessary for efficient storage as well as maintaining the relationship between assets. This is especially important with the methods used in Site Studio 11gR1, where each asset can be used and reused. The metadata fields help maintain the structure of which asset is used with other assets. The metadata is also used other things, such as relevant filtering during searches.

3.2 Metadata Fields

Five custom metadata fields, created by the Site Studio component, are required by the Site Studio product:

- ["xWebsiteObjectType"](#) on page 3-1
- ["xWebsiteSection"](#) on page 3-3
- ["xWebsites"](#) on page 3-3
- ["xDontShowInListsForWebsites"](#) on page 3-4
- ["xRegionDefinition"](#) on page 3-4

3.2.1 xWebsiteObjectType

The xWebsiteObjectType metadata field is used to indicate what type of web site-related item the managed document is. The field is an option list containing the following values for possible managed objects:

- **Data File:** Content files in XML format that are generated by Site Studio. Contributor data files are edited using the Site Studio Contributor application.
- **Page Template:** Fully-formed HTML files that define the layout and high-level look-and-feel of web pages, including the placement of contribution regions (that is, editable areas on the page), navigation aids (in the form of fragments) and

site-wide images (banners and the like). Page templates are the highest-level site design object.

- **Subtemplate:** Partial HTML files (that is, without head and body sections) that can be inserted into placeholders on page templates to divide them into further smaller, reusable areas with their own placeholders and contribution regions.
- **Region Template:** Partial HTML files (that is, without head and body sections) that define the layout and look-and-feel of the data in contribution regions within web pages.
- **Placeholder Definition:** Files that define what region definitions, region templates, and subtemplates are allowed for the associated placeholders. They also specify what contributor actions are allowed for the placeholders.
- **Region Definition:** Files that define the type of content that elements of a particular type consists of. They also specify the content creation and switching options available to contributors for contribution regions, and set default metadata for content files associated with these regions.
- **Element Definition:** Files that define the editing experience for element types. Specifically, they specify what a contributor can do when editing an element.
- **Native Document:** Content files created using familiar third-party applications such as Microsoft Word. Native documents are converted to HTML format using Dynamic Converter, and they are edited using their associated application.
- **Fragment:** Chunks of code that enhance the functionality of a Site Studio Web site (for example, by providing dynamic navigation aids or a standard page footer).
- **Image:** Graphic files (JPG, GIF, PNG) that are included in content files or page templates (for example, corporate banners or product images).
- **Script:** JavaScript files that provide a series of commands that can be executed without user interaction. Scripts are often used to provide additional functionality to web pages.
- **Stylesheet:** Cascading style sheet (CSS) files that provide control over how page content is displayed (more specifically, how different HTML elements, such as headers and links, appear on the page). Links to CSS files are often embedded in page templates, so their formatting rules apply to all web pages based on these templates.
- **Project:** XML files that store all information about a Site Studio Web site that Designer needs to work with the site, such as the site hierarchy, site section properties, data associations, placeholder mappings, and so on
- **Custom Element Form:** HTML files that define custom forms for use in elements (for example, selection forms for specific file types). Site Studio comes with several predefined custom element forms (in `[CS-Dir]\custom\SiteStudio\elementforms`).
- **Custom Configuration Script:** JavaScript files that override the default Contributor editor configuration to provide contributors with a customized editing experience.
- **Validation Script:** JavaScript files that define the validation rules for element data to determine that the data meets the requirements (for example, it does not exceed a certain maximum length or contain some illegal characters).
- **Manager Settings:** Files that define the functionality that is available in Site Studio Manager. Manager is a web-based tool that allows designated users (site managers) to modify the structure of a Web site.

- **Conversions Definition:** Files that specify the conversion rules for native documents on a Web site.
- **Other:** Any other media files that could be used on a Web site, such as Flash animations, video files, audio files, and so on.

See the *User's Guide for Site Studio Designer* for detailed explanations of each of these file types.

3.2.2 xWebsiteSection

The xWebsiteSection field is used to determine which web site section should be used to display a managed item if a link to that item is generated but does not explicitly include a target section already. This is primarily used for contributor data files and native documents. Internally, it contains a *siteId:nodeId* value, and the Site Studio component overrides the standard content server pages to provide a more friendly user interface for picking a site and section.

The contents of this field become important when understanding the different URL formats available in Site Studio. These are described in detail later in this document.

Note: This field replaces the use of the folder-based xCollectionID field used in Site Studio versions before 7.2. When Site Studio is installed, the new xWebsiteSection field will be initialized with the existing values from the xCollectionID field, but only for those documents that exist in a web site related folder (for example, those documents that were part of an earlier Site Studio web site and filed in one of the folders for that web site).

3.2.3 xWebsites

The xWebsites field is used to determine which web site (in the content server) the managed document belongs to. Internally, it is a comma-separated list of site identifiers, and the Site Studio component overrides the standard content server pages to provide an easier-to-use list of site names.

Each Web site has its own ID. When an action is performed within either the Designer or Contributor application that involves a managed document (typically adding or editing an asset), the current site identifier is automatically appended to the xWebsites field for that managed document, if it did not already exist. This means that when you use any managed document within a site in the Designer or Contributor application, that managed document will automatically become part of the site.

It is important to realize that the site identifier will never be automatically removed from this field once it has been added because it is currently impossible for Site Studio to know all of the places that a managed document might be referenced from. Designers can use the site assets pane within the Designer application to manually add and remove managed items to and from their web site.

Note: The xWebsites field replaces the xWebsiteID field, which was used in Site Studio in versions prior to Site Studio version 7.2. If xWebsiteID exists when Site Studio is installed, the new xWebsites field will be initialized with the existing values from the xWebsiteID field. The xWebsiteID field will not be removed and will still behave as it did before in order to maintain backward compatibility with custom fragments created in earlier versions of Site Studio.

3.2.4 xDontShowInListsForWebsites

The xDontShowInListsForWebsites field lists the Web sites for which a contributor has specified, through the user interface, that a contributor data file or native document should not display in dynamic lists on the Web site. This field allows the "Include/Exclude in Dynamic List" feature to work properly.

When a contributor excludes a file from a dynamic list, the Web site ID is added to this value. If the contributor later re-includes the content in dynamic lists for the Web site, the Web site ID is removed from this metadata field and the content becomes available to the dynamic lists again.

Note: If a Web site value displays in this metadata field for a particular data file or native document, then that piece of content will not display in any lists on the site; however, it will still display in search results for the site.

3.2.5 xRegionDefinition

The xRegionDefinition metadata field specifies the region definition that a contributor data file is associated with. A data file can be associated with only one region definition, but a region definition may be associated with many data files.

Region definitions define the types of content used on a Web site. They could be thought of as 'content classes'. They are essentially groups of individual elements which define the various chunks of reusable information for a particular site content type. For example, there could be a region definition ('content class') called "Press-Release," which consists of the elements Title, Subtitle, Intro-Text, Body-Text, and Image. Contributor data files are associated with a region definition to store the data for each element in the region definition. (What a contributor can do with the data is controlled by element definitions.)

In addition to defining site content types in terms of its constituent parts (elements), region definitions also specify the content creation and switching options available to contributors for its associated contribution region(s). For example, if a contribution region is set up to allow contributors to switch the content of that region, they might be allowed to use existing contributor data files on the server only (not native documents or new contributor data files). (Please note that placeholder definitions control whether contributors can actually switch content in contribution regions.) Finally, region definitions also set the default metadata for content in contribution regions as it is checked into the content server.

See the *User's Guide for Site Studio Designer* for more information about region definitions.

Link Formats

This section covers the following topics:

- ["About Link Formats"](#) on page 4-1
- ["Using Path-Based Links"](#) on page 4-1
- ["Using wcmUrl"](#) on page 4-2
- ["Using Server-Side Script Links"](#) on page 4-3
- ["Using Token Links"](#) on page 4-5
- ["Using JavaScript Links"](#) on page 4-7

4.1 About Link Formats

There are many ways to make a link in Site Studio. You use the expected HTML path-based link, as well as some Site Studio extensions that maximize the portability and reusability of the architecture in Site Studio 11g.

Within the Site Studio-specific methods, there are three main types of links. There are links to a content file (such as a data file or a native document), links to a node, and links to a static resource, such as an image.

4.2 Using Path-Based Links

A path-based link is simply a link using a path. You can use Idoc Script variables to replace a section of the path if you wish. The variable will be evaluated by the server and replaced in the served HTML.

Path-based links can be written as absolute or relative. Relative links, as well as links using an Idoc variable, are recommended because they are more portable. Absolute links are very easily broken when the site is modified. The only time an absolute link is recommended is when linking to an outside website.

When a path-based link is created using the Link Wizard, the portion of the path from the server to the Web site root is replaced with an Idoc variable.

Examples

The basic path-based link is a full URL just as expected:

```
http://www.oracle.com/support/index.html
```

The same link created using an Idoc variable (specifically `ssServerRelativeSiteRoot`) would look like this:

```
<!--$ssServerRelativeSiteRoot-->support/index.html
<a href=" [!-$ssServerRelativeSiteRoot--]support/index.html ">
```

Note that the content in the variable includes the final "/" in the path segment.

Relative paths are used as they would be in any static HTML instance:

```
../../support/index.html
<a href="../../support/index.html ">
```

In some examples, you may see the angle bracket used (for example, `<!--$wcmUrl('nodelink', '30')-->`), and in other examples, you may see the square bracket used (`[!--$wcmUrl('nodelink', '30')--]`). Site Studio writes the square bracket to XML files and other places where an angle bracket might cause escaping problems, and it writes the square bracket to templates. You should know that they can be used interchangeably; however you should keep in mind which instance might cause a problem with escaping when you choose to use one or the other.

4.3 Using wcmUrl

This is a script-extension designed to encapsulate the other link forms, specifically the server-side `ssNodeLink`, `ssLink`, and `ssWebLayoutUrl`. This is the recommended link format to use with Site Studio to make the most of its use.

It only requires one parameter, the type. The other parameters may or may not be required depending on the type specified.

The `wcmUrl` script extension is a server-side script link that incorporates the functionality of the script links listed in "Using Server-Side Script Links" on page 4-3. Maintaining them all using one script extension allows not only for use with Oracle Web Content Management, but also makes it easier to search out the links in a content file or template.

Parameters

- `type`: the type of link, entered as one of the following.
[`nodelink` | `link` | `resource`]
`Nodelink` is used to create links to sections, which by default end with `index.htm`.
`Link` is used to create links to content, which by default end with `<contentId>`.
`Resource` is used to link to weblayout static resources, such as images.
- `nodeId`: the ID of the node referenced. Used only when the type is set to either `nodelink` or `link`.
- `siteId`: the site ID of the Web site referenced. Used only when the type is set to either `nodelink` or `link`.
- `dDocName`: the `dDocName` of the content referenced. Used only when the type is set to either `link` or `resource`.
- `Xpath`: path to the content referenced. Used only when the type is set to either `link` or `resource`.

Examples

Before evaluation:

```
[!--$wcmUrl('link', 'switched_region_CDF')--]
[!--$wcmUrl('nodelink', '30')--]
[!--$wcmUrl('resource' 'groups/public/documents/adacct/mydocname.jpg')--]
```


These examples use single quotes for the parameters. Other examples may show double quotes. Both work, however, just as with the angle bracket and square bracket, there may be some instances where using one instead of the other will help prevent possible escaping problems.

Evaluated by the server:

```
<a href="http://YourUrl.com/Website/support/CRM/switch.html"
<a href="http://YourUrl.com/Website/region/West/">
<a href="http://YourUrl.com/Website/groups/public/documents/adacct/mydocname.jpg">
```

The first line shows a link to a content file referenced by the *dDocName*, the second is a link to a node referenced by the *nodeId*, and the third is a link to the *XPath* of an image file on the Oracle Content Server.

If a value for a link evaluates as bad, such as when it is linked to a resource is no longer there, then the link evaluates as a token link. For more information, see "[Token Links as Returned Server-Side Links](#)" on page 4-6.

4.4 Using Server-Side Script Links

A server-side script link is a link that is written in script, but is then evaluated by the server and replaced in the displayed HTML. If someone chooses to view the source of the displayed page, they will only see the URL (either a relative or a full URL).

The server-side script links are written in Idoc Script. Whether it is a variable or a function call, the server-side link evaluates on the server and you will never see it in the displayed Web page's source.

Contains these topics:

- "[ssLink](#)" on page 4-3
- "[ssNodeLink](#)" on page 4-4
- "[ssWebLayoutUrl](#)" on page 4-4

4.4.1 ssLink

The `ssLink` script link is used to create a server-side link to a piece of data, native document, or a data file. The target object is referenced by the `dDocName`.

Parameters

- `dDocName`: the `dDocName` of the managed item. If only one parameter is specified, it is assumed to be the `dDocName`.
- `targetNodeId`: a unique identifier for a node to use as the target context (optional).
- `targetSiteId`: the unique identifier for the Web site to use as the target context (optional). If you do wish to specify a `targetSiteId`, you must also specify the `targetNodeId`.

Examples

Before evaluation:

```
[!--$ssLINK('dDocName')--]
```

Evaluated by the server:

```
<a href="http://YourUrl.com/Website/section/subsection/content/index.htm">
```

Note: If the link generated by `ssLink` is bad, then a tokenized link using `ssLINK` will be used in its place. This allows the page to fully generate, but the URL will be a token link (which still fails).

4.4.2 `ssNodeLink`

The `ssNodeLink` script link is used to create a server-side link to a Web site section.

Parameters

- `nodeId`: a unique identifier for a node.
- `siteId`: a unique identifier for the Web site containing the node (optional).

Examples

Before evaluation:

```
[!--$ssNodeLink('30')--]
```

(where the `/support/CRM` folder on the Web site has a `nodeId` of 30)

Evaluated by the server:

```
<a href="http://YourUrl.com/Website/support/CRM">
```

Note: If the link generated by `ssNodeLink` is bad, then a tokenized link using `ssNODELINK` will be used in its place. This allows the page to fully generate, but the URL will be a token link (which still fails).

4.4.3 `ssWebLayoutUrl`

The `ssWebLayoutUrl` is used to create a server-side link to a weblayout rendition of the document. This resource/rendition can be referenced by either the `dDocName` or the path to the resource.

Parameters

- `dDocName`: the `dDocName` of the content referenced.
- `Xpath`: path to the content referenced.

Examples

Before evaluation:

```
[!--$ssWebLayoutUrl('groups/public/documents/adacct/mydocname.jpg')--]
```

Evaluated by the server:

```
<a href="http://YourUrl.com/Website/groups/public/documents/adacct/mydocname.jpg">
```

4.5 Using Token Links

Token links (also known as late-resolving links) are not resolved until they are actually clicked. View source would reveal a URL that has `ssLINK` or `ssNODELINK` in the URL. Token links are not resolved until they are clicked (thus the term "late-resolving," as the link resolution happens after the page is served). By comparison, the script links are resolved on the server before serving the Web page.

You should be aware that `ssLINK` and `ssNODELINK` are token links; although `ssLink` and `ssNodeLink` have the same name as `ssLINK` and `ssNODELINK`, the camel case `ssLink` and `ssNodeLink` are script links.

Token links are a recommended way of including a link in a native document. Because the Word doc goes through Dynamic Converter and converts to HTML, it is much easier to write them with token links. It is possible otherwise, but this way is simpler.

This section contains these topics:

- ["ssLINK" on page 4-5](#)
- ["ssNODELINK" on page 4-6](#)
- ["Token Links as Returned Server-Side Links" on page 4-6](#)

4.5.1 ssLINK

Used to create a late-resolving link to a piece of data, a native document, or a data file. Unlike `ssLink`, the late-resolving link is not resolved until after the Web page is constructed and served to the browser.

Parameters

- `dDocName`: the `dDocName` of the managed item. If only one parameter is specified, it is assumed to be the `dDocName`.
- `targetNodeId`: a unique identifier for a node to use as the target context (optional).
- `targetSiteId`: the unique identifier for the Web site to use as the target context (optional). If you do wish to specify a `targetSiteId`, you must also specify the `targetNodeId`.

Examples

Before evaluation:

```
[!--$ssLINK('dDocName')--]
```

Evaluated by the server:

```
<a href="http://YourUrl.com/Website/ssLINK/dDocName">
```

Note that the `ssLINK` is still in the URL. It will be visible this way to any browser, either by hovering on the link or by viewing the source. Once the link is clicked, the Oracle Content Server will evaluate the `dDocName` and load the appropriate data file.

4.5.2 ssNODELINK

Used to create a late-resolving link to a Web site section.

Parameters

- `nodeId`: a unique identifier for a node. If only one parameter is specified, it is assumed to be the `nodeId`.
- `siteId`: a unique identifier for the Web site containing the node (optional). If you specify a `siteId`, you must specify a `nodeId`.

Examples

Before evaluation:

```
[!--$ssNODELINK('10027')--]
```

Evaluated by the server:

```
<a href="http://YourUrl.com/Website/ssNODELINK/10027">
```

Note that the `ssNODELINK` is still in the URL. It will be visible this way to any browser, either by hovering on the link or by viewing the source. Once the link is clicked, the Oracle Content Server will evaluate the `nodeId` and load the appropriate Web site section.

4.5.3 Token Links as Returned Server-Side Links

Token links are returned in situations where the server-side link (for example, `ssNodeLink`) fails. The replacement is done to ensure that the page delivery is not hampered by a bad link.

When the server evaluates the server-side link, and the link is faulty (for example, the `dDocName` is bad), then the server returns the link parameter value with the token link. The result is that the Web pages are still delivered without substantial errors, however, the users will then see the token link in the source (or by hovering their pointer on the link), just as they would if a token link had normally been used.

Example

The parameter value for the passed server-side link is retained. If the following is passed to the server to evaluate:

```
<!--$ssLink('Bad_dDocName')-->
```

then the server will return, in the source of the served Web page

```
<a href="http://YourUrl.com/Website/ssLINK/Bad_dDocName">
```

This prevents the server from erroring out the entire Web page when trying to evaluate the link. Clicking the returned tokenized link, however, will display an error page from the server.

This will also happen for any `wcmUrl` script errors, as in these cases the `wcmUrl` is a thin wrapper for the server-side script link.

4.6 Using JavaScript Links

It is possible to use JavaScript to enter a link in a site asset. This is not recommended, but is still available as a possible method.

The links are used just as you would use `ssLINK` or `ssNODELINK`, with the same parameters. And like `ssLINK` and `ssNODELINK`, JavaScript links are late-evaluated. The only difference is in explicitly calling them as JavaScript.

Examples

```
javascript:link('dDocName')  
javascript:nodelink('nodeId')
```

Site Studio Runtime Generated Files

This section covers the following topics:

- ["About Runtime Generated Files"](#) on page 5-1
- ["Runtime Auto-Generated Files"](#) on page 5-1
- ["wcm.toggle.js"](#) on page 5-4

5.1 About Runtime Generated Files

Site Studio depends on a collection of runtime files to deliver a fully-functioning web site. When you change the site hierarchy of a web site, these files are affected, which is why you are prompted to update the runtime files when you change the site hierarchy in Designer.

The runtime files are stored in the runtime folder for the web site (where *<cs_name>* is the name of your content server and *<siteid>* is your web site):

```
<cs_name>\weblayout\websites\<siteid>
```

5.2 Runtime Auto-Generated Files

The following files are automatically generated:

- ["sitenavigation.js"](#) on page 5-1
- ["sitenavigationfunctions.js"](#) on page 5-3
- ["sitenavigation.xml"](#) on page 5-3
- ["sitenavigation.hda"](#) on page 5-4
- ["sitenavigation_co.hda"](#) on page 5-4

5.2.1 sitenavigation.js

The `sitenavigation.js` file contains the necessary JavaScript to define the web site hierarchy. Some of the navigation fragments in Site Studio were designed to read the information from this file and dynamically generate a navigation scheme for the web site using client-side JavaScript.

`Sitenavigation.js` contains the `NavNode` object definition, which represents a single node (also referred to as a "section" in the Designer interface) of the site hierarchy. It has properties that are generated from the true section properties stored in the web site project file.

Node property/method	Definition
m_parent	Contains the parent of the current node.
m_level	Contains the level for the current node (where 0 is the root).
m_id	Contains the node ID for the current node.
m_label	Contains the node label for the current node.
m_href	Contains the server relative path to this node in the site hierarchy.
m_subNodes	Contains the array of child nodes.
addNode()	Can be used to add a new child node.

In addition to these standard properties, the NavNode object also contains a member variable for every custom section property that has been assigned a value for this section. The naming convention for these variables is "cp_XXX" (where XXX is the custom section property name). These data members are constructed by parsing the additional parameters passed to a NavNode constructor; the format for the additional parameters are "name==value" strings (see definition of g_navNode_0_0 in sample code below). These additional parameters are automatically generated by Site Studio when you regenerate your runtime JavaScript files.

The sitenavigation.js file also contains the declaration for the active site hierarchy in terms of NavNode objects, which are defined from a single root node using a well-known name (g_navNode_Root). This value is then made available to the navigation fragments to examine, as appropriate.

Here is an example:

```
var g_navNode_Root = new NavNode('9001','Home',ssUrlPrefix +
'index.htm',null,'PageTitle==Ravenna Hosting Tutorial Site');
g_navNode_1=g_navNode_Root.addNode('9002','Products',ssUrlPrefix +
'Products/index.htm','PageTitle==Ravenna Hosting Products');
g_navNode_1_0=g_navNode_1.addNode('9003','Servers',ssUrlPrefix +
'Products/Servers/index.htm','MainNavIcon==/idcm1/groups/public/documents/rvh_
image/rvh_navicon_1.gif','PageTitle==Ravenna Hosting Servers');
g_navNode_1_0_0=g_navNode_1_0.addNode('9004','Web Servers',ssUrlPrefix +
'Products/Servers/WebServers/index.htm','MainNavIcon==/idcm1/groups/public/documen
ts/rvh_image/rvh_navicon_2.gif','PageTitle==Ravenna Hosting Web
Servers','SidebarProductsListBanner==webservers');
g_navNode_1_0_1=g_navNode_1_0.addNode('9005','Database Servers',ssUrlPrefix +
'Products/Servers/DatabaseServers/index.htm','MainNavIcon==/idcm1/groups/public/do
cuments/rvh_image/rvh_navicon_3.gif','PageTitle==Ravenna Hosting Database
Servers','SidebarProductsListBanner==databaseservers');
g_navNode_1_0_2=g_navNode_1_0.addNode('9006','Application Servers',ssUrlPrefix +
'Products/Servers/ApplicationServers/index.htm','MainNavIcon==/idcm1/groups/public
/documents/rvh_image/rvh_navicon_4.gif','PageTitle==Ravenna Hosting Application
Servers','SidebarProductsListBanner==application servers');
g_navNode_1_0_3=g_navNode_1_0.addNode('9007','File Servers',ssUrlPrefix +
'Products/Servers/FileServers/index.htm','MainNavIcon==/idcm1/groups/public/docume
nts/rvh_image/rvh_navicon_5.gif','PageTitle==Ravenna Hosting File
Servers','SidebarProductsListBanner==fileservers');
g_navNode_1_0_4=g_navNode_1_0.addNode('9008','Mail Servers',ssUrlPrefix +
'Products/Servers/MailServers/index.htm','MainNavIcon==/idcm1/groups/public/docume
nts/rvh_image/rvh_navicon_6.gif','PageTitle==Ravenna Hosting Mail
Servers','SidebarProductsListBanner==mailservers');
```


The NavNode definition and the JavaScript methods in the sitenavigation.js file are obtained from the Site Studio component resources each time the runtime files are regenerated. Only the hierarchy of NavNode objects is generated dynamically. If you want to change any of the other JavaScript, you need to update or override the component resources.

5.2.2 sitenavigationfunctions.js

The sitenavigationfunctions.js file provides methods used for client-side JavaScript navigation mechanisms. The definitions for the client-side ID-based hyperlink functions, link() and nodelink() methods, for instance, are both stored in this file.

5.2.3 sitenavigation.xml

The sitenavigation.xml file contains an XML definition of the active site hierarchy, which can be made available to server-side script. Some navigation fragments are built to read the information provided by this file. They then use the information to dynamically generate the navigation scheme for the web site with server-side script instead of client-side JavaScript.

The XML definition has a single <site> tag as the root. This tag contains a hierarchy of <section> tags to define each section.

Here is an example:

```
<site id="9001" level="0" parent="" label="Home" href="index.htm"
PageTitle="Ravenna Hosting Tutorial Site">
<section id="9002" level="1" label="Products" href="Products/index.htm"
PageTitle="Ravenna Hosting Products">
<section id="9003" level="2" label="Servers" href="Products/Servers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_1.gif"
PageTitle="Ravenna Hosting Servers">
<section id="9004" level="3" label="Web Servers"
href="Products/Servers/WebServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_2.gif"
PageTitle="Ravenna Hosting Web Servers"
SidebarProductsListBanner="webservers"></section>
<section id="9005" level="3" label="Database Servers"
href="Products/Servers/DatabaseServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_3.gif"
PageTitle="Ravenna Hosting Database Servers"
SidebarProductsListBanner="databaseservers"></section>
<section id="9006" level="3" label="Application Servers"
href="Products/Servers/ApplicationServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_4.gif"
PageTitle="Ravenna Hosting Application Servers"
SidebarProductsListBanner="applicationservers"></section>
<section id="9007" level="3" label="File Servers"
href="Products/Servers/FileServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_5.gif"
PageTitle="Ravenna Hosting File Servers"
SidebarProductsListBanner="fileservers"></section>
<section id="9008" level="3" label="Mail Servers"
href="Products/Servers/MailServers/index.htm"
MainNavIcon="/idcm1/groups/public/documents/rvh_image/rvh_navicon_6.gif"
PageTitle="Ravenna Hosting Mail Servers"
SidebarProductsListBanner="mailservers"></section></section>
```

5.2.4 sitenavigation.hda

The sitenavigation.hda file contains a persistent representation of the SiteStudioNavNodes ResultSet that can be made available with the ssLoadSiteNavResultSet() script extension. It contains a definition of the active site hierarchy, which can be made available to server-side script. Some navigation fragments are built to read the information provided by this ResultSet. They then use the information to dynamically generate the navigation scheme for the web site with server-side script instead of client-side JavaScript.

The "SiteStudioNavNodes" ResultSet has five columns:

- **nodeId**: the unique identifier for the node.
- **parentNodeId**: the unique identifier for the parent node.
- **label**: the label for the node.
- **level**: the depth of the node in the site hierarchy; the root section has a level of 0.
- **href**: the site relative path-based URL to the node's primary page.

5.2.5 sitenavigation_co.hda

The sitenavigation_co.hda file contains the same structure as the sitenavigation.hda file, but the "SiteStudioNavNodes" result set also includes contributor-only nodes.

5.3 wcm.toggle.js

The wcm.toggle.js file contains the JavaScript necessary to provide contribution functionality on a web site. Most significantly, the `wcm.contributor.OnKeyDown()` function, which is where the keyboard sequence to enter contribution mode (**Ctrl + Shift + F5**) can be changed to another sequence, if desired.

To change the default keystroke combination, perform these tasks:

1. Browse to the following directory (where *[CS-Dir]* is the installation location of your content server):

```
[CS-Dir]\custom\SiteStudio\publish\resources\wcm\sitestudio\
```
2. Open *wcm.toggle.js* in a text editor.
3. Locate the function `OnKeyDown`.
4. Change the implementation of this function to use a different keystroke combination that will call `wcm.contributor.toggle`.

This function uses virtual key codes to determine the key combination entered by the user. The default value is **Ctrl+Shift+F5**. The **F5** key has a virtual key code of 116 (or 0x74 in hexadecimal). The codes for the other typical function keys, **F1** through **F12** are 112 (0x70) through 123 (0x7B), respectively.

5. Save and close *wcm.toggle.js*.

Note: The next time you upgrade Site Studio or install a patch, you may need to perform these steps again to retain your keystroke combination.

Note: The key codes used to determine the keystrokes should be given special consideration in instances where contributors may use different operating systems, since the virtual key codes may vary among operating systems.

This section covers the following topics:

- ["About Fragments"](#) on page 6-1
- ["Fragment Libraries"](#) on page 6-2
- ["Read-Only Fragment Libraries"](#) on page 6-3
- ["Fragment Snippets and the `ssIncludeXml\(\)`"](#) on page 6-4
- ["Fragments That Use Custom Section Properties"](#) on page 6-4
- ["The Fragment Definition File"](#) on page 6-5
- ["Fragment Instance Structure in the `<ssinfo>` XML Data Island"](#) on page 6-13

6.1 About Fragments

A fragment is a self-contained snippet of HTML or script (including client-side JavaScript and server-side Idoc) that may have value in being reused in more than one template on a web site.

A fragment may contain references to other files, too, with tags like ``, `<SCRIPT SRC=xxx>`, and `<$ docLoadResource(xxx) $>`. Externally referenced files or resources serve as a fragment asset and therefore need to be made available whenever the fragment is used.

Simple fragments may contain atomic content that can be inserted anywhere in a template. More complex fragments require some content to be placed in the `<head>` of the page and other content to be placed in the `<body>`. In the `<body>`, content may be placed at the top of the page, the bottom of the page, or at the cursor's current position. A fragment, therefore, may contain multiple fragment snippets.

While it is possible to include multiple `<body>` snippets with a `<head>` snippet, this is discouraged. Primarily because fragments that use the `<head>` of a page cannot be placed in a region template or a subtemplate. This is because neither region templates nor subtemplates are full HTML pages, and do not contain a `<head>`. Fragments should be created using just the `<body>` to best fit with the flexible nature of Site Studio architecture. Using just a `<body>` would allow the fragment to be used on page templates, subtemplates, and region templates.

A page template does contain a `<head>`, and fragments can have both a `<head>` and `<body>` when used in a fragment. However, these fragments will create an `<ssinfo>` XML data island. See ["Fragment Instance Structure in the `<ssinfo>` XML Data Island"](#) on page 6-13.

The designer may want a fragment to take on a different look and feel or behavior depending on where it is used on the web site. This functionality is made available by the use of fragment parameters. With fragment parameters, the creator of a fragment can specify certain variable parameters, and the site designer can choose from these parameters when the fragment is actually added to a template.

From a high level, fragments contain:

- Zero or more fragment parameter definitions (name, type, default value) that are referenced in the fragment snippets but declared uniquely on each page that includes the fragment instance.
- One or more fragment snippets, which are pieces of HTML or script along with an identifier to indicate where that content should be placed when the fragment is added to a template.
- Zero or more fragment assets, which are the local files referenced by a fragment that will be made available to the fragment when it is used on a web page.

It is common for a fragment to consist of two snippets. The first, usually found in the <head> of the page, may reference a CSS file, which will format the page, or a JavaScript file, which will provide some or all of the fragment's implementation. The second, usually found in the <body> of the page at the drop-point, contains the presentation for the fragment. It may be as simple as a JavaScript call of a method provided by the included .js file, or it may contain a collection of HTML, JavaScript, and Idoc Script.

6.2 Fragment Libraries

Site Studio stores individual fragments in fragment libraries. You can store each fragment in its own library or store related fragments together in the same library. Fragment libraries are stored as managed objects in the content server in the following way:

- **Primary file:** a zip file containing all of the assets required by each fragment in the fragment library.
- **Alternate file:** a fragment definition file that defines the entire structure of each fragment in the fragment library.

Oracle Content Server manages content items as either primary or alternate files, or both (see Oracle Content Server Help). These files are unrelated to, and should not be confused with, primary and secondary pages in Site Studio.

Note: The fragment asset zip file that is checked in as the Primary file above should not be confused with a fragment library zip file that contains both the fragment assets and the fragment definition file. The fragment library zip file is what is used with the Designer's Upload and Download fragment library utilities and provides a simpler way to manage fragment libraries. You will only need to manage fragment asset zip files if you are accessing the managed fragment libraries directly in the content server instead of using the Designer's fragment management utilities.

The fragment definition file contains the root element <fragments>, which includes one or more <fragment> elements to define each fragment in the library. The exact syntax of the fragment definition file is described in the Fragment Definition File. For more information, see "[The Fragment Definition File](#)" on page 6-5.

To add a fragment library to the content server, you won't use the standard content server check-in page. Instead, you use the *"Upload Fragment Library"* feature in Designer. This feature checks in the managed content items and ensures that a copy of the fragment asset zip file is extracted to the appropriate runtime weblayout directory (where `<CS_name>` is the name of your content server):

```
<CS_name>\weblayout\fragments
```

This path can then be referenced by server-side Idoc Script in a fragment by using one of two Idoc variables:

- `HttpFragmentsRoot`: the full HTTP path to the fragments folder.
- `HttpRelativeFragmentsRoot`: the relative HTTP path to the fragments folder.

6.3 Read-Only Fragment Libraries

The root `<fragments>` element within the fragment definition files (sample fragments) that ship with the product are given a read-only attribute to prevent them from being edited or erased within the Designer application. There is no GUI exposed to set or clear this attribute: it is simply intended to avoid changes being made to the out-of-the-box fragments because they will be overridden when upgrading to future versions of Site Studio.

Designers, of course, can copy and edit these fragments and make modifications to their own copy.

6.4 Fragment Inclusion Using wcmFragment

While the Toolbox allows for quick and simple drag-and-drop placement of fragments on a template, some prefer to use the scripting method to place a fragment directly in the source.

The `wcmFragment` script is used on templates to add a fragment. If the tag is used on a region template or subtemplate, only the first drop-point snippet will be used.

If the tag is used on a page template, then a legacy `<ssinfo>` XML data island is inserted by Site Studio. For more information, see ["Fragment Instance Structure in the <ssinfo> XML Data Island"](#) on page 6-13.

Parameters

- `fragmentInstanceId`: ID of the fragment instance on the page. Used to differentiate fragments that have multiple pages of returns, such as dynamic lists. The ID allows the dynamic lists to display different 'pages' of query results. For instance, one fragment can be on page 1 of its query results, while the other can display page 3 of its own results. If the same ID is used for both fragments, then clicking to a particular page of results for one query will cause the other fragment to display the same page number of its own results.
- `fragmentDocName`: dDocName of the fragment library.
- `fragmentId`: ID of fragment within the named library.
- `snippetId`: ID of snippet within the named fragment.
- `tagProperties` The named value pairs of properties of the fragment.

Code Example

```
<!--$wcmFragment("fragmentInstanceID", "fragmentDocName", "fragmentID",
```

```
"snippetID", "ssTheme=default", "ssHoverColor=", "ssTextColor=", "ssFocusColor=",
"ssShowHome=true", "ssShowNext=false", "ssClassName=IDocNavTabsTop")-->
```

6.5 Fragment Snippets and the `ssIncludeXml()`

Each fragment contains a snippet defined by the fragment definition file for the fragment library. A fragment snippet can be included in a fragment in one of three ways:

- **simple:** the fragment snippet is added directly to the layout page with no additional markup and is no longer recognized or managed as a fragment.
- **inline:** the fragment snippet is added directly to the layout page but with special markup surrounding it so that it can be recognized and managed as a fragment.
- **reference:** the fragment snippet is not actually added to the layout page; rather, a reference to the snippet is added (much like an include file) along with surrounding markup so that it can be recognized and managed as a fragment.

It is highly recommended that you use *reference* as the include mechanism for all but the most trivial of snippets. This way you can manage the snippet content in a single place even if the fragment is used many times across the site.

The exact syntax for the special markup surrounding inline and reference snippets can be found in "[ssIncludeXml](#)" on page 9-8. For snippets included by reference, the snippet is added to the layout page using a script extension called `ssIncludeXml()`.

This script extension provides an Idoc mechanism for including elements from a managed XML file and placing them in a layout page. The parameters for the `ssIncludeXml()` include the `dDocName` of the fragment definition file of the fragment library and an `XPath` expression for the XML node to be extracted. (More parameters are explained in "[ssIncludeXml](#)" on page 9-8). The content of the XML node extracted is further evaluated in the scope of the current template and therefore can include additional server-side Idoc Script, if necessary.

6.6 Fragments That Use Custom Section Properties

Custom section properties can be defined by the Designer and unique values can be assigned to each property for each section of the web site (see the *User's Guide for Site Studio Designer*). The definitions and the values are stored in the web site project file.

By themselves, custom section properties are useless. Only when a custom property is referenced by client-side or server-side script within a layout page (or more typically, within a fragment snippet), does it become useful. There are two primary ways to access custom section property values: client-side JavaScript and server-side Idoc Script.

6.6.1 Client-Side JavaScript

The client-side runtime generated file `sitenavigation.js` contains an array of `NavNode` objects containing a definition for the current web site hierarchy (See "[sitenavigation.js](#)" on page 5-1). Each custom section property that has a value for the current section will be contained within the `NavNode` object for that section in a member variable whose name begins with `cp_`.

The most typical use of these client-side representations of the custom section properties is within navigation fragments. When a navigation fragment is iterating through the `NavNode` objects, it can detect the existence of the `cp_XXX` member variables and use them to customize the navigation scheme being displayed. Since the

`cp_XXX` member variable may or may not exist, accessing the parameter is made easier through the use of two JavaScript methods provided by Site Studio:

- `customSectionPropertyExists (prop)`: returns true or false to indicate whether the custom section property exists; if this method returns true, you can use the custom section property directly.
- `getCustomSectionProperty (prop)`: returns the value for the custom section property, if it exists; returns an empty string otherwise.

You can see these two methods in action in the CSP Sample Navigation (client) fragment that ships with Site Studio (for more information, see the *User's Guide for Site Studio Designer*).

6.6.2 Server-Side Idoc Script

There are a number of ways to access custom section properties using server-side Idoc Script:

- `ssGetNodeProperty (name)`: this script extension retrieves the value for the named property for the current web site section.
- `ssGetNodeProperty (nodeId, name)`: this script extension retrieves the value for the named property for the specified web site section.
- `SS_GET_ALL_NODE_PROPERTIES` service: this service retrieves a list of all custom section properties for the specified web site section.

You can see these in action in the "CSP Sample Navigation (server)," "CSP Sample Dynamic List," and "Sample CSP Page Title" fragments, which ship with Site Studio (see "Sample fragments" in the *User's Guide for Site Studio Designer*).

The custom section properties are also stored in the XML rendition of the site hierarchy in the `sitenavigation.xml` file (see "[sitenavigation.xml](#)" on page 5-3). So if you are building a navigation fragment that uses server-side script to parse this XML file, you will also have access to the custom section properties for each section.

The name of the property MUST be in quotes. Thus, if you wanted to get the "label" property of a specific node, you would write:

```
<!--$label = ssGetNodeProperty(nodeId, "label")-->
```

6.7 The Fragment Definition File

A fragment definition file is a pure XML file containing content that inherits its structure from the contribution region that the file is assigned to. From a coding view, a fragment definition file for a fragment library looks like this:

```
<fragments>
  <fragment>

    <parameters>
      <parameter> param definition goes here </parameter>
      <parameter> another param def goes here </parameter>
    </parameters>

    <snippets>
      <snippet> HTML snippet goes here </snippet>
      <snippet> another HTML snippet goes here </snippet>
    </snippets>

  </fragment>
</fragments>
```

```

    <element> element definition goes here </element>
    <element> another element def goes here </element>
</elements>

</fragment>
</fragments>

```

The fragment definition file typically contains the following tags:

- "`<fragments>`" on page 6-6
- "`<fragment>`" on page 6-6
- "`<parameter>`" on page 6-8
- "`<snippet>`" on page 6-12
- "`<designview>`" on page 6-13
- "`<element>`" on page 6-13

6.7.1 `<fragments>`

The `<fragments>` tag represents a fragment library.

Parameters

- `id`: a unique name or identifier for the fragment library.
- `name`: the display name for the fragment library.
- `readonly`: can be set to `True` to prevent the fragments in the library from being edited in Designer.

The `<fragments>` tag contains a collection of one or more `<fragment>` child tags.













6.7.2 `<fragment>`

The `<fragment>` tag represents a single fragment definition.

Parameters

- `id`: a unique name or identifier for the fragment. It is used in XPath expressions when including snippets by reference.
- `name`: the displayed name for the fragment.
- `language`: the server-side language for the fragment implementation. The languages available are `idoc`, `asp` or `jsp`.
- `type`: the type for this fragment. It is used to determine which category in the Designer Toolbox the fragment should display in. It also determines whether the fragment is a static list or dynamic list. The available types are `[navigation | staticlist | dynamiclist | other]`
- `icon`: the name of the icon to use in the Designer Toolbox for this fragment. There is a predefined list of fragment icons to choose from:

Icon name	Image
region	
wysiwyg	

Icon name	Image
plaintext	abc
image1	
image2	
list1	
list2	
list3	
list4	
list5	
list6	
tree	
horizontalrule	—
nonbreakingspace	␣
linebreak	↵
span	<sp>
div	<div>
heading1	<h1>
heading2	<h2>
heading3	<h3>
heading4	<h4>
heading5	<h5>
heading6	<h6>
copyright	©
flash	
companylogo	
documents	

Child Tags

The <fragment> tag contains the following child tags:

- <parameters>: a collection of zero or more parameters. For more information, see "<parameter>" on page 6-8.

- `<snippets>`: a collection of zero or more snippets. For more information, see "`<snippet>`" on page 6-12.
- `<elements>`: a collection of zero or more elements. For more information, see "`<element>`" on page 6-13.

6.7.3 `<parameter>`

The `<parameter>` tag represents a single parameter of a fragment. Its attributes are:

Parameters

- `name`: the parameter name.
- `type`: the parameter type, of which there are several:
 - `text`: a simple text parameter that may contain a predefined list of choices using one or more `<option>` child tags.
 - `bigtext`: a multi-line text parameter.
 - `boolean`: a simple True/False parameter.
 - `integer`: a simple integer parameter.
 - `float`: a simple floating point parameter.
 - `size`: a parameter representing an HTML size (10px, 50%, 3pt, and so forth).
 - `color`: a color value, either in RGB or an HTML color name.
 - `url`: a URL.
 - `manageddoc`: the `dDocName` of a managed document in Oracle Content Server, selected through the Search Results page. The query parameters can be specified using the `<querytext>` child tag.
 - `managedurl`: the `DocUrl` of a managed document in Oracle Content Server, selected through the Search Results page. The query parameters can be specified using the `<querytext>` child tag.
 - `managedquery`: a query text string, selected through the `CAPTURE_QUERY` page.
 - `cssstyle`: a text parameter that represents a CSS style attribute. This type behaves like the "text" type. (It is intended for future use.)
 - `siteid`: a `siteId` value for one of the web sites on the same instance of the content server, selected through the Select Site dialog.
 - `nodeid`: a `nodeId` value for one of the sections in the web site, selected through a site hierarchy in the Select Section dialog.
 - `custom`: a parameter that provides its own custom GUI for entering values using the `<customgui>` child tag.
- `description`: the description of the parameter to appear in the Fragment Parameter Values dialog.
- `required`: a true or false value that determines if the parameters for a fragment are required before that fragment can be added to a layout page.
- `optionsonly`: a True or False value that applies only if one or more `<option>` child options is supplied with possible predefined options.

The `<parameter>` tag may contain text representing the default value (if any) for the parameter. It may also contain the following optional child tags:

- `<option>`: use one or more option tags to specify a list of predefined choices for parameters of type "text." For more information, see "[<option>](#)" on page 6-9.
- `<querytext>`: contains the *QueryText* value for parameters of type *manageddoc* and *managedurl*. For more information, see "[<querytext>](#)" on page 6-9.
- `<validate>`: contains a customized script function for validating the parameter. For more information, see "[<validate>](#)" on page 6-9.
- `<convert>`: contains a customized script function for converting the parameter before inserting it. For more information, see "[<convert>](#)" on page 6-10.
- `<customgui>`: contains a customized HTML snippet to provide its own GUI for entering parameter values. For more information, see "[<customgui>](#)" on page 6-11.

6.7.3.1 `<option>`

The `<option>` tag represents a single option in a choice list and applies only to parameters of the type *text*. The `<option>` tag should contain text that will display in the choice list. Its single attribute is:

- `value`: the value to be inserted if this option is chosen. This attribute is optional if the value displayed in the option is the same as the value to be inserted.

For example:

```
<option>Option 1</option>
<option>Option 2</option>
<option>Option 3</option>
<option>Option 4</option>
```

or

```
<option value="A">Option A</option>
<option value="B">Option B</option>
<option value="C">Option C</option>
<option value="D">Option D</option>
```

6.7.3.2 `<querytext>`

The `<querytext>` tag represents the query parameter to use when performing a search in the content server and applies only to parameters of the type *manageddoc* or *managedurl*. The tag contains the query text within a CDATA section. It has no attributes.

For example:

```
<querytext>
  <![CDATA[
    dExtension <matches> `gif` <or> dExtension <matches> `jpg`
  ]]>
</querytext>
```

6.7.3.3 `<validate>`

The `<validate>` tag is optional and represents a customized script routine that can be provided to validate a parameter before it is inserted into the fragment snippets. The script routine can be written in any WSH-compatible scripting language (VBScript or JScript) and must contain a single function named *validate* that takes a single string as input.

The return value for the function will be either:

- `boolean`: *True* or *False*, to represent a valid or invalid parameter.
or
- `string`: returns an empty string if the parameter is valid, and an error message if the parameter is invalid.

The `<validate>` tag contains the script within a CDATA section. Its single attribute is:

- `language`: the language used for this script is VBScript or JScript.

For example:

```
<validate language="VBScript">
  <![CDATA[
    Function validate(strInput)
      If InStr(strInput, "hello") > 0 Then
        validate = ""
      Else
        validate = "Error, ValidatedParam must contain " _
          "the text 'hello' somewhere in the string"
      End If
    End Function
  ]]>
</validate>
```

Note: The `<validate>` tag is operational, but it is not exposed in the Fragment Editor user interface. It is, therefore, currently unsupported.

6.7.3.4 `<convert>`

The `<convert>` tag is optional and represents a customized script routine that can be provided to convert a parameter as it is inserted into the fragment snippet. The script routine can be written in any WSH-compatible scripting language (VBScript or JScript) and must contain a single function named `convert` that takes a single string as input and returns the converted string as output.

The `<convert>` tag contains the script within a CDATA section.

Parameters

- `language`: the language used for this script is VBScript or JScript.

For example:

```
<convert language="VBScript">
  <![CDATA[
    Function convert(strInput)
      If StrComp(strInput, "") = 0 Then
        convert = "(empty)"
      Else
        convert = "Your Value Was [" & strInput & "]"
      End If
    End Function
  ]]>
</convert>
```

Note: The `<convert>` tag is operational, but it is not exposed in the Fragment Editor user interface. It is, therefore, currently unsupported.

6.7.3.5 `<customgui>`

The `<customgui>` tag is used only for parameters of type *custom*. It allows fragment designers to provide their own (simple) GUI for entering a fragment parameter. It contains an HTML snippet in a CDATA section that can contain any standard HTML or JavaScript.

It interacts with Site Studio through four JavaScript methods.

- `window.external.GetValue()`: the method used to obtain an initial value for the parameter. The custom parameter starts off the same, as a plain text parameter, but it has a button (Figure 6–1) in the edit field to bring up the custom GUI. This method gets the current value from the edit field.

Figure 6–1 Edit Field Custom GUI Button



- `window.external.SetValue()`: the method used to set a new value in the edit field.
- `window.external.OnOK()`: the method used to tell Site Studio it has finished and that it should accept the values passed to it by the `SetValue()` method.
- `window.external.OnCancel()`: the method used to tell Site Studio it has finished but that it should ignore the values passed to it, if any, by the `SetValue()` method.

For example:

```
<customgui>
<![CDATA[
  <HTML>
  <HEAD>
  <SCRIPT language="javascript">
    function initialize()
    {
      strResult = window.external.GetValue() + "00000";
      bold.checked = (strResult.charAt(0) == "1");
      italic.checked = (strResult.charAt(1) == "1");
      underlined.checked = (strResult.charAt(2) == "1");
      font.checked = (strResult.charAt(3) == "1");
    }
    function getvalue()
    {
      strResult = "";
      strResult = strResult + (bold.checked ? "1" : "0");
      strResult = strResult + (italic.checked ? "1" : "0");
      strResult = strResult + (underlined.checked ? "1" : "0");
      strResult = strResult + (font.checked ? "1" : "0");
      return strResult;
    }
  </SCRIPT>
  </HEAD>
  <BODY onload="initialize();">
    Welcome to my Customized Parameter Input Screen. This example
    could perhaps be used as a starting point for entering the
```

```

contribution suppression flags.<BR>
<TABLE>
  <TR>
    <TD>Allow BOLD</TD>
    <TD><INPUT type="checkbox" id="bold"></TD>
  </TR>
  <TR>
    <TD>Allow ITALIC</TD>
    <TD><INPUT type="checkbox" id="italic"></TD>
  </TR>
  <TR>
    <TD>Allow UNDERLINED</TD>
    <TD><INPUT type="checkbox" id="underlined"></TD>
  </TR>
  <TR>
    <TD>Allow Font Changes</TD>
    <TD><INPUT type="checkbox" id="font"></TD>
  </TR>
  <TR><TD COLSPAN="2">&nbsp;   </TD></TR>
  <TR>
    <TD COLSPAN="2">
      <INPUT type="button" value="OK"
        onclick="window.external.SetValue(getvalue()); window.external.OnOK();" />
      <INPUT type="button" value="Cancel"
        onclick="window.external.OnCancel();" />
    </TD>
  </TR>
</TABLE>
</BODY>
]]>
</customgui>

```

Note: The `<customgui>` tag is operational, but it is not exposed in the Fragment Editor user interface. It is, therefore, currently unsupported.

6.7.4 <snippet>

The `<snippet>` tag represents a single snippet of a fragment.

Parameters

- `id`: the unique identifier for the snippet in the fragment definition.
- `location`: the location in the layout page where the snippet should be placed. There are four locations: [`head` | `topofbody` | `drop-point` | `bottomofbody`]
- `include`: how the snippet should be included in a template:
 - `simple`: copy the contents of the snippet directly into the template (do not mark up the snippet with fragment instance details), and replace parameters directly inline by searching for `%paramname%` in the snippet content when it is first inserted.
 - `inline`: copy the contents of the snippet directly into the template and mark the content as a fragment instance snippet so that it can be treated atomically and moved, edited, and deleted as a fragment snippet.

- reference: insert an `ssIncludeXml()` call into the template and mark it as a fragment instance snippet so that it can be treated atomically and moved, edited, and deleted as a fragment snippet.

The `<snippet>` tag contains a single complete snippet of HTML or script that makes up a single atomic piece of the fragment content. The snippet content is contained within a CDATA section. Additionally, the tag may contain the following optional child tags:

- `<designview>`: a design-time representation of the snippet that will be shown in design view in Designer.

6.7.5 <designview>

The `<designview>` tag represents an optional design-time view of a fragment snippet that will be shown in the Design view in Designer (the default display will be the fragment name). The tag contains HTML within a CDATA section. It has no attributes.

For example:

```
<designview>
  <![CDATA[
    example copyright fragment goes here
  ]]>
</designview>
```

6.7.6 <element>

The `<element>` tag represents a simple element contained in a static list fragment definition. The syntax for the `<element>` tag in a `<fragment>` definition tag is identical to the syntax of the `<element>` tag used in a `<region>` definition tag in a layout page.

Only elements of type 1, 4, 5, and 6 (*wysiwyg*, *custom*, *image*, and *plaintext*) are allowed as elements within a static list fragment definition.

6.8 Fragment Instance Structure in the <ssinfo> XML Data Island

The structure that defines a fragment instance and its parameters is maintained in the `<ssinfo>` XML data island by the Designer application. The data island contains a single `<fragmentinstance>` tag for every instance of a fragment on a page template, and each tag contains some Idoc Script that declares the fragment instance parameters. The `<ssinfo>` data island will appear only when a fragment with a `<head>` snippet is added to a page template. Fragments added to region templates and subtemplates will not generate an `<ssinfo>` XML data island.

Advanced fragment types, such as static and dynamic lists, may have additional child tags in the `<fragmentinstance>` tag. For the exact syntax for these tags and their attributes, see the Technical Reference Guide for Site Studio 10gR3.

Native Documents and Conversion

This section covers the following topics:

- ["About Native Documents"](#) on page 7-1
- ["wcmDynamicConversion"](#) on page 7-1
- ["Document Conversion in the Properties Pane"](#) on page 7-1
- ["Common Errors Using Native Documentation"](#) on page 7-2

7.1 About Native Documents

Native documents are a part of many Web sites. While many native documents are typically word processing or documents for presentations, native documents can be most any type of document.

When placed in a Web site, the native documents are converted to a string of HTML, and inserted in the region template.

7.2 wcmDynamicConversion

The *wcmDynamicConversion* method uses Dynamic Converter to convert the document into HTML so that it can be viewed on the web site.

Note that if you are using Native Docs with the WCM_PLACEHOLDER service then the *wcmDynamicConversion* tag on the Region Template must specify the *dDocName* of the conversions definition file as follows:

```
<!--$wcmDynamicConversion("rule1", "dataFileDocName=myNativeDoc", "pageNum=2",  
    "conversionsDefinitionDocName=convDefDocName")-->  
<!--$wcmDynamicConversion("rule1", "dataFileDocName=myNativeDoc", "pageNum=2",  
    "conversionsDefinitionDocName=convDefDocName", "conversionType="simple")-->
```

7.3 Document Conversion in the Properties Pane

Document conversion must be listed in the properties pane. The most common reason that a native document doesn't convert as expected is because the conversion isn't properly listed.

In Designer, the conversion definition is found in the first section of the properties pane. Select an item in the site hierarchy. The conversion must be listed there to work.

However, if *wcmDynamicConversion* is used, then the item listed in the conversion definition in the panel will not be used. The *wcmDynamicConversion* tag allows you to explicitly state the rule. When you associate a dynamic conversion with a region

definition and region template (rather than explicitly coding it), the conversion must be stated in the properties pane. When you use *wcmDynamicConversion*, this is not necessary as you are explicitly stating the rule you wish to use. You can even use a rule not stated in the dynamic conversion listed in the conversion definition in the properties pane.

For more information, see the *User's Guide for Site Studio Designer*.

7.4 Common Errors Using Native Documentation

Generally native documents are easy to implement in Site Studio. Site Studio is constructed so that all of your conversion rules are contained in one conversion definition. It is not recommended to keep different conversion files for different rules; all rules should be listed in one conversion definition.

This definition should be listed in the Conversions Definition property in the properties pane for the Web site. Not listing the conversions definition here is the most common reason for native documents not appearing as expected in a Web site. This is true even if you use the `WCM_PLACEHOLDER` Idoc script extension to explicitly state the conversion rule.

Custom Elements

This section covers the following topics:

- ["About Custom Elements"](#) on page 8-1
- ["Element API"](#) on page 8-1
- ["Custom Elements within Contributor"](#) on page 8-3
- ["Legacy Custom Element Compatibility"](#) on page 8-5

8.1 About Custom Elements

A custom element is a user-defined Site Studio element. In addition to the other productized Site Studio elements, the custom element provides a way to extend the Site Studio product to suit individual business needs.

From a code perspective, custom elements are essentially full HTML type files (for example, htm, hcsp, jsp, and so forth) that reside within an IFRAME in the contributor form. A custom element uses an API and implements a hand-full of callbacks in order function correctly as a Site Studio element.

8.2 Element API

In order for a custom element to function properly in a contributor form, a custom element must utilize an API and implement a hand-full of callbacks. The ElementAPI object is a JavaScript object explicitly loaded into the custom element page that facilitates communication between the Contributor form and the custom element. The ElementAPI provides methods for custom elements to communicate to the Contributor form and a callback mechanism for the Contributor form to pass notifications to the custom element.

This section contains the following topics:

- ["Loading the Element API"](#) on page 8-1
- ["Element API Dependent Scripts"](#) on page 8-2

8.2.1 Loading the Element API

Before the ElementAPI and its supporting libraries can be used; the ElementAPI must first be loaded into the Custom Element page. After the ElementAPI is loaded, the Custom Element should continue with page initialization and notify the Contributor form that the Custom Element is loaded.

```
<html>
<head>
  <title>Default Custom Element Form</title>
  <script type="text/javascript">
    function Initialize()
    {
      //=====
      // TODO: ElementAPI is loaded. Place Custom Element initialization code here.
      //=====

      // Let the Contributor Form know that this Custom Element is loaded and ready.
      ElementAPI.Ready();
    }

    // Load the ElementAPI and its supporting libraries - then call Initialize()
    // Parameter 1: The Custom Element's window object. This parameter uniquely
    // identifies the Custom Element to the Contributor Form.
    // Parameter 2: A function pointer. This function will be executed after the
    // ElementAPI and its supporting libraries are loaded.
    try {
      window.top.WCM.InitializeCustomElement(window, Initialize);
    } catch(e) { }
  </script>
</head>
<body>
  <h3>Default Custom Element Form</h3>
</body>
</html>
```

8.2.2 Element API Dependent Scripts

When the ElementAPI is loaded into the Custom Element page, so are the ElementAPI dependent scripts. These scripts contain most of the JavaScript WCM library and is also available to use in authoring custom elements. The following is a list of script files loaded into a custom element.

- wcm.js
- ./base/wcm.dhtml.js
- ./base/wcm.get.js
- ./base/wcm.http.js
- ./base/wcm.popup.js
- ./sitestudio/wcm.contentserver.popup.js
- ./form/elements/wcm.elementapi.js
- ./sitestudio/elements/wcm.sitestudio.elementapi.js
- ./sitestudio/wcm.idc.js
- ./form/elements/element/wcm.element.js
- ./form/elements/custom/wcm.custom.js

As with other custom scripts, you can modify any of these as you need and place the modified script in the /custom directory.

8.3 Custom Elements within Contributor

The contributor form communicates to a custom element by executing functions implemented by the custom element. As part of the initialization process, a custom element needs to register these functions by passing their function pointers to the contributor form.

This section contains the following topics:

- ["Communication from a Contributor Form to a Custom Element"](#) on page 8-3
- ["Communication from a Custom Element to a Contributor Form"](#) on page 8-4

8.3.1 Communication from a Contributor Form to a Custom Element

The contributor form communicates to a custom element by executing functions implemented by the custom element. As part of the initialization process, a custom element needs to register these functions by passing their function pointers to the contributor form.

The following is a list of functions that can be registered with the contributor form. None of these functions need to be implemented by the custom element; however, a few of them are required if the intention is to collect and save data from a Contributor user. Furthermore, all of these functions except the `IsDirty()` function, when executed, will be passed a callback function pointer to execute when the task is complete. This allows for asynchronous communication if a custom element needs to perform an asynchronous task during execution.

Function	Description
<code>CanCloseElement (callback) ;</code>	The contributor form will execute this method when the contributor updates information within the element. The implementation of the function should calculate whether the custom element can be safely closed. For instance, if the data does not pass validation, then the custom element should indicate that it cannot be closed.
<code>GetElementContent (callback) ;</code>	The contributor form will execute this method when the contributor updates information within the element. The implementation of the function should pass back string content to be saved.
<code>Hide (callback) ;</code>	<p>The contributor form will execute this method whenever the form performs a DHTML task that overlays a HTML element over the custom element. For instance, this method will be executed when the metadata tab is activated and the contributor elements are obscured.</p> <p>This method was introduced specifically for the Ephox-based elements, because Java applets always have top z-index. All other elements (HTML-based elements) can ignore this method.</p>
<code>Show (callback) ;</code>	<p>The contributor form will execute this method whenever the form performs a DHTML task that removes an overlay that makes custom elements reappear.</p> <p>This method was introduced specifically for the Ephox-based elements, because Java applets always have top z-index. All other elements (HTML-based elements) can ignore this method.</p>

Function	Description
IsDirty();	The contributor form will execute this method whenever the form popup is closed. The custom element should calculate whether or not unsaved changes exist and then notify the contributor if there are unsaved changes.

The following is a JavaScript code snippet of how a custom element can register functions with the contributor form:

```
function CanCloseElement(callback)
{
    // No data validation in this sample - just pass back a true value.
    callback({canClose: true});

    // Here is an example of passing a false value
    // callback({canClose: false, reason: 'Failed validation. Use only lowercase
    // letters.'});
}

function GetElementContent(callback)
{
    // Pass back some sample content for demo purposes.
    callback('This is my Custom Element Content.');
```

```
function Show(callback)
{
    // Just handle this notification by executing the callback.
    callback();
}

function Hide(callback)
{
    // Just handle this notification by executing the callback.
    callback();
}

function IsDirty()
{
    // This Custom Element is never dirty - so pass a false value.
    return {isDirty: false};
}

// Set callback methods for the Contributor Form to send notifications to this
Element.
ElementAPI.SetCallback('CanCloseElement', CanCloseElement);
ElementAPI.SetCallback('GetElementContent', GetElementContent);
ElementAPI.SetCallback('Show', Show);
ElementAPI.SetCallback('Hide', Hide);
ElementAPI.SetCallback('IsDirty', IsDirty);
```

8.3.2 Communication from a Custom Element to a Contributor Form

A custom element initiates communication with the contributor form by using the ElementAPI JavaScript object. The following is a list of available ElementAPI methods.

Function	Description
<code>ElementAPI.GetDefaultData();</code>	Retrieves the default content stored in the data file.
<code>ElementAPI.SetHostHeight(height);</code>	Sets the height of the elements containing IFRAME.
<code>ElementAPI.SetRequiredIndicator(is Required);</code>	Toggles the Required graphic indicator in the Contributor Form UI.
<code>ElementAPI.GetSite(options);</code>	Displays the Choose Website picker UI.
<code>ElementAPI.GetSection(options);</code>	Displays the Choose Website Section picker UI.
<code>ElementAPI.GetColor(options);</code>	Displays the Color picker UI.
<code>ElementAPI.GetFont(options);</code>	Displays the Get Font picker UI.
<code>ElementAPI.GetSearchResults(options);</code>	Displays the Oracle Content Server's Get Search Results page.
<code>ElementAPI.GetQueryText(options);</code>	Displays the Get Query Text UI.
<code>ElementAPI.CaptureQuery(options);</code>	Displays the Oracle Content Server's Capture Query page.
<code>ElementAPI.GetHyperlink(options);</code>	Displays the Hyperlink Wizard UI.
<code>ElementAPI.FocusForm(options);</code>	Focuses the parent window thereby blurring the Element window.

8.4 Legacy Custom Element Compatibility

All custom element forms created with Site Studio releases 10gR3 (10.1.3.3.2) and earlier are not compatible with Site Studio 11gR1 and will need to be manually upgraded (re-authored). The primary reason for not maintaining backward compatibility is Site Studio's prior dependency upon Internet Explorer's proprietary *window.external* functionality. The *window.external* functionality of custom elements used in Site Studio release 10gR3 and earlier is blocked at the point of code execution and is not easily duplicated in a cross-browser and cross-platform DHTML solution.

The upside to breaking backward compatibility is that current custom elements are much more flexible, and better integrated into the Contributor application architecture (in addition to being a cross-browser and cross-platform solution).

This section contains the following topics:

- ["Detecting Legacy Custom Element Forms"](#) on page 8-5
- ["Upgrading Legacy Custom Elements"](#) on page 8-6

8.4.1 Detecting Legacy Custom Element Forms

A custom element form created using Site Studio 10gR3 (10.1.3.3.2) or earlier (that is, a legacy custom element form), if loaded into the Contributor application with the *SSValidateCustomElements* flag is set to *true*, will be detected. An error message will be displayed in its place within the contributor form.

The Contributor application does this by first downloading the custom element form, parsing the source code, and determining whether or not the custom element form is compatible with the new Contributor application.

The functionality and overhead to detect legacy custom element forms is unnecessary on production installations and is turned off by default. To turn on legacy custom element form detection, add the following line to Oracle Content Server's *config.cfg* file and restart the server:

```
SSValidateCustomElements=true
```

8.4.2 Upgrading Legacy Custom Elements

Any custom element forms created using a Site Studio release up to 10gR3 (10.1.3.3.2) are not compatible with Site Studio 11gR1. They must be manually upgraded and re-authored. The primary reason for not maintaining backward compatibility is Site Studio's prior dependency on Internet Explorer's proprietary *window.external* functionality (due to the ActiveX control used for the legacy Contributor application). This functionality was removed from Site Studio as a result of the browser-independent, JavaScript-based Contributor application that is used in Site Studio 10gR3 (10.1.3.3.3) and higher (including 10gR4 and 11gR1).

Idoc Script Extensions

Site Studio uses several Idoc Script extensions that are used to run a Web site:

- ["About Idoc Script Extensions"](#) on page 9-2
- ["wcmPlaceholder"](#) on page 9-2
- ["wcmElement"](#) on page 9-3
- ["wcmListStart"](#) on page 9-4
- ["wcmListEnd"](#) on page 9-4
- ["wcmListElement"](#) on page 9-4
- ["wcmListRowCount"](#) on page 9-5
- ["wcmDynamicList"](#) on page 9-5
- ["wcmIncludeElement"](#) on page 9-6
- ["wcmDynamicConversion"](#) on page 9-6
- ["wcmFragment"](#) on page 9-7
- ["wcmUrl"](#) on page 9-7
- ["ssIncludeXml"](#) on page 9-8
- ["ssGetDocInfo"](#) on page 9-9
- ["ssGetXmlNodeCount"](#) on page 9-9
- ["ssIncDynamicConversion"](#) on page 9-10
- ["ssIncDynamicConversionByRule"](#) on page 9-10
- ["ssIncDynamicConversionByRulesEngine"](#) on page 9-11
- ["ssIncInlineDynamicConversion"](#) on page 9-11
- ["ssIsNativeDoc"](#) on page 9-11
- ["ssRandom"](#) on page 9-12
- ["ssGetNodeProperty"](#) on page 9-12
- ["ssGetWebsiteNodeType"](#) on page 9-13
- ["ssGetCoreMajorVersion"](#) on page 9-13
- ["ssSplitString"](#) on page 9-13
- ["ssGetWebsiteName"](#) on page 9-14
- ["ssGetSiteProperty"](#) on page 9-14

- ["ssGetFirstNodeId"](#) on page 9-15
- ["ssGetRelativeNodeId"](#) on page 9-15
- ["ssLoadSiteNavResultSet"](#) on page 9-15
- ["ssGetServerRelativeUrl"](#) on page 9-16
- ["ssGetServerRelativePath"](#) on page 9-16
- ["ssGetUrlPageName"](#) on page 9-17
- ["ssGetNodeLabel"](#) on page 9-17
- ["ssGetNodeLabelPath"](#) on page 9-17
- ["ssGetAllSites"](#) on page 9-18
- ["ssLink"](#) on page 9-18
- ["ssNodeLink"](#) on page 9-18
- ["ssWeblayoutUrl"](#) on page 9-19

Note: These script extensions are subject to change with each release of Site Studio.

9.1 About Idoc Script Extensions

Idoc Script is the server-side custom scripting language for Oracle Content Server. It enables you to reference variables, conditionally include content in HTML pages, and loop over results returned from queries.

When a component is installed on Oracle Content Server, the component can add extensions to the Idoc Script and variables, allowing for further customization.

Idoc Script is used primarily for the presentation of HTML templates and configuration settings. The Site Studio component adds a number of script extensions which are described in this section.

9.2 wcmPlaceholder

Description

The `wcmPlaceholder()` function defines an area on the page that will hold content. The actual content that the placeholder will display depends upon the data file, subtemplate, and other Web site objects associated with the placeholder.

Parameters

- `placeholderName`: The name of the placeholder. Required.
- `dataFileDocName`: The *dDocName* of the data file to associate with the placeholder.
- `templateDocName`: The *dDocName* of the region template or subtemplate to associate with the placeholder.
- `placeholderDefinitionDocName`: The *dDocName* of the placeholder definition to map to the placeholder.
- `regionDefinitionDocName`: The *dDocName* of the region definition to associate with the region template named in `templateDocName`.

- `placeholderActions`: The allowed actions of the placeholder definition, as follows:
 - `E` allows contributor update
 - `P` allows workflow approve
 - `R` allows workflow reject
 - `I` allows viewing docInfo
 - `S` allows switching the data file
 - `U` allows viewing the web usage report
 - `T` allows viewing the web tracker report
 - `M` allows updating the docInfo
 - `V` allows switching the region template
 - `N` allows remove content

Each selection corresponds to the checkbox for the action in the design view of the placeholder definition in Designer.

If you use parameters that do not work together (for example, specifying a subtemplate as well as a region definition), then the script will execute based on the order of parameters listed above.

Code Examples

```
<!--$wcmPlaceholder("placeholderName")-->

<!--$wcmPlaceholder("placeholderName", "dataFileDocName=dDocName",
"templateDocName=[region template dDocName|subtemplate
dDocName]", "placeholderDefinitionDocName=dDocName",
"regionDefinitionDocName=dDocName", placeholderActions=EPRI SUTMVN)-->
```

9.3 wcmElement

Description

The `wcmElement()` script inserts an element in a region template. The element does not have to be listed in the region definition.

This script allows you to retrieve out of the data file the content associated with the named element. This is only for WYSIWYG, Image, Text, and Custom elements. The content of the element that is inserted will be further evaluated in the scope of the current layout page and therefore can include further server-side Idoc Script, if necessary.

Parameters

- `dDocName`: the *dDocName* of the data file containing the `elementName`. If a *dDocName* is not specified, then the current data file associated with the placeholder is used.
- `elementName`: The name of the element. Required.

Code Example

```
<!--$wcmElement("elementName")-->

<!--$wcmElement("dDocName", "elementName")-->
```

9.4 wcmListStart

Description

This function defines the start of a List element.

Parameters

- `elementName`: The name of the list element. Required.

Code Example

```
<!--$wcmListStart("elementName")-->
```

9.5 wcmListEnd

Description

This function defines the end of a list element.

Parameters

- `elementName`: The name of the list element. Required.

Code Example

```
<!--$wcmListEnd("elementName")-->
```

9.6 wcmListElement

Description

This performs the same function as `wcmElement` but it is for placing static list element content into the region template. This script allows you to retrieve out of the data file the content associated with the named static list element. The content of the element that is inserted will be further evaluated in the scope of the current layout page and therefore can include further server-side Idoc Script, if necessary.

Parameters

- `listName`: The name of the list element. If `wcmListElement` is used between `wcmListStart` and `wcmListEnd` then `listName` can be omitted.
- `elementName`: The name of the element within the list used to construct the `listElement`.
- `rowNum`: The row position of the `elementName` within the `listName`.

Code Example

```
<!--$wcmListElement("Wysiwyg", rowNum)-->
```

The above example uses only `elementName` and `rowNum`.

Example within context:

```
<!--$wcmListStart("StaticList")-->
<!--$wcmListIndex = 0-->
<!--$wcmListNumRows = wcmListRowCount()-->
<table><!--$loopwhile wcmListIndex < wcmListNumRows-->
<tr>
<td><!--$wcmListElement("Wysiwyg", wcmListIndex)--></td>
```

```

<td><!--$wcmListElement("plaintext", wcmListIndex)--></td>
</tr>
<!--$wcmListIndex = wcmListIndex + 1-->
<!--$endloop--></table>
<!--$wcmListEnd("StaticList")-->

```

9.7 wcmListRowCount

Description

This script retrieves the number of rows in the list element. An integer value is returned.

Parameters

- `listName`: The name of the list element. If `wcmListRowCount` is used between `wcmListStart` and `wcmListEnd` then `listName` can be omitted.

Code Example

```

<!--$wcmListRowCount("listName")-->

```

Example within context:

```

<!--$wcmListStart("listName")-->
<!--$wcmListIndex = 0-->
<!--$wcmListNumRows = wcmListRowCount()-->
<table><!--$loopwhile wcmListIndex < wcmListNumRows-->
<tr>
<td><!--$wcmListElement("Wysiwyg", wcmListIndex)--></td>
<td><!--$wcmListElement("plaintext", wcmListIndex)--></td>
</tr>
<!--$wcmListIndex = wcmListIndex + 1-->
<!--$endloop--></table>
<!--$wcmListEnd("listName")-->

```

9.8 wcmDynamicList

Description

Prepares the service binder for a call to `SS_GET_SEARCH_RESULTS` by retrieving the search parameters from the named dynamic list element. It returns a Boolean indicating success or failure.

Parameters

- `elementName`: The name of the element within the list used to construct the `listElement`

Code Example

```

<!--$wcmDynamicList("elementName")-->

```

Example within context:

```

<!--$if wcmDynamicList("dynamicList")-->
<!--$executeService("SS_GET_SEARCH_RESULTS")-->
<!--$loop SearchResults-->
<!--$xml(dDocTitle)--><br />
<!--$endloop-->

```

```
<!--$endif-->
```

9.9 wcmIncludeElement

Description

This script allows the contents of an element to be displayed outside the context of the region. When an element is in the context of a region (that is, in a region template), then there is an implied data file association based on that region template.

With `wcmIncludeElement`, you can force a data file association when the element is outside the context of a region.

Parameters

- `dDocName`: The *dDocName* for the data file.
- `expression`: The XPath expression (to identify a specific node or nodes of the XML data file).

Returns

The contents of the element are returned.

Code Example

```
queryVal = <!--$wcmIncludeElement("datafiledDocName",  
"wcm:root/wcm:element[@name='elementName']/node()")-->
```

9.10 wcmDynamicConversion

Description

Used to create a dynamic conversion of a native document.

Parameters

- `ruleName`: The name of the rule as defined in the `ConversionsDefinition` file. Required.
- `dataFileDocName`: The *dDocName* of the native document to convert.
- `pageNum`: The page number within the native document to convert.
- `conversionsDefinitionDocName`: The *dDocName* of the conversion definition.
- `conversionType`: Possible type values = [simple|full|rule|engine|command].
- `conversionTemplate`: The *dDocName* of the template. Valid only when using `conversionType="full"`.
- `conversionLayout`: The *dDocName* of the layout. Valid only when using `conversionType="full"`.
- `conversionRuleName`: The name of the rule. Valid only when using `conversionType="rule"`.
- `conversionCommand`: A piece of Idoc Script executed in the context of the page. Valid only when using `conversionType="command"`.

Returns

The HTML generated by the dynamic conversion based on the parameters.

Code Example

Using the default:

```
<!--$wcmDynamicConversion("ruleName")-->
```

Including a type value:

```
<!--$wcmDynamicConversion("ruleName", "dataFileDocName=dDocName", "pageNum=number", "
conversionsDefinitionDocName=dDocName", "conversionType=engine")-->
```

9.11 wcmFragment

Description

This tag is used on templates to add a fragment. If the tag is used on a region template or subtemplate, only the first drop-point snippet will be used.

Parameters

- `fragmentInstanceId`: The ID of the fragment instance on the page. Used to differentiate fragments that have multiple pages of returns, such as dynamic lists. The ID allows the dynamic lists to display different 'pages' of query results. For instance, one fragment can be on page 1 of its query results, while the other can display page 3 of its own results. If the same ID is used for both fragments, then clicking to a particular page of results for one query will cause the other fragment to display the same page number of its own results.
- `fragmentDocName`: The *dDocName* of the fragment library.
- `fragmentId`: The ID of fragment within the named library.
- `snippetId`: The ID of snippet within the named fragment.
- `tagProperties`: The named value pairs of properties of the fragment.

Code Example

```
<!--$wcmFragment("fragmentInstanceID", "fragmentDocName", "fragmentID",
"snippetID", "ssTheme=default", "ssHoverColor=", "ssTextColor=", "ssFocusColor=",
"ssShowHome=true", "ssShowNext=false", "ssClassName=IDocNavTabsTop")-->
```

9.12 wcmUrl

Description

This is a single script which can be used to make all link formats. Legacy sites required three different scripts to generate the different types of links available in Site Studio.

Parameters

- `type`: The type of link, entered as one of the following: [nodelink | link | resource | rendition]
 - `nodelink` is used to create links to sections, which by default end with `index.htm`.

- `link` is used to create links to content, which by default end with `<contentId>`.
- `resource` is used to link to weblayout static resources, such as images.
- `rendition` is used to link to Digital Asset Manager (DAM) renditions.
- `nodeId`: the ID of the node referenced. Used only when the type is set to either `nodelink` or `link`.
- `siteId`: the site ID of the Web site referenced. Used only when the type is set to either `nodelink` or `link`.
- `dDocName`: the *dDocName* of the content referenced. Used only when the type is set to either `link` or `resource`.
- `path`: the path to the content referenced. Used only when the type is set to either `link` or `resource`.
- `renditionName`: the name of the rendition linked to. Used only when the type is set to `rendition`.

Code Example

Links to sections; by default these links end with `index.htm`.

```
<!--$wcmUrl('nodelink',nodeId)-->
<!--$wcmUrl('nodelink',nodeId,siteId)-->
```

Links to content; by default these links end with a `contentId`.

```
<!--$wcmUrl('link','dDocName')-->
<!--$wcmUrl('link','dDocName','nodeId')-->
<!--$wcmUrl('link','dDocName','nodeId','siteId')-->
```

Links to weblayout static resources; for example, images.

```
<!--$wcmUrl('resource','dDocName')-->
<!--$wcmUrl('resource','groups/public/documents/adacct/mydocname.jpg')-->
```

see also "[ssWeblayoutUrl](#)" on page 9-19.

Links to a DAM rendition:

```
<!--$wcmUrl('rendition',dDocName,'smallimage')-->
```

9.13 ssIncludeXml

Description

This script extension is a core Site Studio method that allows any element within a managed XML file to be extracted and returned in an Idoc string variable, which can be placed directly on a web page as an HTML snippet. The content of the XML node that is being extracted will be further evaluated in the scope of the current layout page and therefore can include further server-side Idoc Script, if necessary.

Parameters

- `dDocName`: The *dDocName* of XML data file.
- `expression`: XPath expression (to identify a specific node or nodes of the XML data file).

Returns

The content of the element listed in the Xpath.

Code Example

```
<!--$ssIncludeXml("dDocName", "expression"-->

<!--$ssIncludeXml("dDocName", "wcm:portal/wcm:element[@name =
"eleName"]/node() "-->
```

9.14 ssGetDocInfo

Description

This script extension retrieves a *DOC_INFO* result set for the named *dDocName*.

Parameters

- *dDocName*: the identifier of the document whose *DOC_INFO* should be retrieved.

Returns

A boolean value indicating the success of the operation.

Code Example

```
<!--$ssGetDocInfo("MyDocName")-->

<!--$if ssGetDocInfo("MyDocName")-->
<!--$endif-->
```

9.15 ssGetXmlNodeCount

Description

This script extension returns a count of the number of repeating instances of a given element within a managed XML file. This is used primarily in the implementation of static list fragments to display a set of repeating contribution elements.

Parameters

- *dDocName*: The *dDocName* of XML data file.
- *expression*: XPath expression (to identify a specific node or nodes of the XML data file).

Returns

The integer value of the number of rows in the list.

Code Example

```
<!--$ssIncludeXml("dDocName", "expression"-->

<!--$ssIncludeXml("dDocName", 'wcm:portal/wcm:element[@name = "eleName"]'-->
```

9.16 ssIncDynamicConversion

Description

This script extension can be used to perform a dynamic conversion of a native document where the resulting HTML is returned in an Idoc string variable that can be placed directly on a web page in the form of an HTML snippet. This method allows the caller to specify the managed conversion template and conversion layout to be used.

Parameters

- `dDocName1`: *dDocName* of native document to convert.
- `dDocName2`: *dDocName* of template to use in conversion.
- `dDocName3`: *dDocName* of layout template to use in conversion.
- `dcPageNum`: The page number of the document to convert (optional).

Returns

An HTML string snippet of the document conversion.

Code Example

```
<!--$ssIncDynamicConversion("dDocName1", "dDocName2", "dDocName3", "3")-->
```

9.17 ssIncDynamicConversionByRule

Description

This script extension, like the *ssIncDynamicConversion* extension, can be used to perform a dynamic conversion of a native document where the resulting HTML is returned in an Idoc string variable that can be placed directly on a web page, in the form of an HTML snippet.

Unlike *ssIncDynamicConversion*, this extension uses a specified rule from the Dynamic Converter rules engine in order to decide which conversion template to use. It therefore does not require the caller to specify a managed conversion template or conversion layout.

Parameters

- `dDocName`: The *dDocName* of native document to convert.
- `rule`: The name of Dynamic Converter rule to use.
- `dcPageNum`: The page number of the document to convert (optional).

Returns

An HTML string snippet of the document conversion.

Code Example

```
<!--$ssIncDynamicConversionByRule("dDocName", "rule", "1")-->
```

9.18 sslIncDynamicConversionByRulesEngine

Description

This script extension, like the *ssIncDynamicConversion* extension, can be used to perform a dynamic conversion of a native document where the resulting HTML is returned in an Idoc string variable that can be placed directly on a web page, in the form of an HTML snippet.

Unlike *ssIncDynamicConversion*, this extension uses the normal Dynamic Converter rules engine in order to decide which conversion template to use. It therefore does not require the caller to specify a managed conversion template or conversion layout.

Parameters

- `dDocName`: The *dDocName* of native document to convert.
- `dcPageNum`: The page number of the document to convert (optional).

Returns

An HTML string snippet of the document conversion.

Code Example

```
<!--$ssIncDynamicConversionByRulesEngine("dDocName", "1")-->
```

9.19 sslIncInlineDynamicConversion

This script extension, like the *ssIncDynamicConversion* extension, can be used to perform a dynamic conversion of a native document where the resulting HTML is returned in an Idoc string variable that can be placed directly on a web page, in the form of an HTML snippet. Unlike *ssIncDynamicConversion*, this extension uses a built-in blank template for the dynamic conversion and therefore does not require the caller to specify a managed conversion template or conversion layout.

Parameters

- `dDocName`: The *dDocName* of native document to convert.
- `dcPageNum`: the page number of the document to convert (optional).

Returns

An HTML string snippet of the document conversion.

Code Example

```
<!--$ssIncInlineDynamicConversion("dDocName", "1")-->
```

9.20 sslsNativeDoc

Description

This script extension can be used to detect whether a managed document is to be considered a native document, and thus available for dynamic conversion, or a contributor data file, which requires no conversion.

The comparison is not an inclusive test for a native document; the test is an exclusive test for data files. The return of *yes* as a result means that the document was shown to not be a data file.

Parameters

- `dDocName`: The *dDocName* of document to test.

Returns

Boolean value [1 | 0].

Code Example

```
<!--$Variable = ssIsNativeDoc("dDocName")-->
```

9.21 ssRandom

Description

This script extension is used to generate a random number.

Parameters

It has no parameters.

Returns

A non-negative integer value random number.

Code Example

```
<!--$ssRandom()-->  
<!--$Variable = ssRandom()-->
```

9.22 ssGetNodeProperty

Description

This script extension can be used to obtain a node property for either the current section or an explicitly specified section. It is typically used to access custom node properties.

Parameters

- `nodeId`: The `nodeId` of the section to be queried (optional). If omitted, the current section is used.
- `name`: The property name.

Returns

Returns the value of the specified property. If "label" is used, the section label will be returned, but other properties can be specified.

Code Example

For example, the following command will retrieve the section label for the current section:

```
<!--$ssGetNodeProperty('label')-->
```

While the following command will retrieve the section label for section 474:

```
<!--$ssGetNodeProperty('474', 'label') -->
```

9.23 ssGetWebsiteNodeType

Description

This script extension can be used to tell the difference between a site node (the root of the hierarchy) and a regular section node. It can also tell the difference between an ASP and a non-ASP site or section node.

Parameters

- `siteId`: The *siteId* of the Web site to be queried (optional).
- `nodeId`: The *nodeId* of the section to be queried.

Returns

The return value is one of the following:

- 0: Site node for an HCSP/JSP type site.
- 2: Section node for an HCSP/JSP type site.
- 100: Site node for an ASP type site.
- 102: Section node for an ASP type site.

Code Example

```
<!--$ssGetWebsiteNodeType("siteId", "nodeId")-->  
<!--$Variable = ssGetWebsiteNodeType("nodeId")-->
```

9.24 ssGetCoreMajorVersion

Description

This script extension can be used to obtain the major version of the underlying content server.

Parameters

It has no parameters.

Returns

Oracle Content Server major version number.

Code Example

```
<!--$Variable = ssGetCoreMajorVersion()-->
```

9.25 ssSplitString

Description

This script extension can be used to split a string into segments based on a specified delimiter.

Parameters

- `string`: The string to be split.
- `delimiter`: The delimiter used to split the string (for example ";").
- `name`: The name of the result set to store the results of the split operation.

Returns

The return value is an integer indicating the number of segments that were split out of the supplied string. The extension also generates a `ResultSet` with one column, named **String**, listing the split strings.

Code Example

```
<!--$ssSplitString("ABC;123;XYZ" , ";" , "StorageRows")-->
```

9.26 ssGetWebsiteName

Description

This script extension can be used to obtain the Web site name for any Web site on the content server.

Parameters

The parameters are:

- `siteId`: The unique identifier for the Web site.

Returns

The Web site name (label) of the `siteId`.

Code Example

```
<!--$ssGetWebsiteName("siteId")-->
```

9.27 ssGetSiteProperty

Description

This script extension can be used to obtain a site property for any Web site on the content server.

Parameters

- `siteId`: The unique identifier for the Web site.
- `propertyName`: The name of the property whose value is required.

Returns

The property listed in `propertyName`, as a string.

Code Example

```
<!--$ssGetSiteProperty("siteId" , "propertyName")-->
```


9.28 ssGetFirstNodeId

Description

This script extension can be used to obtain the identifier (*nodeId*) for the root node of any Web site on the content server.

Parameters

- *siteId*: The unique identifier for the Web site

Returns

The Id of the first node.

Code Example

```
<!--$Variable = ssGetFirstNodeId("siteId")-->
```

9.29 ssGetRelativeNodeId

Description

This script extension can be used to obtain the identifier (*nodeId*) of a node relative to a given node.

Parameters

- *siteId*: The unique identifier for the Web site.
- *nodeId*: The unique identifier for the context node.
- *relative*: The relationship; must be one of: [*child*|*parent*|*prior*|*next*].
- *respectContribOnly*: A boolean value [*yes*|*no*]; the default is *yes* (optional).

Returns

The *nodeId* of the node that has the stated relationship to the context node, or an empty string if there is no node with that stated relationship.

Code Example

```
<!--$Variable = ssGetRelativeNodeId("siteId" , "nodeId" , "child" , "no")-->
```

9.30 ssLoadSiteNavResultSet

Description

This script extension can be used to create a "SiteStudioNavNodes" ResultSet in the Idoc execution environment that will contain the active hierarchy for any Web site on the content server.

Parameters

- *siteId*: the unique identifier for the Web site

Returns

A ResultSet of the Web site hierarchy, called "SiteStudioNavNodes."

The "SiteStudioNavNodes" ResultSet has 5 or more columns, depending on the settings of the configuration flag *SSAdditionalNavResultSetFields* (see ["SSAdditionalNavResultSetFields"](#) on page 12-8 for more information):

- `nodeId`: the unique identifier for the node.
- `parentNodeId`: the unique identifier for the parent node.
- `label`: the label for the node.
- `level`: the depth of the node in the site hierarchy. The root section has a level of 0.
- `href`: the site relative path-based URL to the nodes primary page.

Code Example

```
<!--$ssLoadSiteNavResultSet("siteId")-->
```

9.31 ssGetServerRelativeUrl

Description

This script extension can be used to generate a server relative URL to the primary layout for a specified node, including the trailing `urlPageName` (usually *index.htm*).

Parameters

- `siteId`: The unique identifier for the Web site.
- `nodeId`: A unique identifier for a node.

Returns

A full friendly URL relative to the server.

Code Example

```
<!--$ssGetServerRelativeUrl("siteId" , "nodeId")-->
```

9.32 ssGetServerRelativePath

Description

This script extension can be used to generate a server relative URL to the primary layout for a specified node, excluding the trailing `urlPageName` (usually *index.htm*).

Parameters

- `siteId`: The unique identifier for the Web site.
- `nodeId`: A unique identifier for a node.

Returns

A full friendly URL relative to the server, not including the page name.

Code Example

```
<!--$ssGetServerRelativePath("siteId" , "nodeId")-->
```

9.33 ssGetUrlPageName

This script extension can be used to obtain the *urlPageName* for a specified node.

Parameters

- `siteId`: The unique identifier for the Web site.
- `nodeId`: A unique identifier for a node.

Returns

The *urlPageName* of the node. Usually this is *index.htm*.

Code Example

```
<!--$ssGetUrlPageName("siteId")-->
```

9.34 ssGetNodeLabel

Description

This script extension can be used to obtain the label for a specified node.

Parameters

- `siteId`: The unique identifier for the Web site.
- `nodeId`: A unique identifier for a node.

Returns

The label of the specified node.

Code Example

```
<!--$ssGetNodeLabel("siteId" , "nodeId")-->
```

9.35 ssGetNodeLabelPath

Description

This script extension can be used to obtain a "/" separated string relative to the root of the Web site that can be used for a GUI label (for example, "/products/servers/web server").

Parameters

- `siteId`: The unique identifier for the Web site.
- `nodeId`: A unique identifier for a node.

Returns

The script returns a string separated by "/" of the folders in the stated node in the stated Web site.

Code Example

```
<!--$ssGetNodeLabelPath("siteId" , "nodeId")-->
```

9.36 ssGetAllSites

Description

This script extension can be used to generate a `ResultSet` containing a list of all available Web sites.

Parameters

- `resultSet`: the name for the generated `ResultSet`.

Returns

The list of all available Web sites on the server is returned in a `ResultSet`.

Code Example

```
<!--$ssGetServerRelativePath("resultSet")-->
```

9.37 ssLink

Description

This script extension is used to generate a path-based URL to a named document. This is deprecated in preference of `wcmUrl` (see "[wcmUrl](#)" on page 9-7).

For more information, see "[Using Server-Side Script Links](#)" on page 4-3.

Parameters

- `dDocName`: the *dDocName* of the managed item.
- `targetNodeId`: a unique identifier for a node to use as the target context (optional).
- `targetSiteId`: the unique identifier for the Web site to use as the target context (optional).

Returns

A string which is the friendly link to a specific piece of content.

Code Example

```
<!--$ssLink("dDocName" , "targetSiteId")-->  
<!--$ssLink("dDocName")-->  
<!--$ssLink("dDocName" , "targetNodeId" , "targetSiteId")-->
```

9.38 ssNodeLink

Description

This script extension is used to generate a path-based URL to a specified node. This is deprecated in preference of `wcmUrl` (see "[wcmUrl](#)" on page 9-7).

For more information, see "[Using Server-Side Script Links](#)" on page 4-3.

Parameters

- `nodeId`: A unique identifier for a node.
- `siteId`: A unique identifier for the Web site containing the node (optional).

Returns

The friendly link to the primary (or landing) page of a node.

Code Example

```
<!--$ssNodeLink("nodeId" , "siteId")-->  
<!--$ssNodeLink("nodeId")-->
```

9.39 ssWeblayoutUrl

Description

This script extension is used to determine the full web address of a file from either the path or *dDocName*. This is most typically used for paths to images in data files. This is deprecated in preference of *wcmUrl* (see "[wcmUrl](#)" on page 9-7).

For more information, see "[Using Server-Side Script Links](#)" on page 4-3.

Parameters

- *targetId*: Either the path to a file or the *dDocName* value of the file.

Returns

A path-based weblayout URL.

Code Example

```
<!--$ssWeblayoutUrl('mydocname')-->  
<!--$ssWeblayoutUrl('groups/public/documents/adacct/mydocname.jpg')-->
```

Idoc Script Variables

Site Studio uses several Idoc variables that are used to run a web site:

- ["HttpWebsitesRoot"](#) on page 10-1
- ["HttpRelativeWebsitesRoot"](#) on page 10-1
- ["HttpFragmentsRoot"](#) on page 10-2
- ["HttpRelativeFragmentsRoot"](#) on page 10-2
- ["SS_SERVER_NAME"](#) on page 10-2
- ["HttpASPPath"](#) on page 10-2
- ["ssServerRelativeSiteRoot"](#) on page 10-2

Note: These variables are subject to change with each release of Site Studio.

10.1 About Idoc Script Variables

Idoc script variables are used to give you more flexibility and customization when running your Web site. Many of the variables are used to store the HTTP location of different folders on the Oracle Content Server.

10.2 HttpWebsitesRoot

Defines the full HTTP location for the Site Studio websites folder. The typical location (where "server" is the name of the machine and "instance" is the name of the instance of the content server) is:

```
http://server/instance/websites/
```

10.3 HttpRelativeWebsitesRoot

Defines the relative HTTP location for the Site Studio websites folder. The typical location (where "instance" is the name of the instance of the content server) is:

```
instance/websites/
```

10.4 HttpFragmentsRoot

Defines the full HTTP location for the Site Studio fragments folder. The typical location (where "server" is the name of the machine and "instance" is the name of the instance of the content server) is:

```
http://server/instance/fragments/
```

10.5 HttpRelativeFragmentsRoot

Defines the relative HTTP location for the Site Studio fragments folder. The typical location (where "instance" is the name of the instance of the content server) is:

```
instance/fragments/
```

10.6 SS_SERVER_NAME

Defines the protocol, server name, and port number (if applicable) for the current machine. For example (where "server" is the name of the server):

```
http://server  
http://server:81  
https://server
```

10.7 HttpASPPath

Defines the relative path to the standard Site Studio *get_page.asp* file.

10.8 ssServerRelativeSiteRoot

Defines the root portion of a path-based URL for the current web site relative to the server. Will contain one of the following values:

- / - if the current web site is accessed using a friendly domain addressing mode
- /siteId/ - if the current web site is accessed using a friendly folder addressing mode

The `urlDirName` of the root node will override *siteId* if a value is present.

Site Studio Services

Site Studio introduces a number of new services in Oracle Content Server, which are used specifically to run a web site. The most important ones are detailed here.

This section contains the following topics:

- ["About Site Studio Services"](#) on page 11-1
- ["Services Related to Contributor"](#) on page 11-1
- ["Services Related to Designer"](#) on page 11-2
- ["Services Related to Manager"](#) on page 11-3
- ["Services Related to Switch Content"](#) on page 11-4
- ["Services Related to Link Wizard"](#) on page 11-4
- ["List of Services"](#) on page 11-4

Note: These services are used by the Site Studio Designer and Contributor applications as well as by the Site Studio component. They are subject to change with each release of the Site Studio application.

11.1 About Site Studio Services

A service is an HTTP request to the Oracle Content Server to perform an action. Each service has a defined set of actions (for example, database transactions, executing a piece of code, queries, and so forth).

When you install the Site Studio component, many services are added.

11.2 Services Related to Contributor

These services are used with Site Studio Contributor:

- ["SS_ADD_WEBSITE_ID"](#) on page 11-7
- ["SS_CHOOSE_WEBSITE_SECTION"](#) on page 11-8
- ["SS_CHOOSE_WEBSITES"](#) on page 11-8
- ["SS_CLEAR_PREVIEW"](#) on page 11-8
- ["SS_DECODE_LINK"](#) on page 11-10
- ["SS_EDIT_NATIVE_DOCUMENT"](#) on page 11-11

- ["SS_GET_CONFIG_INFO"](#) on page 11-13
- ["SS_GET_CONTRIBUTOR_CONFIG"](#) on page 11-13
- ["SS_GET_CONTRIBUTOR_STRINGS"](#) on page 11-14
- ["SS_GET_DOCUMENT_USAGE"](#) on page 11-14
- ["SS_GET_FRIENDLY_URL"](#) on page 11-15
- ["SS_GET_PLACEHOLDER_SWITCH_CONTENT_CONFIG"](#) on page 11-21
- ["SS_GET_SEARCH_RESULTS"](#) on page 11-22
- ["SS_GET_SWITCH_CONTENT_CONFIG"](#) on page 11-26
- ["SS_PUBLISH_THIS_PAGE"](#) on page 11-29
- ["SS_REMOVE_WEBSITE_ID"](#) on page 11-29
- ["SS_SET_ELEMENT_DATA"](#) on page 11-29
- ["SS_SET_PREVIEW_ELEMENT_DATA"](#) on page 11-31

11.3 Services Related to Designer

These services are related to Site Studio Designer:

- ["SS_ADD_NODE"](#) on page 11-7
- ["SS_ADD_WEBSITE_ID"](#) on page 11-7
- ["SS_CHECKIN_FRAGMENT_LIBRARY"](#) on page 11-8
- ["SS_CLEAR_PREVIEW"](#) on page 11-8
- ["SS_COMMIT_SITE_CHANGES"](#) on page 11-9
- ["SS_CREATE_NEW_SITE_EX2"](#) on page 11-9
- ["SS_CREATE_SITE_NAV_JS"](#) on page 11-10
- ["SS_DELETE_NODE"](#) on page 11-11
- ["SS_DOC_INFO_LATEST"](#) on page 11-11
- ["SS_EDIT_NATIVE_DOCUMENT"](#) on page 11-11
- ["SS_GET_ADMIN_PAGE"](#) on page 11-12
- ["SS_GET_ALL_CUSTOM_NODE_PROP_DEFS"](#) on page 11-12
- ["SS_GET_ALL_SITE_DOMAINS"](#) on page 11-12
- ["SS_GET_ALL_SITE_PROPERTIES"](#) on page 11-13
- ["SS_GET_ALL_SITES_EX2"](#) on page 11-13
- ["SS_GET_CONFIG_INFO"](#) on page 11-13
- ["SS_GET_DC_RULES"](#) on page 11-14
- ["SS_GET_DOCUMENT_USAGE"](#) on page 11-14
- ["SS_GET_ENVIRONMENT_PROPERTY_NAMES"](#) on page 11-15
- ["SS_GET_FRIENDLY_URL"](#) on page 11-15
- ["SS_GET_NODE_PROPERTY"](#) on page 11-18
- ["SS_GET_PAGE"](#) on page 11-18

- "SS_GET_REGION_DEFINITION_ELEMENTS" on page 11-22
- "SS_GET_SEARCH_RESULTS" on page 11-22
- "SS_GET_SITE_AS_XML_EX2" on page 11-23
- "SS_GET_SITE_ASSET_CATEGORIES" on page 11-24
- "SS_GET_SITE_CHANGE_MONITOR_TOKEN" on page 11-24
- "SS_GET_SITE_DEFINITION" on page 11-24
- "SS_GET_SITE_DEFINITION_FOR_USER" on page 11-24
- "SS_GET_SITE_DOMAINS" on page 11-25
- "SS_GET_SITE_FRAGMENT_ASSET_REPORT" on page 11-25
- "SS_GET_SITE_INFO" on page 11-25
- "SS_GET_SITE_PROPERTY" on page 11-25
- "SS_GET_SITE_REPORT" on page 11-26
- "SS_GET_UNIQUE_NODE_SITE_ID" on page 11-26
- "SS_GET_VERSION" on page 11-27
- "SS_IS_JS_NAV_OUT_OF_DATE" on page 11-27
- "SS_MAP_FRIENDLY_NAME" on page 11-27
- "SS_MOVE_NODE" on page 11-28
- "SS_PARSE_FRIENDLY_URL" on page 11-28
- "SS_PREPARE_PREVIEW" on page 11-28
- "SS_REMOVE_WEBSITE_ID" on page 11-29
- "SS_SET_ALL_CUSTOM_NODE_PROP_DEFS" on page 11-29
- "SS_SET_ENVIRONMENT_PROPERTY_NAMES" on page 11-30
- "SS_SET_NODE_PROPERTY" on page 11-30
- "SS_SET_NODES_PROPERTIES" on page 11-30
- "SS_SET_SITE_ASSET_CATEGORIES" on page 11-31
- "SS_SET_SITE_DOMAINS" on page 11-31
- "SS_SET_SITE_PROPERTIES" on page 11-32
- "SS_SET_SITE_PROPERTY" on page 11-32
- "SS_SWITCH_REGION_ASSOCIATION" on page 11-32
- "SS_VALIDATE_WEBSITE_OBJECT" on page 11-33

11.4 Services Related to Manager

These services are related to Site Studio Manager:

- "SS_ADD_NODE" on page 11-7
- "SS_DELETE_NODE" on page 11-11
- "SS_GET_ALL_CUSTOM_NODE_PROP_DEFS" on page 11-12
- "SS_GET_SEARCH_RESULTS" on page 11-22

- ["SS_GET_SITE_DEFINITION_FOR_USER"](#) on page 11-24
- ["SS_GET_SITE_PROPERTY"](#) on page 11-25
- ["SS_MOVE_NODE"](#) on page 11-28
- ["SS_SET_NODE_PROPERTY"](#) on page 11-30
- ["SS_SET_SITE_PROPERTY"](#) on page 11-32

11.5 Services Related to Switch Content

These services are related to the use of the Switch Content Wizard:

- ["SS_ADD_WEBSITE_ID"](#) on page 11-7
- ["SS_GET_ALL_NODE_PROPERTIES"](#) on page 11-12
- ["SS_GET_CONFIG_INFO"](#) on page 11-13
- ["SS_GET_CONTRIBUTOR_STRINGS"](#) on page 11-14
- ["SS_GET_DOCUMENT_LABELS"](#) on page 11-14
- ["SS_GET_LINK"](#) on page 11-16
- ["SS_GET_SEARCH_RESULTS"](#) on page 11-22
- ["SS_GET_SITE_DEFINITION_FOR_USER"](#) on page 11-24
- ["SS_GET_SWITCH_CONTENT_CONFIG"](#) on page 11-26
- ["SS_SET_NODE_PROPERTY"](#) on page 11-30
- ["SS_SET_NODES_PROPERTIES"](#) on page 11-30
- ["SS_SWITCH_REGION_ASSOCIATION"](#) on page 11-32

11.6 Services Related to Link Wizard

These services are related to constructing and modifying links in Site Studio:

- ["SS_ADD_WEBSITE_ID"](#) on page 11-7
- ["SS_DECODE_LINK"](#) on page 11-10
- ["SS_GET_CONTRIBUTOR_STRINGS"](#) on page 11-14
- ["SS_GET_FRIENDLY_URL"](#) on page 11-15
- ["SS_GET_LINK_WIZARD_CONFIG"](#) on page 11-16
- ["SS_GET_LINK_WIZARD_CONFIG_WITH_SITE"](#) on page 11-17
- ["SS_GET_SITE_DEFINITION_FOR_USER"](#) on page 11-24

11.7 List of Services

This list of services includes the most useful ones to customize Site Studio, especially to use it with third-party software:

- ["SS_ADD_NODE"](#) on page 11-7
- ["SS_ADD_WEBSITE_ID"](#) on page 11-7
- ["SS_BATCH_DECODE_LINK"](#) on page 11-7
- ["SS_CHECKIN_FRAGMENT_LIBRARY"](#) on page 11-8

- "SS_CHOOSE_WEBSITE_SECTION" on page 11-8
- "SS_CHOOSE_WEBSITES" on page 11-8
- "SS_CLEAR_PREVIEW" on page 11-8
- "SS_CLEAR_REGION_ASSOCIATIONS" on page 11-9
- "SS_CLEAR_WEBSITE_ID" on page 11-9
- "SS_COMMIT_SITE_CHANGES" on page 11-9
- "SS_CREATE_NEW_SITE_EX2" on page 11-9
- "SS_CREATE_SITE_NAV_JS" on page 11-10
- "SS_DECODE_LINK" on page 11-10
- "SS_DELETE_NODE" on page 11-11
- "SS_DOC_INFO_LATEST" on page 11-11
- "SS_EDIT_NATIVE_DOCUMENT" on page 11-11
- "SS_GET_ADMIN_PAGE" on page 11-12
- "SS_GET_ALL_CUSTOM_NODE_PROP_DEFS" on page 11-12
- "SS_GET_ALL_NODE_PROPERTIES" on page 11-12
- "SS_GET_ALL_SITE_DOMAINS" on page 11-12
- "SS_GET_ALL_SITE_PROPERTIES" on page 11-13
- "SS_GET_ALL_SITES_EX2" on page 11-13
- "SS_GET_CONFIG_INFO" on page 11-13
- "SS_GET_CONTRIBUTOR_CONFIG" on page 11-13
- "SS_GET_CONTRIBUTOR_STRINGS" on page 11-14
- "SS_GET_DC_RULES" on page 11-14
- "SS_GET_DOCUMENT_LABELS" on page 11-14
- "SS_GET_DOCUMENT_USAGE" on page 11-14
- "SS_GET_ENVIRONMENT_PROPERTY_NAMES" on page 11-15
- "SS_GET_FIRST_NODE_ID" on page 11-15
- "SS_GET_FRIENDLY_URL" on page 11-15
- "SS_GET_LINK" on page 11-16
- "SS_GET_LINK_MANAGEMENT_REPORT" on page 11-16
- "SS_GET_LINK_WIZARD_CONFIG" on page 11-16
- "SS_GET_LINK_WIZARD_CONFIG_WITH_SITE" on page 11-17
- "SS_GET_NODE_LINK" on page 11-17
- "SS_GET_NODE_PROPERTY" on page 11-18
- "SS_GET_PAGE" on page 11-18
- "SS_GET_PLACEHOLDER_SWITCH_CONTENT_CONFIG" on page 11-21
- "SS_GET_REGION_ASSOCIATIONS" on page 11-22
- "SS_GET_REGION_DEFINITION_ELEMENTS" on page 11-22

- ["SS_GET_RELATIVE_NODE_ID"](#) on page 11-22
- ["SS_GET_SEARCH_RESULTS"](#) on page 11-22
- ["SS_GET_SITE_AS_XML_EX2"](#) on page 11-23
- ["SS_GET_SITE_ASSET_CATEGORIES"](#) on page 11-24
- ["SS_GET_SITE_CHANGE_MONITOR_TOKEN"](#) on page 11-24
- ["SS_GET_SITE_DEFINITION"](#) on page 11-24
- ["SS_GET_SITE_DEFINITION_FOR_USER"](#) on page 11-24
- ["SS_GET_SITE_DOMAINS"](#) on page 11-25
- ["SS_GET_SITE_FRAGMENT_ASSET_REPORT"](#) on page 11-25
- ["SS_GET_SITE_INFO"](#) on page 11-25
- ["SS_GET_SITE_PROPERTY"](#) on page 11-25
- ["SS_GET_SITE_PUBLISH_REPORT"](#) on page 11-26
- ["SS_GET_SITE_REPORT"](#) on page 11-26
- ["SS_GET_SWITCH_CONTENT_CONFIG"](#) on page 11-26
- ["SS_GET_UNIQUE_NODE_SITE_ID"](#) on page 11-26
- ["SS_GET_VERSION"](#) on page 11-27
- ["SS_GET_WEBLAYOUT_URL"](#) on page 11-27
- ["SS_IS_JS_NAV_OUT_OF_DATE"](#) on page 11-27
- ["SS_MAP_FRIENDLY_NAME"](#) on page 11-27
- ["SS_MOVE_NODE"](#) on page 11-28
- ["SS_PARSE_FRIENDLY_URL"](#) on page 11-28
- ["SS_PREPARE_PREVIEW"](#) on page 11-28
- ["SS_PUBLISH_THIS_PAGE"](#) on page 11-29
- ["SS_REMOVE_WEBSITE_ID"](#) on page 11-29
- ["SS_SET_ALL_CUSTOM_NODE_PROP_DEFS"](#) on page 11-29
- ["SS_SET_ELEMENT_DATA"](#) on page 11-29
- ["SS_SET_ENVIRONMENT_PROPERTY_NAMES"](#) on page 11-30
- ["SS_SET_NODE_PROPERTY"](#) on page 11-30
- ["SS_SET_NODES_PROPERTIES"](#) on page 11-30
- ["SS_SET_PREVIEW_ELEMENT_DATA"](#) on page 11-31
- ["SS_SET_SITE_ASSET_CATEGORIES"](#) on page 11-31
- ["SS_SET_SITE_DOMAINS"](#) on page 11-31
- ["SS_SET_SITE_PROPERTIES"](#) on page 11-32
- ["SS_SET_SITE_PROPERTY"](#) on page 11-32
- ["SS_SWITCH_REGION_ASSOCIATION"](#) on page 11-32
- ["SS_VALIDATE_WEBSITE_OBJECT"](#) on page 11-33
- ["WCM_PLACEHOLDER"](#) on page 11-33

- ["WCM_EDIT_DATA_FILE"](#) on page 11-34
- ["WCM_BEGIN_EDIT_SESSION"](#) on page 11-34

11.7.1 SS_ADD_NODE

- **Description:** This service adds a child node to the given site node.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the unique identifier of the node (required). The newly added node will be a child of this node.
 - `newNodeId`: the identifier of the new child node (optional). If this value is not specified, a new, unique identifier will be generated.
 - `newNodeName`: the label of the new child node (optional). If this value is not specified, the text "New Section" will be used.
- **Returns:**
 - `newNodeId`: the identifier of the newly created node.
- **Security:** The user must have at least write access to the node to which a new child will be added.

11.7.2 SS_ADD_WEBSITE_ID

- **Description:** This service adds the given `siteId` to the list of web sites for the given document.
- **Parameters:**
 - `dDocName`: the `dDocName` (required).
 - `siteId`: the unique identifier of the site (required).
 - `fieldName`: either `xWebsites` or `xDontShowInListsForWebsite` (required).
- **Returns:** Nothing.
- **Security:** The user must have write access to the latest revision of the given `dDocName`.

11.7.3 SS_BATCH_DECODE_LINK

- **Description:** This service passes and returns a set to `SS_DECODE_LINK` to run on all listed links and `siteIds` in the set.
- **Parameters:**
 - a result set titled *Link* containing a list of links and `siteIds` as listed in `SS_DECODE_LINK`.
- **Returns:**
 - A result set titled *Links*, one row for each input row with information as described in `SS_DECODE_LINK`.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.4 SS_CHECKIN_FRAGMENT_LIBRARY

- **Description:** This service is a wrapper for CHECKIN_UNIVERSAL followed by a post-process where the fragment *.zip* file is unzipped and deployed to the weblayout.
- **Parameters:**
 - `primaryFile`: the fragment *.zip* file. Required.
 - `alternateFile`: the XML fragment definition. Required.
- **Returns:** None
- **Security:** Same requirements as for CHECKIN_UNIVERSAL.

11.7.5 SS_CHOOSE_WEBSITE_SECTION

- **Description:** This service is used to call a UI to select a section. It is used when creating a custom user interface and there is a need for a Web site section picker. This service has a Oracle Content Server template (in `\templates\`) called `ss_choose_website_section.htm`.
- **Parameters:**
 - `siteId`: The site ID for a Web site. Optional.
- **Returns:**
 - The UI as written in `ss_choose_website_section.htm`.
- **Security:** None.

11.7.6 SS_CHOOSE_WEBSITES

- **Description:** This service displays a screen that allows selecting from a list of web sites. This is used in conjunction with the customization of the *xWebsites* form elements on the content server UI.
- **Parameters:**
 - `xWebsites`: a comma-separated list of web site IDs.
- **Returns:** Nothing.
- **Security:** Anyone with read access to a security group within the content server will be able to invoke this service.

11.7.7 SS_CLEAR_PREVIEW

- **Description:** This service is used to clear a preview. Previews clear themselves each week if there is an issue with the clear not running; however, many prefer to remove the files from the server to save space.
- **Parameters:**
 - `previewId`: The preview ID (the folder name) in the preview cache.
- **Returns:** Nothing.
- **Security:** Write access to the document you are previewing.

11.7.8 SS_CLEAR_REGION_ASSOCIATIONS

- **Description:** This service clears the memory of the switched region associations.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the identifier of the node (optional). If this parameter is not specified, the associations for the entire site will be cleared.
 - `property`: either `primaryUrl` or `secondaryUrl` (required when `nodeId` is also specified).
- **Returns:** Nothing.
- **Security:** The user must have at least write access to the node or to the site if `nodeId` is not specified.

11.7.9 SS_CLEAR_WEBSITE_ID

- **Description:** This service clears the specified `siteId` from the `xWebsites` or `xDontShowInListsForWebsites` field for the given `dDocName`.
- **Parameters:**
 - `dDocName`: the *dDocName* of the file whose `xWebsites` or `xDontShowInListsForWebsites` field should be changed (required).
 - `removeWebsiteID`: the boolean value `yes` (required).
 - `fieldName`: The metadata field whose `siteId` should be cleared (either `xWebsites` or `xDontShowInListsForWebsites`) (required).
 - `siteId`: The *siteId* to remove from the `xWebsites` or `xDontShowInListsForWebsites` field (required).
- **Returns:** Nothing.
- **Security:** The user must have at least write access to the specified `dDocName`.

11.7.10 SS_COMMIT_SITE_CHANGES

- **Description:** This service commits site changes when run. Site Studio commits changes automatically every ten minutes, but this script can be run at any time to commit changes on demand.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:** Nothing.
- **Security:** You must have write access to the project file to execute the service.

11.7.11 SS_CREATE_NEW_SITE_EX2

- **Description:** This service creates a new, empty site on the content server
- **Parameters:**
 - `siteId`: the unique identifier of the site being created (required). The identifier must follow the `siteId` naming rules.
 - `siteLabel`: a label for the site (optional). If no label is specified, the text "Unnamed Site" will be used.

- `siteType`: the type of site being created. For example, "asp" or "idoc" (optional).
- `initialNodeId`: the *nodeId* of the root section (optional). If this value is not specified, a unique identifier will be created.
- `rootSectionActive`: indicates whether or not the root section should be marked as active (optional). Legal values are "TRUE" or "FALSE".
- `rootSectionLabel`: the label of the root section (optional). If this value is not specified, the value "Home" will be assigned.
- `rootSectionUrlPageName`: the *urlPageName* value for the root section (optional).
- `rootSectionUrlSecondaryPageName`: the *urlSecondaryPageName* for the root section (optional).
- `rootSectionUrlDirName`: the *urlDirName* value for the root section (optional).
- **Returns:**
 - `siteId`: the unique identifier of the created site (required).
- **Security:** The user executing the service must have write access to the security group used to check in Site Studio project files, as configured in the "Set Default Project Document Information" administration page.

11.7.12 SS_CREATE_SITE_NAV_JS

- **Description:** This service creates the site navigation files. This regenerates the *sitenavigation.js*, *sitenavigationfunctions.js*, *sitenavigation.xml*, *sitenavigation.hda*, and *sitenavigation_co.hda* files.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - `siteNavUrl`: the full HTTP URL to the site navigation file.
- **Security:** The user must have at least write access to the root node.

11.7.13 SS_DECODE_LINK

- **Description:** This service decodes any Site Studio link to determine where it resolves.
- **Parameters:**
 - `link`: the Site Studio link, in any supported format (required).
 - `siteId`: the unique identifier of the site (required).
- **Returns:** (may be blank)
 - `targetIsSection`: True/False
 - `nodeLabel`: the navigation label for the node.
 - `targetDocName`: the *dDocName* targeted by the link.
 - `targetNodeId`: the identifier of the section targeted by the link.
 - `targetSiteId`: the identifier of the site targeted by the link.

- targetIsSecondary: True/False
- linkType: the link type of the entered Site Studio link.
- errors: one of the following string values - unknown Error, noInputLink, invalidInputSiteId, invalidSiteId, invalidNodeId, invalidPathLink, noDocInfo, notSiteStudioUrl, badUrlFormat, cantFindUrlSection, parameterCount, parameterFormat.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.14 SS_DELETE_NODE

- **Description:** This service deletes the specified site node and all of its children.
- **Parameters:**
 - siteId: the unique identifier of the site (required).
 - nodeId: the identifier of the node to delete (required).
- **Returns:** Nothing.
- **Security:** The user must have at least delete access to the node to be deleted.

11.7.15 SS_DOC_INFO_LATEST

- **Description:** This service retrieves information about the latest document, including those in workflow, and not just the latest released document. This service behaves similar to the DOC_INFO_BY_NAME Oracle Content Server service.
- **Parameters:**
 - dDocName: the dDocName of the document you want to retrieve the info from.
- **Returns:**
 - The DOC_INFO result set for the latest version of the supplied dDocName.
 - The document author's email address. The email address is not in the result set.
- **Security:** Read access to the latest version. This may require access to the document's workflow .

11.7.16 SS_EDIT_NATIVE_DOCUMENT

- **Description:** This service returns an HTML page that can be used to launch "Check Out and Open" (COAO). The page has the necessary JavaScript and <object> tag required to launch COAO.
- **Parameters:**
 - dDocName: the dDocName of the file that you want to edit with the COAO functionality (required).
- **Returns:** Nothing.
- **Security:** Must have write access on at least one security group; otherwise, the user would not be able to use COAO.

11.7.17 SS_GET_ADMIN_PAGE

- **Description:** This service displays the root Site Studio Administration page.
- **Parameters:** None.
- **Returns:**
 - The Administration page
- **Security:** Requires administration access.

11.7.18 SS_GET_ALL_CUSTOM_NODE_PROP_DEFS

- **Description:** Retrieves the custom node property definitions for the given site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - A result set named *CustomNodePropertyDefs*, which has the following columns: *name*, *type*, and *description*.
- **Security:** The user must have at least write access to the root node.

11.7.19 SS_GET_ALL_NODE_PROPERTIES

- **Description:** Retrieves the complete set of node properties.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the identifier of the node (required).
- **Returns:**
 - Returns two result sets, *StandardProperties* and *SiteStudioProperties*. Each result set has one row. The column names are the property names.
 - The *StandardProperties* result set will contain only one row and column reporting the *nodeId*.
- **Security:**
 - The user must have at least read access to the node's folder.
 - This service can be executed from Idoc script.

11.7.20 SS_GET_ALL_SITE_DOMAINS

- **Description:** This service retrieves a complete set of the domains on the server.
- **Parameters:** None.
- **Returns:**
 - A result set named *DomainMap* of all domain addresses with rows named *Key*, *SiteId*, *Patterns*, and *Default*. *Key* is the domain, *SiteId* is the ID, *Pattern* is what the URL is re-written to as an *SS_GET_PAGE*, and *Default* is the default address to the site.
- **Security:** Requires Administrator access.

11.7.21 SS_GET_ALL_SITE_PROPERTIES

- **Description:** Retrieves the complete set of site properties.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - A result set named *SiteStudioProperties*, which has one row of values. The column names are the property names.
- **Security:**
 - The user must have at least read access to the site.
 - This service can be executed from Idoc script.

11.7.22 SS_GET_ALL_SITES_EX2

- **Description:** This service returns a list of all of the sites on the content server.
- **Parameters:** None.
- **Returns:**
 - `numSites=N`
 - `site0=siteId`
 - ...
 - `siteN-1=siteId`The same information will be returned in the *SiteIds* result set.
- **Security:** The list of sites returned will be those to which the user has at least read access.

11.7.23 SS_GET_CONFIG_INFO

- **Description:** This service gets all Site Studio base configuration information for Designer.
- **Parameters:** None.
- **Returns:**
 - The base configuration information.
- **Security:**
 - The user must have at least read access to the site.
 - This service can be executed from Idoc script.

11.7.24 SS_GET_CONTRIBUTOR_CONFIG

- **Description:** This service retrieves the Site Studio base configuration information for Contributor.
- **Parameters:** None.
- **Returns:**
 - The base configuration information.

- **Security:**
 - The user must have at least read access to the site.
 - This service can be executed from Idoc script.

11.7.25 SS_GET_CONTRIBUTOR_STRINGS

- **Description:** This service loads a JavaScript file (*wcm.strings.js*) from `<weblayout>\resources\wcm\base\lang\[language id]` for the current user's language. If there is no *wcm.strings.js* file for the user's current language, the default EN strings will be returned.
- **Parameters:** None
- **Returns:**
 - The contents of the JavaScript file.
- **Security:** None. If the user is not logged in, the language ID is EN.

11.7.26 SS_GET_DC_RULES

- **Description:** This service returns a result set of all defined document conversion rules.
- **Parameters:** None
- **Returns:**
 - A result set of all defined rules for Dynamic Converter named either *DCCConversions* or *DCCConversions80* depending on the version of Dynamic Converter.
- **Security:** The user must have at least write access to the site.

11.7.27 SS_GET_DOCUMENT_LABELS

- **Description:** This service is used by Contributor to get labels from a content ID.
- **Parameters:**
 - A result set of *dDocNames* named *contentIds*.
- **Returns:**
 - A label for each of the supplied *dDocName* values. The response will be in JSON format with the *dDocName* values as the keys.
- **Security:** The user must have at least write access.

11.7.28 SS_GET_DOCUMENT_USAGE

- **Description:** This service displays the Web site usage report.
- **Parameters:**
 - *dDocName*: the *dDocName* of the file to run the Web site usage report for (required).
- **Returns:**
 - The Web site usage report.
- **Security:** The user must have at least read access.

11.7.29 SS_GET_ENVIRONMENT_PROPERTY_NAMES

- **Description:** This service retrieves the properties of this site that are identified as environment properties.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - A result set called *EnvironmentProperties* with a single column called *name* that has a row for each property identified as an environment property.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.30 SS_GET_FIRST_NODE_ID

- **Description:** This service retrieves the first node of the site; useful for enumerating the site hierarchy.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - `firstId`: the identifier of the first node in the site hierarchy.
- **Security:**
 - The user must have at least read access to the root node.
 - This service can be executed from Idoc script.

11.7.31 SS_GET_FRIENDLY_URL

- **Description:** This service constructs a friendly URL to either a document or a section.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `ssDocName`: the *dDocName* of the document to display (optional).
 - `ssTargetNodeId`: the identifier of the section that will display the document (optional).
Either *ssDocName* or *ssTargetNodeId* must be specified.
 - `ssTargetSiteId`: the identifier of the site containing the specified *ssTargetNodeId*.
- **Returns:**
 - `ssFriendlyUrl`: The site-relative URL to the specified document or section.
 - `HttpSiteAddress`: The address of the root of the Web site. (The full URL can be constructed by concatenating these two return values.)
- **Security:** The user must have write access to at least one security group on the content server.

11.7.32 SS_GET_LINK

- **Description:** This is the service version of ssLink server side link.
- **Parameters:**
 - `ssDocName`: the *dDocName* of the document to display (required).
 - `TargetNodeId`: the identifier of the section that will display the document (optional).
 - `TargetSiteId`: the identifier of the site containing the specified *TargetNodeId* (optional).
 - `SourceNodeId`: the identifier of the section that will display the document if the document's *xWebsiteSection* value is not available (optional).
 - `SourceSiteId`: the identifier of the site containing the specified *ssSourceNodeId* (optional).
- **Returns:**
 - An *ssLink* parameter denoting an absolute URL to the specified document.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.33 SS_GET_LINK_MANAGEMENT_REPORT

- **Description:** This service obtains information about links within a Site Studio Site.
- **Parameters:** none.
- **Returns:** a result set labeled *Manifest* with these columns:
 - `siteId`: the unique identifier of the site (required).
 - `layoutResultSet`: contains these values:
 - * `nodeId`: the identifier of the section.
 - * `dDocName`: the name of the document.
 - * `isPrimaryUrl`: if the result is designated as a primary page.
 - `urlDataFiles`: contains these values:
 - * `nodeId`: the identifier of the section.
 - * `dDocName`: the name of the document.
 - * `isPrimaryUrl`: if the result is designated as a primary page.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.34 SS_GET_LINK_WIZARD_CONFIG

- **Description:** This service obtains the configuration information used to populate the Link Wizard.
- **Parameters:**
 - `link`: the value of the link. It can be of any format.
 - `target`: the target window.

Note: While these parameters are passed in the service call, they are not used.

- **Returns:**
 - The configuration in JSON for the link wizard is returned.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.35 SS_GET_LINK_WIZARD_CONFIG_WITH_SITE

- **Description:** This service is used to launch the link wizard within the context of a site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the unique identifier of the node.
 - `link`: the value of the link. It can be of any format.
 - `target`: the target window.

Note: While the `nodeId`, `link`, and `target` parameters are passed in the service call, they are not used.

- **Returns:**
 - The configuration in JSON for the link wizard is returned.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.36 SS_GET_NODE_LINK

- **Description:** This is the service version of *ssNodeLink* server side link.
- **Parameters:**
 - `TargetNodeId`: the identifier of the section to construct the URL to (required).
 - `TargetSiteId`: the identifier of the site containing the specified *TargetNodeId* (optional).
 - `SourceNodeId`: the identifier of the section from which the URL should be constructed. This is useful for ascertaining if the link should be an absolute or relative URL (optional).
 - `SourceSiteId`: the identifier of the site containing the specified *SourceNodeId* (optional).
- **Returns:**
 - The result is returned as an *ssNodeLink* parameter.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.37 SS_GET_NODE_PROPERTY

- **Description:** This service retrieves a node property.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the identifier of the node (required).
 - `property`: the name of the property to remove (required).
- **Returns:**
 - `value`: the value of the requested node property. If the requested property does not exist, this parameter will not be returned but the service call will succeed.
- **Security:**
 - The user must have at least read access to the node.
 - This service can be executed from Idoc script.

11.7.38 SS_GET_PAGE

The core service "SS_GET_PAGE" was created specifically for Site Studio. It performs a central function for the product by dynamically generating a single web page from its many component parts.

In Site Studio version 7.2.1 and earlier versions, the SS_GET_PAGE service was used directly to browse to a Site Studio web page URL. For the current version of Site Studio version, the SS_GET_PAGE service will rarely be used in the web page URL directly. Instead, web pages are accessed using more standard path-based URLs.

However, the core SS_GET_PAGE service call still exists to perform the work behind the scenes for these URLs. In addition, we now have a web server filter plug-in that intercepts all path-based Site Studio related URLs and converts them to their component parts before passing them on to the underlying SS_GET_PAGE service call. As a result, this core service continues to be described here.

The SS_GET_PAGE service requires one of the following, mutually exclusive parameters as part of the query string:

- `nodeId`: If specified, the service looks for that node in the site hierarchy and retrieves the node's `primaryUrl` property, which provides enough information to serve up the node's primary page directly.

It is also possible to retrieve the node's secondary page, instead, by specifying the `"useSecondary=true"` parameter in the URL. In this case, the service will retrieve the node's `secondaryUrl` property and serve up the secondary page. This feature is used primarily by the Designer application.
- `ssDocName`: If specified (with a valid `dDocName` value), the service looks for the managed content item and determines the web site section that it is to be displayed in using a three-rule evaluation (see "Three Rules for Reusing Content" below).

It then performs a decision-making process to determine whether it should display the section's primary or secondary page by parsing the sections `primaryUrl` and `secondaryUrl` properties. Once the decision is made, the correct page can be served up.

In the Site Studio Designer interface and in the *User's Guide for Site Studio Designer*, a site hierarchy is described as having "sections" that make up the web site. These were originally called "nodes" when the product was first developed, and, as such, many behind-the-scenes features like the SS_GET_PAGE service still refer to them as such. In Site Studio terminology, a "node" is the same as a "section."

Three Rules for Reusing Content

When SS_GET_PAGE is used with the `nodeId` parameter, the service is told explicitly to display the primary page of the specified section, and no more decisions are needed. But when SS_GET_PAGE is used with the `ssDocName` parameter, the service must determine which section to display the document in, and this is determined by the following three rules:

1. **TARGET:** If the optional `ssTargetNodeId` parameter is specified in the URL, then it explicitly displays the managed document in that section.
2. **SECTION:** If the managed document has a value in its `xWebsiteSection` metadata field, then it uses that value as the section to display the managed document in.
3. **SOURCE:** If the optional `ssSourceNodeId` parameter is specified in the URL as the current pages node, then it explicitly displays the managed document in that section.

All of this allows contributors to share and reuse content (contributor data files and native documents) in different sections of the same site and even different sites on the same content server (see "Sharing region content using target sections" in the *User's Guide for Site Studio Designer*).

Once the SS_GET_PAGE service (using an `ssDocName` parameter) has determined which section to display the managed document in, it still needs to determine whether it should serve up the primary or secondary layout page. The following algorithm controls this:

Condition	Action
If <code>ssDocName</code> matches any of the region parameters in the sections <code>primaryUrl</code> property	Serve up the primary page.
If <code>ssDocName</code> matches any of the region parameters in the section's <code>secondaryUrl</code> property	Serve up the secondary page.
Else	Serve up the secondary page but replace the REPLACEABLE region parameter with the <code>ssDocName</code> value. The REPLACEABLE region is defined by the sections <code>secondaryUrlVariableField</code> property.

The above description assumes that the `ssDocName` parameter points to a contributor data file or native document. If the `ssDocName` parameter points to a layout file, then a different decision-making process occurs, whereby Site Studio walks the site hierarchy until it locates a section using that layout. However, a direct link to a managed layout is rare; it is typically used internally by Designer.

In addition to `nodeId`, `ssDocName`, `ssTargetNodeId`, and `ssSourceNodeId`, the SS_GET_PAGE service also recognizes the following optional URL parameters or cookie values:

- **SSContributor:** This parameter contains a true or false value to indicate whether the web page should be displayed in contribution mode. In contribution mode, the web page displays workflow versions of content items and a contribution icon for each contribution region. If necessary, this parameter will also cause a login prompt. If passed as a URL parameter, this value will automatically get set as a cookie value as well so that it is automatically propagated as the contributor navigates the site.
- **PreviewId:** This parameter indicates that a temporary preview version of the web page should display instead of the latest checked-in version. It is used only by the Designer and Contributor applications (in combination with the SS_GET_PAGE service) to provide a preview service without checking in new versions of layout pages and contributor data files. Again, if passed as a URL parameter, this value will automatically get set as a cookie value so that it is automatically propagated as the contributor or designer navigates the site in order to remain in preview mode.

You should not be using this parameter directly.

The Error Handler Section

An error can occur at any point during the SS_GET_PAGE service call. For example, if a REPLACEABLE region is required but has not been specified for a particular section, you may encounter an error. Normally, if these types of errors occur, they are shown within a standard content server error page, which takes the consumer out of the context of the web site.

In order to show the error, but remain within the context of the web site, one section within the site can be specified as an "Error Handler" section in Designer. If an error occurs within the SS_GET_PAGE service, then the consumer is redirected to the primary layout page associated with the error handler section. Two Idoc variables are available for the layout page displaying error information:

- **ssErrorMessage:** a text description of the error that has occurred.
- **ssErrorCode:** a numeric error code to indicate what type of error has occurred so that you can present your own error description.

Internally, when setting the error handler section, a site property, `errorNodeId`, is set to contain the `nodeId` of the error handler section.

In Site Studio, the potential list of error codes includes:

ssErrorCode	ssErrorMessage
-0x100	"No Layout page specified for this part of the web site."
-0x101	"Failed to locate document information for document with content ID"
-0x102	"Document with Content ID '{1}' does not match the Primary or Secondary Url at section '{2}' (Id={3}) and there is no Replaceable Region defined."
-0x103	"Link to Section '{1}' (Id={2}) failed because there is no Primary URL defined for the section."
-0x104	"Link to Section '{1}' (Id={2}) failed because there is no Secondary URL defined for the section."
-0x105	"The Section '{1}' (Id={2}) is not part of a Site Studio web site."
-0x106	"Layout with Content ID '{1}' is not filed in a Site Studio web site."
-0x107	"The Layout with Content ID '{1}' was not found in any section Url of any web site."

ssErrorCode	ssErrorMessage
-0x108	"Unable to identify in which web site section to display document with Content ID '{1}'."
-0x200	"An unknown error has occurred in the SS_GET_PAGE service call."

- **Description:** Displays a Site Studio web page.
- **Parameters:**
 - To display the primary (splash) page of a section:
 - `siteId`: the identifier of the site (optional). This value is computed if not specified.
 - `nodeId`: the identifier of the section (required).
 - To display a particular document in a section:
 - `ssDocName`: the `dDocName` of the document to display (required).
 - `ssTargetNodeId`: the identifier of the section that will display the document (optional).
 - `ssTargetSiteId`: the identifier of the site containing the specified `ssTargetNodeId` (optional). This value is computed if not specified.
 - `ssSourceNodeId`: the identifier of the section that will display the document if the document's `xWebsiteSection` value is not available (optional).
 - `ssSourceSiteId`: the identifier of the site containing the specified `ssSourceNodeId` (optional). This value is computed if not specified.
 - `SSContributor`: indicates whether the page should be delivered in contribution mode (optional). Allowed values are "true" and "false".
- **Returns:**
 - The Site Studio web page.
- **Security:** The user must have at least read access to the documents that comprise the web site page.

11.7.39 SS_GET_PLACEHOLDER_SWITCH_CONTENT_CONFIG

- **Description:** This service collects all configuration necessary for the Oracle Content Wizard for a given placeholder definition.
- **Parameters:**
 - This requires one of the following:
 - `contentId`: the unique identifier of the placeholder definition.
 - Or
 - The placeholder definition xml file passed as a string.
- **Returns:** Returns a result set for each of the following:
 - `RegionTemplates`: a result set of the region templates.
 - `RegionDefinitions`: a result set of the region definitions.
 - `Subtemplates`: a result set of the subtemplates.
- **Security:** The user must have write access.

11.7.40 SS_GET_REGION_ASSOCIATIONS

- **Description:** This service returns region associations of primary and secondary URLs that have been changed via the *SS_SWITCH_REGION_ASSOCIATION* service URLs.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - `Primary`: all primary URLs for the site.
 - `Secondary`: all secondary URLs for the site.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.41 SS_GET_REGION_DEFINITION_ELEMENTS

- **Description:** This service retrieves all element definitions listed in a region definition.
- **Parameters:**
 - `regionDefinition`: the unique identifier of the region definition.
- **Returns:**
 - A result set named *ssElementConfigs* containing the following columns: *SSElementType*, *SSElementTitle*, *SSElementName*, *SSElementIsList*, *SSSubelements*.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.42 SS_GET_RELATIVE_NODE_ID

- **Description:** This service retrieves the node identifier of the given relative; useful for enumerating the site hierarchy.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the identifier of the node (required).
 - `relative`: one of the following: `parent`, `child`, `next`, or `prior` (required).
- **Returns:**
 - `relativeId`: the identifier of the relative node in the site hierarchy. If there is no such relative, the empty string is returned.
- **Security:**
 - The user must have at least read access to the specified node.
 - This service can be executed from Idoc script.

11.7.43 SS_GET_SEARCH_RESULTS

- **Description:** This service is a wrapper around the regular *GET_SEARCH_RESULTS* service. It allows some Site Studio features to be specified simply as flags and have the real query syntax be constructed on the server. It can modify

the standard *SearchResults* result set to contain an additional column called *ssUrl* which will contain the Site Studio-friendly url for all of the search results.

- **Parameters:**
 - *siteId*: the unique identifier of the site (required).
 - *ssLimitScope*: this restricting scope to within the site only (optional). It is *true*, *false*, or *-1*.
 - True* alters the *QueryText* with a clause for *xWebsites*
 - False* does not alter the *QueryText* for *xWebsites*
 - 1* alters the *QueryText* with a *<not>* clause for *xWebsites*
 - *ssUserSearchText*: a user-supplied search text (optional). This is joined to the supplied *QueryText*.
 - *ssWebsiteObjectType*: the object type (optional). This appends to the *QueryText* the requirement that *xWebsiteObjectType* match the specified object type.
 - *computeFriendlyUrls*: this modifies results to include *ssUrl* (optional). This parameter defaults to *true*.
 - *ssDontShowInLists*: this is used to limit the search to those documents where the *siteId* is contained in the *xDontShowInListsForWebsites* metadata field. This parameter is *true/false*.

These parameters are all in addition to those required by *GET_SEARCH_RESULTS*.

- **Returns:**
 - A result set named *Search Results*.
- **Security:** The user must have read access to at least one security group on the content server.

11.7.44 SS_GET_SITE_AS_XML_EX2

- **Description:** This service retrieves the entire site with nodes and attributes as XML.
- **Parameters:**
 - *siteId*: the unique identifier of the site (required).
 - *includeProperties*: A Boolean parameter (*True* or *False*) that indicates whether the properties of the nodes should be included as result sets in the response (optional).
- **Returns:**
 - *siteXml*: an XML representation of the site.
 - Optionally returns two result sets, *StandardProperties* and *SiteStudioProperties*, which give the properties of the nodes in the XML file. A *nodeId* parameter associates rows in the result set to the nodes of the XML.
- **Security:** This requires that the user have at least write access to the root node of the hierarchy. The returned XML will enumerate every node in the hierarchy. This is because Site Studio has to be able to show a tree structure that the user can navigate through to get to nodes that he can modify.

11.7.45 SS_GET_SITE_ASSET_CATEGORIES

- **Description:** This service retrieves the site asset categories for a site from the project file.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - A result set named *SiteAssetCategories* containing the following columns: *name*, *type*, *querytext*, *metadata*, and *description*.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.46 SS_GET_SITE_CHANGE_MONITOR_TOKEN

- **Description:** This is a 'heartbeat' service that Designer normally runs every 10 seconds by default to determine if changes have been made to the site by someone else.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - `ssChangeMonitorToken`: an identifier that changes when the web site changes.
- **Security:**
 - The user must have at least read access to the project root element.
 - This service can be executed from Idoc script.

11.7.47 SS_GET_SITE_DEFINITION

- **Description:** This service returns the site definition in XML.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - `siteXml`: an XML representation of the site hierarchy.
- **Security:** The user must have write access to the project root element.

11.7.48 SS_GET_SITE_DEFINITION_FOR_USER

- **Description:** This service returns the site definition in XML, showing all nodes a user has read access to.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:** Nothing.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.49 SS_GET_SITE_DOMAINS

- **Description:** This service returns domain mapping information for a site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - `ssSiteDomains`: a result set listing the domains that are mapped to the site.
- **Security:** The user must have write access to one security group on the content server.

11.7.50 SS_GET_SITE_FRAGMENT_ASSET_REPORT

- **Description:** This service lists all information for assets being used in all fragments in a site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - The information for the assets is returned. The same information is displayed in the Administration page.
- **Security:** The user must have read access and be logged in.

11.7.51 SS_GET_SITE_INFO

- **Description:** This service returns the *dDocName* of the project file for a site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:**
 - The *dDocName* of the project file.
- **Security:** The user must have read access.

11.7.52 SS_GET_SITE_PROPERTY

- **Description:** This service retrieves a site property.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `property`: the name of the property to retrieve (required).
- **Returns:**
 - `value`: the value of the requested site property.

If the requested site property does not exist, this parameter will not be returned, but the service call will succeed.
- **Security:** The user must have at least read access to the site. This service can be executed from Idoc script.

11.7.53 SS_GET_SITE_PUBLISH_REPORT

- **Description:** This service obtains a report of the items used in the site. This is designed to be used with the Site Studio Publishing Utility (SSPU) or Site Studio Publisher to ensure the fidelity of the "scraped" site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:** Returns numerous result sets describing the site content.
- **Security:** The user must have at least write access to the site's root folder.

11.7.54 SS_GET_SITE_REPORT

- **Description:** This service obtains a report of the items used in the site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
- **Returns:** Returns numerous result sets describing the site content.
- **Security:** The user must have at least write access to the site.

11.7.55 SS_GET_SWITCH_CONTENT_CONFIG

- **Description:** This service returns the Switch Content configuration for the listed region definition.
- **Parameters:**
 - `regionDefinition`: the unique identifier of the region definition (required).
- **Returns:**
 - A JSON response of the XML of the region definition named *ssRegionConfig*.
- **Security:** The user must have at least write access to the site.

11.7.56 SS_GET_UNIQUE_NODE_SITE_ID

- **Description:** This service returns what site a given node ID is associated with. This should be used with servers set up to use unique nodes.
- **Parameters:**
 - `nodeId`: the unique identifier of the node, to determine what site it is associated with (required).
 - `preferredSiteId`: the unique identifier of the site to search first. This is used in those cases where you might think the nodes may not be unique.
- **Returns:**
 - `siteId`: the unique identifier of the site. If the *nodeId* could not be found in any site, the service will return a `StatusCode` with a negative value.
- **Security:**
 - The user must have at least read access to the specified node.
 - This service can be executed from Idoc script.

11.7.57 SS_GET_VERSION

- **Description:** This service returns the version number and build number for Site Studio as well as the version number for Oracle Content Server.
- **Parameters:** None
- **Returns:**
 - Product version number and build number for the Site Studio component, and the version number for the Oracle Content Server.
- **Security:**
 - The user must have at least read access to the specified node.
 - This service can be executed from Idoc script.

11.7.58 SS_GET_WEBLAYOUT_URL

- **Description:** This service returns the full weblayout url in the parameter labeled *ssWeblayoutUrl*.
- **Parameters:**
 - *ssWebLayoutParam*: either *dDocName*, or a weblayout url starting with the path groups/ (required).
- **Returns:**
 - A full weblayout url in a parameter labeled *ssWeblayoutUrl*.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.59 SS_IS_JS_NAV_OUT_OF_DATE

- **Description:** This service determines if the navigation files are out of date.
- **Parameters:**
 - *siteId*: the unique identifier of the site (required).
- **Returns:**
 - Variable *jsNavIsOutOfDate* of 0 or 1.
- **Security:**
 - The user must have at least read access.
 - This service can be executed from Idoc script.

11.7.60 SS_MAP_FRIENDLY_NAME

- **Description:** This service maps either a *dDocName* value to a friendly name or a friendly name to a *dDocName* - this relates to use of the *SSUrlFieldName* configuration setting.
- **Parameters:**
 - *inputName*: value of either the *dDocName* or the friendly name (required).
 - *direction*: either *fromDocName* or *toDocName* (required).
- **Returns:** Nothing.

- **Security:** The user must have write access to at least one security group on the content server.

11.7.61 SS_MOVE_NODE

- **Description:** This service moves a site node in the hierarchy.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the identifier of the node to move (required).
 - `newParentId`: the identifier of the new parent for the node (required).
 - `insertAfterId`: the identifier of the prior sibling to the moved node (optional). If this parameter is not specified, the node will be the first child of *new_parent_id*.
- **Returns:** Nothing.
- **Security:** The user must have at least write access to the new parent node.

11.7.62 SS_PARSE_FRIENDLY_URL

- **Description:** This service returns the *siteId* and *siteRelativeUrl* for the target of the link passed.
- **Parameters:**
 - `ssFriendlyUrl`: the friendly url located on the site (required).
- **Returns:**
 - `siteId`: the unique identifier of the site.
 - `siteRelativeUrl`: the parsed url of the entered friendly url.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.63 SS_PREPARE_PREVIEW

- **Description:** The preview of a file is a temporary file in a directory under the directory of the actual checked-in item. This service creates a preview ID for a given file.
- **Parameters:**
 - `path`: the local file system path of the file to be previewed.
 - `ssDocName`: the content ID of the item to preview.
- **Returns:**
 - The Preview ID.
- **Security:** The user must have write access to the file you are trying to preview.

11.7.64 SS_PUBLISH_THIS_PAGE

- **Description:** This service is the same as the Publish Now functionality used to interact with Site Studio Publisher to publish the URL for the node.

For this service to work with a site, the site must be marked with the *Enable Publish Site* action in Site Studio Designer to work.

- **Parameters:**
 - `nodeId`: the unique identifier of the node. (required)
 - `isSecondaryPage`: A boolean value. The default is false.
- **Returns:** Nothing.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.65 SS_REMOVE_WEBSITE_ID

- **Description:** This service removes the given `siteId` to the list of web sites for the given document.
- **Parameters:**
 - `dDocName`: the unique identifier for the file (required).
 - `siteId`: the unique identifier of the site (required).
 - `fieldName`: either `xWebsites` or `xDontShowInListsForWebsite` (required).
- **Returns:** Nothing.
- **Security:** The user must have write access to the latest revision of the given `dDocName`.

11.7.66 SS_SET_ALL_CUSTOM_NODE_PROP_DEFS

- **Description:** This service sets the custom node property definitions for the given site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required). Takes as input a result set named *CustomNodePropertyDefs*, which has the following columns: *name*, *type*, and *description* (required).
- **Returns:** Nothing.
- **Security:** The user must have at least write access to the root node.

11.7.67 SS_SET_ELEMENT_DATA

- **Description:** This service is used by Contributor to update and save changes. This is called at the end of a revision. The checkout of the data file is independent of this service; the data file must be checked out before it can be checked in with changes.

- **Parameters:**
 - `dDocName`: the unique identifier of the data file (required).
 - `dId`: the ID of the revision checked out (required).

The two above values are obtained with the check-out of the data file.
 - A result set named *SSElementData*, with the following columns: *element*, *value*, and *isList*. The *isList* is boolean, so if it is yes, then an additional result set with the name of the element and the columns named with the list sub-elements.
- **Returns:** Nothing.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.68 SS_SET_ENVIRONMENT_PROPERTY_NAMES

- **Description:** This service sets the defined environment properties for the specified site.
- **Parameters:**
 - *siteId*: the unique identifier of the site (required).
 - a result set called *EnvironmentProperties* with a single column called *name* that has a row for each property identified as an environment property.
- **Returns:** Nothing.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.69 SS_SET_NODE_PROPERTY

- **Description:** This service sets a node property.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the unique identifier of the node (required).
 - `property`: the name of the property to set or remove (required).
 - `value`: the new value of the attribute (optional). If the value is not specified, the property will be removed.
- **Returns:** Nothing.
- **Security:** The user must have at least write access to the node.

11.7.70 SS_SET_NODES_PROPERTIES

- **Description:** This service is a method to set multiple node properties on multiple nodes.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId0`: the unique identifier of the node (or nodes) (required).
 - ...
 - `nodeIdN`

- `property0`: the name of the property (or properties) (required).
...
`propertyN`
- `value0`: the value of the associated property (required).
...
`valueN`
- **Returns:**
 - A *rowsProcessed* value of the number of properties set.
- **Security:** The user must have at least write access to the node.

11.7.71 SS_SET_PREVIEW_ELEMENT_DATA

This service is a specialization of `SET_ELEMENT_DATA`, where the data is saved in a preview location and not checked in.

See [Section 11.7.67, "SS_SET_ELEMENT_DATA"](#) for more information.

11.7.72 SS_SET_SITE_ASSET_CATEGORIES

- **Description:** This service is used to save changes to the site asset categories.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - A result set called *SiteAssetCategories* with the following columns: *description*, *metadata*, *name*, *querytext*, and *type*. Values for *name* and *type* must be supplied. These values match the descriptions detailed in the Site Asset Categories dialog in the *User's Guide for Site Studio Designer*.
- **Returns:** Nothing.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.73 SS_SET_SITE_DOMAINS

- **Description:** This service sets the domain mapping information for a site.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `ssSiteDomains`: a result set listing the domains to be mapped to the site.
- **Returns:** Nothing.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.74 SS_SET_SITE_PROPERTIES

- **Description:** This service sets multiple site properties in a single service call.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `property`: the name of the property to retrieve (required).
 - `value`: the value of the requested site property.
- **Returns:** Nothing.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.75 SS_SET_SITE_PROPERTY

- **Description:** This service sets a site property.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `property`: the name of the property to set or remove (required).
 - `value`: the new value of the attribute (optional). If value is not specified, the property will be removed.
- **Returns:** Nothing.
- **Security:** The user must have at least write access to the site.

11.7.76 SS_SWITCH_REGION_ASSOCIATION

- **Description:** This service changes the *dDocName* associated with a particular region in a particular node.
- **Parameters:**
 - `siteId`: the unique identifier of the site (required).
 - `nodeId`: the identifier of the node (required).
 - `region`: the region identifier (required).
 - `primaryUrl`: the *dDocName* value to set for the specified "region" in the primaryUrl (optional).
 - `secondaryUrl`: the *dDocName* value to set for the specified "region" in the secondaryUrl (optional).
 - `primaryTemplateUrl`: the *dDocName* value of a region template or subtemplate to set for the specified "region" in the primary page (optional).
 - `secondaryTemplateUrl`: the *dDocName* value of a region template or subtemplate to set for the specified "region" in the secondary page (optional).
- **Returns:** Nothing.
- **Security:** The user must have at least write access to the node.

11.7.77 SS_VALIDATE_WEBSITE_OBJECT

- **Description:** This service validates a website object (in XML) against the known XSD Schema for the object type.
- **Parameters:**
 - `websiteObject:path`: the path to the local object file to validate and upload.
 - `websiteObjectType`: the the object type of the local object file.
- **Returns:**
 - A binder response with message and status code.
- **Security:** The user must have write access to at least one security group on the content server.

11.7.78 WCM_PLACEHOLDER

This service evaluates a placeholder, allowing the contents of the placeholder to be retrieved directly from anywhere the Oracle Content Server can be seen from.

This allows you to create a third party application on top of xml data files that are managed by the Oracle Content Server and can be edited by the Site Studio Contributor, without being forced into using a full Site Studio Web site.

- **Parameters:**
 - `dataFileDocName`: the *dDocName* of the data file to associate with the placeholder.
 - `templateDocName`: the *dDocName* of the region template or subtemplate to associate with the placeholder.
 - `placeholderDefinitionDocName`: the *dDocName* of the placeholder definition to map to the placeholder.
 - `regionDefinitionDocName`: the *dDocName* of the region definition to associate with the region template named in `templateDocName`.
 - `placeholderActions`: the allowed actions of the placeholder definition, as follows:
 - E allows contributor update
 - P allows workflow approve
 - R allows workflow reject
 - I allows viewing docInfo
 - S allows switching the data file
 - U allows viewing the web usage report
 - T allows viewing the web tracker report
 - M allows updating the docInfo
 - V allows switching the region template
 - N allows remove content

Each selection corresponds to the checkbox for the action in the design view of the placeholder definition in Designer.

- **Returns:**

- The evaluated placeholder content.

These are the same parameters required by the `wcmPlaceholder` script extension.

This service can be used with webcache and Edge Side Includes (ESI) for partial page caching. When the page is rendered in an ESI environment, the ESI server can cache the request for placeholder contents.

11.7.79 WCM_EDIT_DATA_FILE

This service displays the Site Studio Contributor, allowing the user to edit the specified data file.

Parameters

- `dDocName`: the unique identifier of a data file.

When you check the file in, if it is no longer the head revision, then you cannot complete the check-in.

Note: This service should rarely, if ever, be called directly. Consider calling `WCM_BEGIN_EDIT_SESSION` instead.

11.7.80 WCM_BEGIN_EDIT_SESSION

This service can be used to check out a data file before editing it with the `WCM_EDIT_DATA_FILE` service. `WCM_BEGIN_EDIT_SESSION` will automatically redirect to `WCM_EDIT_DATA_FILE` after the checkout.

- **Parameters:**

- `dDocName`: the unique identifier of a Site Studio data file (required).

Site Studio Configuration Flags

This section covers the following topics:

- ["About Site Studio Flags"](#) on page 12-5
- ["DisableSiteStudioContribution"](#) on page 12-5
- ["ShowSiteStudioMissingDataFileErrors"](#) on page 12-5
- ["SiteStudioValidateElementDefinitions"](#) on page 12-6
- ["SiteStudioValidateRegionDefinitions"](#) on page 12-6
- ["SiteStudioValidatePlaceholderDefinitions"](#) on page 12-6
- ["SiteStudioValidateConversionsDefinitions"](#) on page 12-6
- ["SiteStudioValidateDataFiles"](#) on page 12-7
- ["SiteStudioValidateProjects"](#) on page 12-7
- ["SSAccessDeniedHeader"](#) on page 12-7
- ["SSAccessDeniedReplacementHeader"](#) on page 12-7
- ["SSAccessDeniedUserAgentExceptions"](#) on page 12-8
- ["SSAccommodateWelcomeFile"](#) on page 12-8
- ["SSAdditionalNavResultSetFields"](#) on page 12-8
- ["SSAddSecurityIDValues"](#) on page 12-8
- ["SSAfterProjectLoadedProperties"](#) on page 12-9
- ["SSAllowDynamicDefinitions"](#) on page 12-9
- ["SSAllowEmptyUrlPageName"](#) on page 12-9
- ["SSAllowNotModifiedHeader"](#) on page 12-9
- ["SSAltTagFieldName"](#) on page 12-10
- ["SSAlwaysRecordServerConfig"](#) on page 12-10
- ["SSAssumeXmlIsUtf8"](#) on page 12-10
- ["SSAutoCheckinBusyTimeout"](#) on page 12-10
- ["SSBackupCollectionName"](#) on page 12-11
- ["SSCacheControlOverride"](#) on page 12-11
- ["SSCanGenerateUniqueDataFiles"](#) on page 12-11
- ["SSChangeAccessDeniedHeaders"](#) on page 12-11

-
- ["SSCheckAssignedContentAccess"](#) on page 12-12
 - ["SSCheckBrowserForSiteRoot"](#) on page 12-12
 - ["SSCheckWebsiteObjectSecurity"](#) on page 12-12
 - ["SSClearDefinitionArchiveWebsites"](#) on page 12-12
 - ["SSCompressorArguments"](#) on page 12-13
 - ["SSCompressorCommand"](#) on page 12-13
 - ["SSCompressorDir"](#) on page 12-13
 - ["SSCompressorJar"](#) on page 12-13
 - ["SSCompressorMainClass"](#) on page 12-14
 - ["SSCompressorTimeout"](#) on page 12-14
 - ["SSCompressorTimerInterval"](#) on page 12-14
 - ["SSCompressorWaitForever"](#) on page 12-14
 - ["SSContributorSourceDir"](#) on page 12-15
 - ["SSCustomNodePropertyDefsPermissions"](#) on page 12-15
 - ["SSDefaultDocumentsFields"](#) on page 12-15
 - ["SSDefaultEditor"](#) on page 12-15
 - ["SSDefaultExternalDocNamePrefix"](#) on page 12-16
 - ["SSDefaultExternalDocNameSuffix"](#) on page 12-16
 - ["SSDefaultExternalUrlPrefix"](#) on page 12-16
 - ["SSDefaultExternalUrlSuffix"](#) on page 12-17
 - ["SSDefaultPlaceholderDefinition"](#) on page 12-17
 - ["SSDefaultRegionTemplate"](#) on page 12-17
 - ["SSDefaultUrlPageName"](#) on page 12-17
 - ["SSDetectIncludeFileEncoding"](#) on page 12-18
 - ["SSDICPlaceholderDefinition"](#) on page 12-18
 - ["SSDirectDeliveryExtensions"](#) on page 12-18
 - ["SSDirectDeliveryOverrideProperty"](#) on page 12-18
 - ["SSDirectDeliveryProperty"](#) on page 12-19
 - ["SSDirectDeliveryRequiredExtensions"](#) on page 12-19
 - ["SSDisableDeferredNodeExpansion"](#) on page 12-19
 - ["SSDisableIncludeXmlCache"](#) on page 12-19
 - ["SSDisableLinkResolutionSiteLocking"](#) on page 12-20
 - ["SSDisableProjectDeferredNodeExpansion"](#) on page 12-20
 - ["SSDomCacheDefaultFileSizeFactor"](#) on page 12-20
 - ["SSDomCacheFileSizeFactors"](#) on page 12-20
 - ["SSDomCacheLowerBound"](#) on page 12-21
 - ["SSDomCacheMultiplier"](#) on page 12-21

-
- ["SSDomCacheNodeMultiplier"](#) on page 12-21
 - ["SSDomCacheStringMultiplier"](#) on page 12-22
 - ["SSDomCacheStringOverhead"](#) on page 12-22
 - ["SSDomCacheUseDOM"](#) on page 12-22
 - ["SSDomCacheUseFileSize"](#) on page 12-22
 - ["SSEditorDebugLevel"](#) on page 12-22
 - ["SSEnableASPSupport"](#) on page 12-23
 - ["SSEnableDirectDelivery"](#) on page 12-23
 - ["SSEnableExtranetLookCompatibility"](#) on page 12-23
 - ["SSEnableFolioEditing"](#) on page 12-23
 - ["SSEnableFormEditing"](#) on page 12-24
 - ["SSEnableJavaScriptCompressor"](#) on page 12-24
 - ["SSEnableUpperCaseColumnsCheck"](#) on page 12-24
 - ["SSGenerateUniqueNodeIds"](#) on page 12-24
 - ["SSHidePrimaryFileInContributor"](#) on page 12-25
 - ["SSHttpAbsoluteHelpRoot"](#) on page 12-25
 - ["SSHttpLayerManager"](#) on page 12-25
 - ["SSIdocMarker"](#) on page 12-25
 - ["SSIgnoreMaxAgeNodeProperties"](#) on page 12-25
 - ["SSIgnoreNoProjectDefaultMetadataMessage"](#) on page 12-26
 - ["SSIgnoreReadyToReplicate"](#) on page 12-26
 - ["SSIimportOnlyLatestRevs"](#) on page 12-26
 - ["SSIincludeInactiveNodesInNavResultSet"](#) on page 12-26
 - ["SSIincludeInactiveNodesInNavXML"](#) on page 12-27
 - ["SSIincludeRegionTemplatesInDefinitionBundles"](#) on page 12-27
 - ["SSIincludeXmlTransformFormat"](#) on page 12-27
 - ["SSIincludeXmlTransformIndent"](#) on page 12-27
 - ["SSJavaExecutablePath"](#) on page 12-27
 - ["SSJSONContentType"](#) on page 12-28
 - ["SSLoadCustomElementsWithOnDemandEditors"](#) on page 12-28
 - ["SSLoadProjectsAtStartup"](#) on page 12-28
 - ["SSLoadUncompressedFckSource"](#) on page 12-29
 - ["SSManuallyValidateNodeIdUniqueness"](#) on page 12-29
 - ["SSMaxNodeIdLength"](#) on page 12-29
 - ["SSMaxSiteIdLength"](#) on page 12-29
 - ["SSMaxSitesMenuItems"](#) on page 12-30
 - ["SSMaxTemplateEvaluationStack"](#) on page 12-30

-
- ["SSMigrationCollectionName"](#) on page 12-30
 - ["SSOmitFragmentLibrariesInArchiverQueries"](#) on page 12-30
 - ["SSOnDemandEditorsThresholdCount"](#) on page 12-31
 - ["SSPrefillUrlDirNamesDuringUpgrade"](#) on page 12-31
 - ["SSProjectAutoCheckinInterval"](#) on page 12-31
 - ["SSProjectLoadFailureTracingSection"](#) on page 12-31
 - ["SSProjectReleaseSleepTime"](#) on page 12-32
 - ["SSProjectReleaseWaitTime"](#) on page 12-32
 - ["SSQuickDiffDefaultRegionTemplate"](#) on page 12-32
 - ["SSShowAssignmentTooltips"](#) on page 12-32
 - ["SSSQLUseContains"](#) on page 12-32
 - ["SSStoppedSiteResponsePageDocName"](#) on page 12-33
 - ["SSSuppressAddToWebsite"](#) on page 12-33
 - ["SSSuppressLargeCssOptimization"](#) on page 12-33
 - ["SSTempProjectLifetime"](#) on page 12-34
 - ["SSTitleTagFieldName"](#) on page 12-34
 - ["SSTrackContentAccess"](#) on page 12-34
 - ["SSTrackFragmentAccess"](#) on page 12-34
 - ["SSUrlFieldName"](#) on page 12-35
 - ["SSUrlFixupExceptions"](#) on page 12-35
 - ["SSUrlPageNames"](#) on page 12-35
 - ["SSUseAbsoluteRedirects"](#) on page 12-35
 - ["SSUseCallbackTrackingForASP"](#) on page 12-36
 - ["SSUseDefaultDocNamePrefix"](#) on page 12-36
 - ["SSUseDefaultServerRelativeSiteRoot"](#) on page 12-36
 - ["SSUseDefaultUrlPrefix"](#) on page 12-36
 - ["SSUseMissingLinkTargetFallback"](#) on page 12-36
 - ["SSUseOnDemandContributionModeMenus"](#) on page 12-37
 - ["SSUseUrlSegmentSessionInfo"](#) on page 12-37
 - ["SSValidateCustomElements"](#) on page 12-37
 - ["SSWebFilterIgnoreList"](#) on page 12-38
 - ["SSWebLayoutUrlUsesDocNames"](#) on page 12-38
 - ["SSWelcomeFile"](#) on page 12-38
 - ["SSWelcomeFileReplacement"](#) on page 12-38

12.1 About Site Studio Flags

Site Studio supports configuration flags that can be used to customize the operation of Site Studio. These flags allow you to finely control the operation of Site Studio both on the Oracle Content Server as well as for designers and contributors.

Whenever you change a flag, you should restart the Content Server.

Perform these steps to add configuration parameters for Oracle UCM:

1. Log in to Oracle Content Server as an administrator at the following URL:

```
http://Host_Name:Port/cs
```

The default protocol for Oracle Content Server is *http*, and *cs* is the default http relative web root for Oracle Content Server. The default port number is 16200.

2. Open the **Administration** page, and then choose **Admin Server**. The Content Admin Server page is displayed.
3. Click **General Configuration** on the left. The General Configuration page is displayed.
4. In the **Additional Configuration Variables** box, add or change the parameter or parameters as needed.
5. Click **Save** and restart the content server.

Flags that have no default value must be added to the configuration file, they are not included in the *config.cfg* file when shipped.

12.2 DisableSiteStudioContribution

This flag prevents pages from opening in contribution mode. All methods of switching to contribution mode are blocked. This can be useful when creating production or read-only Web sites.

Values

Boolean

Default

no

12.3 ShowSiteStudioMissingDataFileErrors

This flag is used to control if a message is shown in contribution regions with empty data file assignments.

Values

Boolean

Default

no

12.4 SiteStudioValidateElementDefinitions

This flag is used to control if element definition files are validated against the XSD schema.

Values

Boolean

Default

yes

12.5 SiteStudioValidateRegionDefinitions

This flag is used to control if region definition files are validated against the XSD schema.

Values

Boolean

Default

yes

12.6 SiteStudioValidatePlaceholderDefinitions

This flag is used to control if placeholder definition files are validated against the XSD schema.

Values

Boolean

Default

yes

12.7 SiteStudioValidateConversionsDefinitions

This flag is used to control if conversion definition files are validated against the XSD schema.

Values

Boolean

Default

yes

12.8 SiteStudioValidateDataFiles

This flag is used to control if data files are validated against the XSD schema.

Values

Boolean

Default

yes

12.9 SiteStudioValidateProjects

This flag is used to control if project files are validated against the XSD schema.

Values

Boolean

Default

yes

12.10 SSAccessDeniedHeader

This flag is used to define a header string to search for in http responses. If found, this will be replaced with the value of the flag *SSAccessDeniedReplacementHeader* (see "[SSAccessDeniedReplacementHeader](#)" on page 12-7).

Values

String

Default

401 Access denied

12.11 SSAccessDeniedReplacementHeader

This flag is used to define a header string to use as the replacement for the value of *SSAccessDeniedHeader* (see "[SSAccessDeniedHeader](#)" on page 12-7).

Values

String

Default

499 Oracle SSO

12.12 SSAccessDeniedUserAgentExceptions

This flag is used to set a comma-separated list of values that, if found in the *HTTP-USER-AGENT* header, will nullify the *SSChangeAccessDeniedHeaders* flag (see "[SSChangeAccessDeniedHeaders](#)" on page 12-11) for the particular request.

Values

CSV String

Default

No default value.

12.13 SSAccommodateWelcomeFile

This flag is used to control if the `<welcome-file>` mechanism of WLS-based content server is accommodated.

Values

Boolean

Default

yes

12.14 SSAdditionalNavResultSetFields

This flag is used to specify a comma-separated list of additional section properties to add to the standard set of properties in the *SiteStudioNavNodes ResultSet* used with the *ssLoadSiteNavResultSet* Idoc function (see "[ssLoadSiteNavResultSet](#)" on page 9-15 for more information).

Values

CSV String

Default

No default value.

12.15 SSAddSecurityIDValues

This flag is used to specify if a *did* value of 0 is placed in the *ResultSet* used to check user access to sections. This was needed for operability with some early versions of Records Management.

Values

Boolean

Default

no

12.16 SSAfterProjectLoadedProperties

This flag is used to specify a comma-separated list of site properties to send to the *SSAfterProjectLoaded* PluginFilter used with Site Studio for External Applications.

Values

CSV String

Default

siteLabel, siteType, isExternal

12.17 SSAllowDynamicDefinitions

This flag is used to enable or disable Idoc evaluation of definition files loaded from disk.

Values

Boolean

Default

yes

12.18 SSAllowEmptyUrlPageName

This flag is used to specify if a landing page is delivered for an incoming URL with an empty value for Url Page Name. In previous versions of Site Studio, the behavior was to do a redirect in these circumstances to generate the full landing page URL. Set the option false to restore that old behavior.

Values

Boolean

Default

yes

12.19 SSAllowNotModifiedHeader

This flag is used to override customized behavior for Site Studio Publisher Utility. The SSPETag flag is used to return a 304 to indicate that the result from a GET_FILE URL has not changed. Use the SSAllowNotModifiedHeader flag to disable this behavior on >= 7.2 servers.

Values

Boolean

Default

yes

12.20 SSAltTagName

This flag is used to specify the name of the metadata field to use for *alt tag* tags on images inserted via Contributor.

Values

String

Default

dDocTitle

12.21 SSAlwaysRecordServerConfig

This flag is used to specify whether or not to record the "server config" in a backup archive.

Values

Boolean

Default

no

12.22 SSAssumeXmlIsUtf8

This flag is used to determine if the content of files with the extension ".xml" are encoded in UTF-8 format. With the flag set to "yes", making this assumption speeds the processing of XML files by avoiding the inspection of the encoding declaration in the file itself.

Values

Boolean

Default

yes

12.23 SSAutoCheckinBusyTimeout

This flag is used to set the minimum time duration (in seconds) before the auto check-in mechanism attempts a check-in of a project file. This prevents two nodes from trying to check the project file in at the same time.

Values

Integer

Default

30

12.24 SSBackupCollectionName

This flag is used to override the name of the Site Studio backup collection used with Archiver.

Values

String

Default

No default value.

12.25 SSCacheControlOverride

This flag is used to provide the same cache-control header on every response. It replaces anything that would be provided by the `maxage` and `maxagesecondary` node properties. See also "[SSIgnoreMaxAgeNodeProperties](#)" on page 12-25.

Values

String

Default

No default value.

12.26 SSCanGenerateUniqueDataFiles

This flag is used as a global override for the Generate Unique Data Files option in the Region Menu in Design mode. When set to *yes*, the option is available for Design mode.

Values

Boolean

Default

yes

12.27 SSChangeAccessDeniedHeaders

This flag is used to control whether 401 responses will be changed to 499 responses for use with Oracle SSO.

Values

Boolean

Default

no

12.28 SSSheckAssignedContentAccess

This flag is used to check the *dDocNames* assigned to the primary or secondary url for security access. This check happens during the actions part of *SS_GET_PAGE*, so the error page can be shown.

Values

Boolean

Default

no

12.29 SSSheckBrowserForSiteRoot

This flag is used to indicate whether or not to produce server relative site root values with respect to the browser URL. If set to *no*, the server-relative URL prefix will be generated from the default site address.

Values

Boolean

Default

yes

12.30 SSSheckWebsiteObjectSecurity

This flag is used to activate security checking for website objects. Normally, since **use** of a website object by the component/server while delivering a page doesn't really constitute end-user "access" to the resource, there are no security checks on these objects. Setting this flag to *true* will activate security checking for each website object.

Values

Boolean

Default

no

12.31 SSSclearDefinitionArchiveWebsites

This flag is used to forcibly remove the existing *xWebsites* values when importing a definition archive.

Values

Boolean

Default

no

12.32 SSCompressorArguments

This flag is used for additional arguments passed to the JavaScript compressor's process.

Values

String

Default

No default value.

12.33 SSCompressorCommand

This flag is used as a command-line replacement to launch the JavaScript compressor's process.

Values

String

Default

No default value.

For more information on the string value that would be used, see the README file located in the `wcm\tools\yui-compressor\doc\` folder on the Oracle Content Server.

12.34 SSCompressorDir

This flag is used as an override for the JavaScript compressor's implementation location.

Values

String

Default

`<weblayout>/resources/wcm/tools/optimize`

12.35 SSCompressorJar

This flag is used as an override for the JavaScript compressor's Rhino jar file location.

Values

String

Default

`<weblayout>/resources/wcm/tools/rhino/rhino1_7R2/js.jar`

12.36 SSCompressorMainClass

This flag is used as an override for the JavaScript compressor's main class.

Values

String

Default

`org.mozilla.javascript.tools.shell.Main`

12.37 SSCompressorTimeout

This flag is used to override the JavaScript compressor's minimum amount of time allowed (in seconds) for the process to complete.

Values

Integer

Default

900

12.38 SSCompressorTimerInterval

This flag is used to override the length of the JavaScript compressor's process sleep interval (in seconds).

Values

Integer

Default

10

12.39 SSCompressorWaitForever

This flag is used to override the JavaScript compressor's file lock wait duration.

Values

Boolean

Default

no

12.40 SSContributorSourceDir

This flag is used to determine the directory within the `<weblayout>/resources` directory where the Contributor JavaScript code is referenced.

Values

String

Default

wcm

12.41 SSCustomNodePropertyDefsPermissions

This flag is used to determine the default permission a user needs to access custom node properties.

The values are one of the following four:

- 1 = Read Permission
- 2 = Write Permission
- 4 = Delete Permission
- 8 = Admin Permission

Values

Integer

Default

2

12.42 SSDefaultDocumentsFields

This flag is used to override the default documents fields. Enter the default documents fields to use.

Values

CSV String

Default

No default. Values entered are field names.

12.43 SSDefaultEditor

This flag is used to specify the base editor for the Contributor application.

Values

String

Default

fck

12.44 SSDefaultExternalDocNamePrefix

This flag is used to specify a string (possibly a partial URL) to prefix evaluated `wcmUrl` links of type `resource` that have a `dDocName` parameter. This is used when delivering content through the `WCM_PLACEHOLDER` service where there might be a need to alter the URLs produced.

Values

String

Default

The evaluation of `<$HttpAbsoluteWebRoot$>`

12.45 SSDefaultExternalDocNameSuffix

This flag is used to specify a string (possibly a URL Query segment) to affix to evaluated `wcmUrl` links of type `resource` that have a `dDocName` parameter. This is used when delivering content through the `WCM_PLACEHOLDER` service where there might be a need to alter the URLs produced.

Values

String

Default

No default

12.46 SSDefaultExternalServerRelativeSiteRoot

This flag is used to specify a string to use when evaluating `<$ssServerRelativeSiteRoot$>`. This is used when delivering content through the `WCM_PLACEHOLDER` service where there might be a need to alter the URLs produced.

Values

String

Default

No default

12.47 SSDefaultExternalUrlPrefix

This flag is used to specify a string (possibly a partial URL) to prefix evaluated `wcmUrl` links of type `resource` that have a partial `weblayout path` parameter. This is used when delivering content through the `WCM_PLACEHOLDER` service where there might be a need to alter the URLs produced.

Values

String

Default

The evaluation of `<$HttpAbsoluteWebRoot$>`

12.48 SSDefaultExternalUrlSuffix

This flag is used to SPECIFY a string (possibly a URL Query segment) to affix to evaluated `wcmUrl` links of type `resource` that have a partial weblayout path parameter. This is used when delivering content through the WCM_PLACEHOLDER service where there might be a need to alter the URLs produced.

Values

String

Default

No default

12.49 SSDefaultPlaceholderDefinition

This flag is used to override the placeholder definition.

Values

String

Default

SS_DEFAULT_PLACEHOLDER_DEFN

12.50 SSDefaultRegionTemplate

This flag is used to override the default region template.

Values

String

Default

SS_DEFAULT_REGION_TEMPLATE

12.51 SSDefaultUrlPageName

This flag is used to change the value of the default url page name. This allows files other than those named `index.htm` to be the default page for a section of a Web site.

The flag *SSUrlPageNames* (see "[SSUrlPageNames](#)" on page 12-35) allows additional url page names to be used to deliver the primary page.

Values

String

Default

No default value.

12.52 SSDetectIncludeFileEncoding

This flag is used to specify if the encoding of Web site objects have their encoding determined or not.

Values

Boolean

Default

no

12.53 SSDICPlaceholderDefinition

This flag is used to specify a default Placeholder Definition name to be used in *Doc Info Contribution*.

Values

String

Default

SS_DEFAULT_PLACEHOLDER_DEFN

12.54 SSDirectDeliveryExtensions

This flag is used to specify which file extensions are delivered directly. If the flag is not listed in *config.cfg*, then PDF files will still pass via direct delivery, and other native documents will follow conversion rules.

When the flag is present, all file types listed here (by filename extension) will be delivered directly. File types not specifically listed will be displayed via conversion. This includes pdf files, if the flag is present but pdf files are not listed.

Values

CSV String

Default

pdf

12.55 SSDirectDeliveryOverrideProperty

This flag is used to specify which custom section property is used to override the global values of the section. The property named in the flag should use a boolean value indicating if the global values should be overridden or not.

Values

String

Default

OverrideDirectDeliveryExtensions

12.56 SSDirectDeliveryProperty

This flag is used to specify the name of a custom section property. This property identifies the file extensions to deliver directly from this particular section.

The file extensions listed here will override the global list in *SSDirectDeliveryExtensions* (see "[SSDirectDeliveryExtensions](#)" on page 12-18).

Values

String

Default

DirectDeliveryExtensions

12.57 SSDirectDeliveryRequiredExtensions

This flag is used to specify file types that are always to be delivered by direct delivery. The file extensions listed here are not overridden by those listed in *SSDirectDeliveryProperty* (see "[SSDirectDeliveryProperty](#)" on page 12-19).

Values

CSV String

Default

pdf

12.58 SSDisableDeferredNodeExpansion

This flag is used to enable or disable deferred node expansion. The Xerces parser's deferred node expansion feature is known to be a very inefficient feature for small DOMs, so setting this flag to no might impede performance.

Values

Boolean

Default

yes

12.59 SSDisableIncludeXmlCache

This flag is used to control if extracted content from data files is cached. This cache makes retrieval of content from data files much faster.

Values

Boolean

Default

no

12.60 SSDisableLinkResolutionSiteLocking

This configuration flag is used to reduce site lock contention during page assembly. When the flag is set to `yes`, it prevents the Idoc script extensions `ssLink` and `ssNodeLink`, as well as the `wcmUrl` equivalents `wcmUrl('link'...` and `wcmUrl('nodelink'...`, from synchronizing their accesses of internal web site information structures. This can improve performance by reducing lock contention, which then allows multiple requests to process simultaneously. However, this comes at the risk of producing inaccurate links in a changing web site.

Setting the flag to `yes` is most useful when the Oracle Content Server is run under the following conditions:

- The Site Studio web sites change infrequently; it is largely a read-only environment.
- The web pages contain numerous `ssLink` or `ssNodeLink` calls to produce links.
- The server is typically sufficiently loaded to cause lock contention inside Site Studio.

Values

Boolean

Default

`no`

12.61 SSDisableProjectDeferredNodeExpansion

This flag is used to control whether DOM nodes will be expanded in memory during the loading of XML files, or if the node content will be loaded on-demand.

Values

Boolean

Default

`yes`

12.62 SSDomCacheDefaultFileSizeFactor

This flag specifies the default file size value to multiply the file size by to obtain a cache size.

Values

Numeric

Default

`2.0`

12.63 SSDomCacheFileSizeFactors

This flag defines a comma-separated list of file sizes and multipliers that control the computed cache size.

For example, the following string

1000, 6.0, 10000, 2.7, 50000, 2.1, 100000, 1.9, 300000, 1.6

multiplies files sized [0..999] by 6.0,

multiplies files sized [1000..9999] by 2.7,

multiplies files sized [10000..49999] by 2.1,

multiplies files sized [50000..99999] by 1.9,

multiplies files sized [100000..299999] by 1.6,

Files outside the range above are multiplied by the value of *SSDomCacheDefaultFileSizeFactor* (see "[SSDomCacheDefaultFileSizeFactor](#)" on page 12-20).

Values

CSV string

Default

No default value.

12.64 SSDomCacheLowerBound

This flag is used to define a lower bound on the reported cache size.

Values

Numeric

Default

6000

12.65 SSDomCacheMultiplier

This flag is used to set a multiplier on the computed cache size to arrive at a final value reported to the cache. The cache will then multiply the reported value by 10 in its computations.

Values

Numeric

Default

0.1

12.66 SSDomCacheNodeMultiplier

This flag specifies the number of bytes to count per DOM node.

Values

Numeric

Default

12

12.67 SSDomCacheStringMultiplier

This flag specifies the number of bytes to multiply string lengths in the DOM by to produce the string size.

Values

Numeric

Default

2

12.68 SSDomCacheStringOverhead

This flag specifies the number of bytes to add per string in the DOM.

Values

Numeric

Default

24

12.69 SSDomCacheUseDOM

This flag is used to compute a cache size based on an enumeration of the XML DOM.

Values

Boolean

Default

no

12.70 SSDomCacheUseFileSize

This flag is used to compute a cache size based on the size of the file.

Values

Boolean

Default

yes

12.71 SSEditorDebugLevel

This flag is used to override the Ephox editor's debug level.

The available values are *http*, *debug*, *info*, *warn*, *error*, and *fatal*.

Values

String

Default

No default value.

12.72 SSEnableASPSupport

This flag is used to enable ASP support in Site Studio 11gR1 and above. In the 11gR1 component and later, ASP support is disabled by default.

Values

Boolean

Default

no

12.73 SSEnableDirectDelivery

This flag is used to control direct delivery. Direct delivery allows you to link to a native document content file for download, rather than having it display in the page using conversion rules.

For more information, see the *User's Guide for Site Studio Designer*.

Values

Boolean

Default

no

12.74 SSEnableExtranetLookCompatibility

This flag is used to allow interoperability with the ExtranetLook component, as well as to preserve friendly URLs when certain Single Sign On (SSO) systems are used.

Values

Boolean

Default

no

12.75 SSEnableFolioEditing

This flag allows Folios assigned to a Site Studio region or placeholder to be edited.

Values

Boolean

Default

no

12.76 SSEnableFormEditing

This flag is used to override including the Form Editor option in the Region Menu. Setting the flag to `yes` includes the Form Editor in the menu.

Values

Boolean

Default

`no`

12.77 SSEnableJavaScriptCompressor

This flag enables the JavaScript compression from the Site Studio Administrator pages in the Oracle Content Server.

Values

Boolean

Default

`yes`

12.78 SSEnableUpperCaseColumnsCheck

This flag is used to ensure that *xRegionDefinition* has the same case-preserving aspects as *dDocName*.

Values

Boolean

Default

`yes`

12.79 SSGenerateUniqueNodeIds

This flag is used to ensure that *nodeIds* are unique server-wide. If set to boolean `no`, the *nodeId* only be unique within each project. This latter action mimics older Site Studio behavior.

Values

Boolean

Default

`yes`

12.80 SSHidePrimaryFileInContributor

This flag is used to hide the *filename* field when creating a new web asset. This helps to avoid modifying the the name to something inappropriate.

Values

Boolean

Default

no

12.81 SSHttpAbsoluteHelpRoot

This flag is used to direct Site Studio Contributor to the help file location.

Values

String

Default

<\${HttpWebRoot\$}>help/

12.82 SSHttpLayerManager

This flag is used as an alternate *HttpLayerManager*. Using this can be useful with some implementations with Ephox.

Valid values for Ephox include `default` and `sun`.

Values

String

Default

No default value.

12.83 SSIdocMarker

This flag is used to set the value that determines if a Web site object contains Idoc Script code or not, and thus if it is a candidate for Idoc parsing and evaluation.

Values

String

Default

!--\$

12.84 SSIgnoreMaxAgeNodeProperties

This flag is used to override the *maxage* and *maxagesecondary* node properties.

While you might want these properties to generate cache-control headers on your live system, it could be desirable to disable that behavior on the development environment. See also: "[SSCacheControlOverride](#)" on page 12-11.

Values

Boolean

Default

no

12.85 SSIgnoreNoProjectDefaultMetadataMessage

This flag is used to override any errors caused with default project metadata. Normal behavior is to throw an exception if the default project metadata has not been set. Override that by setting this option to boolean *yes*.

Values

Boolean

Default

no

12.86 SSIgnoreReadyToReplicate

This flag is used to override section level settings for Ready to Replicate. Setting the flag to *yes* will cause all sections to be replicated regardless of section level setting.

Values

Boolean

Default

no

12.87 SSImportOnlyLatestRevs

This flag is used to import only the latest revision when importing an archive. The default behavior when importing an archive is to import everything.

Values

Boolean

Default

no

12.88 SSIncludeInactiveNodesInNavResultSet

This flag is used to include inactive nodes in the *ssNavNodes* result set. By default the *ssNavNodes* result set does not include inactive nodes.

Values

Boolean

Default

no

12.89 SSIncludeInactiveNodesInNavXML

This flag is used to include inactive nodes in the navigation XML file. By default the navigation XML does not include inactive nodes.

Values

Boolean

Default

no

12.90 SSIncludeRegionTemplatesInDefinitionBundles

This flag is used to determine if region templates are included in definition bundles. The default behavior is to include Region Templates in a definition bundle. Omit them by setting this option false.

Values

Boolean

Default

yes

12.91 SSIncludeXmlTransformFormat

This flag is used to set the file type that data files are transformed into when transformed. The transformation can be specified as xml, xhtml or html.

Results may change considerably by modifying this flag.

Values

String

Default

HTML

12.92 SSIncludeXmlTransformIndent

This flag is used to format the XML file to include indentations. This allows the XML code to be printed in an indented format, rather than as a single line.

Values

Boolean

Default

no

12.93 SSJavaExecutablePath

This flag is used to override what Java executable to use for the JavaScript compressor implementation. This flag should be used when the default (*java.home* environment variable) should not be used.

The default variable for this flag should be changed if you intend to use the flag.

Values

String

Default

`java.home env variable`

12.94 SSJSONContentType

This flag is used to override the JSON content type.

Values

String

Default

`application/jsonrequest`

12.95 SSLoadCustomElementsWithOnDemandEditors

This flag allows custom elements to be fully loaded when all standard elements are loaded on demand. Custom elements provide their own UI so if they are loaded on demand they will have no preview. In general custom elements load more quickly than standard elements so allowing these to load fully should not have a large impact on the load time performance of the Contributor form.

Values

Boolean

Default

`yes`

12.96 SSLoadProjectsAtStartup

This flag is used to override the loading of projects at startup. If there are a substantial number of projects in the system, this could prevent the Oracle Content Server service from starting as a service under Windows.

Values

Boolean

Default

`yes`

12.97 SSLoadUncompressedFckSource

This flag is used to load uncompressed FCK Editor source code. This is a debugging aid to FCK Editor.

Values

Boolean

Default

no

12.98 SSManuallyValidateNodeIdUniqueness

This flag is used to override the use of the XML parser to validate unique *nodeIds*. Certain implementations will validate large projects slowly when using the XML parser, so the default value is to not use it.

Values

Boolean

Default

yes

12.99 SSMaxNodeIdLength

This flag is used to override the maximum node ID value length.

Values

Integer

Default

30

12.100 SSMaxSiteIdLength

This flag is used to override the maximum site ID value length.

Values

Integer

Default

30

12.101 SSMaxSitesMenuItems

This flag is used to set the maximum number of Web sites that can be displayed in the websites menu.

Values

Integer

Default

No default value.

12.102 SSMaxTemplateEvaluationStack

This flag is used to override the maximum number of Placeholder nestings.

Values

Integer

Default

200

12.103 SSMigrationCollectionName

This flag is used to set the name of the collection that will be used for backup archives.

Values

String

Default

The value of the content server "Instance Name".

12.104 SSOmitFragmentLibrariesInArchiverQueries

This flag is used to omit the fragment libraries in a backup archive. The default setting is to include them.

Values

Boolean

Default

no

12.105 SSOnDemandEditorsThresholdCount

This flag is used to set the number of elements that must be present in a contribution to cause the elements to load on demand in Contributor.

Values

Integer

Default

6

12.106 SSPrefillUrlDirNamesDuringUpgrade

This flag is used to generate and populate values for the *Url Dir Name* section property with Site Studio Web sites before version 7.5.

Values

Boolean

Default

yes

12.107 SSProjectAutoCheckinInterval

This flag is used to set the time interval (in seconds) the project file is automatically checked in.

Values

Integer

Default

600

12.108 SSProjectLoadFailureTracingSection

This flag is used to specify a *NAME* of a tracing section to be used to dump diagnostics when a project load error occurs. By default these exceptions will always be dumped. Use this flag to selectively dump them when you enable the specified trace section.

Values

String

Default

No default value.

12.109 SSProjectReleaseSleepTime

This flag is used to set the time in seconds to sleep while waiting for project release. Also see "[SSProjectReleaseWaitTime](#)" on page 12-32

Values

Integer

Default

3

12.110 SSProjectReleaseWaitTime

This flag is used to set the time in seconds to wait for a newly committed project file to become "released" during an archive operation. Also see "[SSProjectReleaseSleepTime](#)" on page 12-32.

Values

Integer

Default

30

12.111 SSQuickDiffDefaultRegionTemplate

This flag is used to set the template used as the Quick Diff's region template.

Values

String

Default

SS_DEFAULT_REGION_TEMPLATE

12.112 SSShowAssignmentTooltips

This flag is used to enable or disable the Assignment tooltip while in Contributor mode. When set to *yes*, the tooltip will display when hovering over the region marker.

Values

Boolean

Default

no

12.113 SSSQLUseContains

This flag is used to control the query in SQL searches. When the SQL contains operator is not available for the *xWebsites* or *xDontShowInListsForWebsites* columns.

Setting this flag to *no* uses the *like* query; setting it to *yes* uses the *contains* query.

Additionally, there can be problems for some queries used by Site Studio if Site Id values contain the "_" character. These problems are caused by the fact that the "_" is used as a word break character at indexing time and as a wildcard character at query time.

To avoid these issues Site Studio can be told to not use a `contains` query but to instead use a four-part `like` query.

Values

Boolean

Default

no

12.114 SSStoppedSiteResponsePageDocName

This flag is used to set the *dDocName* of a content item defining a page to be returned if you try to access a website that has been stopped.

Values

String

Default

No default value.

12.115 SSSuppressAddToWebsite

This flag is used to prevent images that are linked to from being marked as part of the current Web site. When linking to an image, Site Studio normally marks that image as part of the current Web site, which requires write permission on the image. Use this flag to not mark the image.

Note that this image will not be picked up by site archives if this is used.

Values

Boolean

Default

no

12.116 SSSuppressLargeCssOptimization

This flag is used to enable or disable the CSS optimization calculation. The optimization calculation happens in Contributor mode, rather than when the Contributor form is being rendered so that the Contributor form loads faster.

When this flag is set to `yes`, the calculation will be used.

Values

Boolean

Default

no

12.117 SSTempProjectLifetime

This flag is used to override for the time (in seconds) to leave temp project files after they have been checked in to the content server.

You must use extreme caution if you change this.

Values

Integer

Default

120

12.118 SStitleTagFieldName

This flag is used to set which metadata field is used for title tags.

Values

String

Default

dDocTitle

12.119 SStrackContentAccess

This flag is used to specify the name of the metadata field to use for title tags on images inserted via Contributor.

Values

Boolean

Default

yes

12.120 SStrackFragmentAccess

This flag is used as an override to enable tracking of fragments with Tracker.

Values

Boolean

Default

no

12.121 SSUrlFieldName

This flag is used as an override to specify a metadata field name to be used instead of *dDocName* when producing the page identifier in URLs. Specify a name without the leading *x*, for instance `Foo`, not `xFoo`.

Values

String

Default

No default value.

12.122 SSUrlFixupExceptions

This flag defines a list of regular expressions applied during link fixup to determine if this link should not be fixed up.

Values

String, values separated by 'pipes': |

Default

No default value.

12.123 SSUrlPageNames

This flag is used to nominate other allowable URL page names that will deliver the primary page. This is helpful if there are existing data files with path-based links using the previous default URL page name.

The default URL page name is whatever is specified in each section property, otherwise, it is `index.htm`. The related flag *SSDefaultUrlPageName* (see "[SSDefaultUrlPageName](#)" on page 12-17) allows you to specify a default file that is not `index.htm`.

Values

CSV string

Default

No default value.

12.124 SSUseAbsoluteRedirects

This flag is used to override to restore previous Site Studio versions behavior of redirects. The current default and recommended value is to redirect to a relative URL. When this flag is set to Boolean `yes`, all redirects will use absolute URLs.

Values

Boolean

Default

no

12.125 SSUseCallbackTrackingForASP

This flag is used to determine if ASP uses the callback type of reporting content access for tracker. The default is to enable content access tracking. To enable the callback type that is potentially more accurate, but much slower, then enable this flag.

Values

Boolean

Default

no

12.126 SSUseDefaultDocNamePrefix

This flag is used to determine default value for the `SSDefaultExternalDocNamePrefix` configuration entry if it is not specified.

Values

Boolean

Default

yes

12.127 SSUseDefaultServerRelativeSiteRoot

This flag is used to determine if the default value for the `SSDefaultServerRelativeSiteRoot` configuration entry if it is not specified.

Values

Boolean

Default

yes

12.128 SSUseDefaultUrlPrefix

This flag is used to determine if the default value for the `SSDefaultUrlPrefix` configuration entry if it is not specified.

Values

Boolean

Default

yes

12.129 SSUseMissingLinkTargetFallback

This flag is used to generate a tokenized link in the cases where a `computeUrl` function does not have a target. If the target `dDocName` or the target `nodeId` no longer exist, then a replacement URL will not be generated. The default action of the flag computes a tokenized link to avoid script extension errors.

Values

Boolean

Default

yes

12.130 SSUseOnDemandContributionModeMenus

This flag controls the creation of the menus in Contributor. When a page has multiple placeholders, a lot of DOM manipulation is required to construct the popup menus for each placeholder marker. When using Internet Explorer this process can take a long time and cause the CPU to spike to 100%. When this flag is set to yes, the creation of the menus is delayed until the user actually clicks on the icon.

Values

Boolean

Default

yes

12.131 SSUseUrlSegmentSessionInfo

This flag allows the URL to determine which mode you are in by including an extra path segment. (For example: *contributor/design/preview<previewId>*)

Values

Boolean

Default

no

12.132 SSValidateCustomElements

This flag is used to determine whether or not a custom element validates the compatibility of a custom element form. This feature was intended for notifying users to upgrade their legacy custom element forms. This flag should be set to false for performance reasons.

Values

Boolean

Default

no

12.133 SSWebFilterIgnoreList

This flag is used to specify which folders to have the web server filter plugin ignore. This allows domain based sites to address resources external to the Oracle Content Server.

Values

String

Default

No default value.

12.134 SSWeblayoutUrlUsesDocNames

This flag is used to allow *dDocNames* to be used in place of weblayout paths with *ssWeblayoutUrl* and corresponding *wcmUrl('resource' ...)* links.

Values

Boolean

Default

no

12.135 SSWelcomeFile

This flag is used to specify the welcome file, a URL suffix automatically generated by WLS. This will typically be `portal.htm`, which will not match any of Site Studio's URLs.

Values

String

Default

`/portal.htm`

12.136 SSWelcomeFileReplacement

This flag is used to specify the replacement for the welcome file to use when matching the incoming URL against the project hierarchy.

Values

String

Default

`/`

Site Studio Performance Tuning

This section covers the following topics:

- ["About Site Studio Performance"](#) on page 13-1
- ["On-Demand Web Site Management"](#) on page 13-1
- ["On-Demand Contributor Editors"](#) on page 13-1
- ["Optimizing Contributor Code"](#) on page 13-2
- ["Memory Usage"](#) on page 13-5

13.1 About Site Studio Performance

Site Studio Web sites are dynamic Web sites that allow for the quick delivery of different forms of content within each different page. But there are some portions of Site Studio that can be customized to allow for even better performance.

13.2 On-Demand Web Site Management

Some of the Web sites you create and maintain may not be used often. In these cases, it would be easier to keep the server from loading these sites into memory until they are actually requested, improving the server performance through memory management.

The sites you want to load only when requested can be marked as such in the Administrator, using on-demand management you can select which sites load when the server boots, and those that load when requested.

When a site has been requested, it then stays in memory as the other sites.

For more information on On-Demand Web sites, see the *Administrator and Manager's Guide for Site Studio*.

13.3 On-Demand Contributor Editors

Contributor can occasionally take some time to load if there are a large number of elements in a contribution region. This is to be expected. There are some configuration flags in Site Studio that can be set to allow for faster loading of Contributor by using on-demand editors.

This section covers the following topics:

- ["About Configuration Flags"](#) on page 13-2
- ["Configuration Flags used for On-Demand Editors"](#) on page 13-2

13.3.1 About Configuration Flags

Site Studio uses a large number of configuration flags for different actions. Since there are so many flags, only those used in the most common instances will be described.

13.3.2 Configuration Flags used for On-Demand Editors

Contributor loads an editor for each element in a contribution region, whether or not it displays on the page. When there are a large number of elements, this can cause much longer loading and waiting times for the contributors.

There are flags that can be modified to allow for the use of on-demand editors in Contributor. The on-demand editor shows a preview of the data contained in the element, but the editor will not load until requested.

13.3.2.1 SSONDemandEditorsThresholdCount

This configuration flag is used to enable the on demand editors in Contributor only when there are a certain number of elements to load. When you set this flag to a certain number n , the elements will become on demand elements when there are $n+1$ elements in the contribution region.

Values

Integer

Default

5

13.4 Optimizing Contributor Code

There are substantial portions of the code in Contributor that are based on JavaScript and CSS. This code can be optimized.

Optimized Contributor source code is JavaScript and CSS code that has been compressed and consolidated. Optimized source code improves the Contributor application's load time and execution speed in the browser. By default, Site Studio is initially configured to use un-optimized source code for debugging and configuration purposes.

This section includes the following topics:

- ["Optimization Requirements"](#) on page 13-3
- ["The Build Process"](#) on page 13-3
- ["Building the Optimized Code"](#) on page 13-4
- ["Debugging the Build Script"](#) on page 13-4
- ["Configuring Site Studio to Use Optimized Code"](#) on page 13-5
- ["Customizations and the Build Process"](#) on page 13-5

13.4.1 Optimization Requirements

The build process requires a JDK for Java that implements the 1.5 or greater specifications.

It is important to note that the build system will only work correctly by default if the *tools* directory resides just within the root directory of the Contributor application. In most cases, the tools directory will reside in the following location:

```
<CONTENT_SERVER_INSTALL_DIR>\weblayout\resources\wcm\tools
```

Note: If the computer is configured to use a default Java Runtime Environment with a version less than 1.5, the invocation of the compression module in the *WCM.Compress* function in the *build.js* script should be changed to reference a compliant runtime environment. Specifically, the first parameter to the *runCommand* function should be changed from *java* to a full path reference to the java executable version 1.5 or greater.

13.4.2 The Build Process

All the logic to build optimized Contributor source code is located in *build.js*, written in JavaScript. The build process uses Rhino, a JavaScript interpreter, to execute the build script to create optimized Contributor source code. The *build.js* script file is located in following directory:

```
\tools\optimize\build.js
```

The build process first makes a duplicate copy of the un-optimized code. By default, the destination directory will be called 'wcm_min' and will reside just outside the root directory of the Contributor application in the following location:

```
<CONTENT_SERVER_INSTALL_DIR>\weblayout\resources\wcm_min
```

After that, the build process manipulates the newly copied code into optimized source code by compressing all JavaScript and CSS files, then scanning each HTM file, and finally concatenating all JavaScript and CSS files referenced in the HTM file. The build process reads the HTM files at run-time and enumerates all the JavaScript and CSS files in the HTM file that reside within a well-known comment syntax.

For example, all the JavaScript references within the following comment syntax

```
<!-- BEGIN:SCRIPT -->
<script type="text/javascript" src=" ../wcm.js"></script>
<script type="text/javascript" src=" ../base/wcm.dhtml.js"></script>
<script type="text/javascript" src=" ../base/wcm.get.js"></script>
<script type="text/javascript" src=" ../base/wcm.http.js"></script>
<script type="text/javascript" src=" ../base/wcm.popup.js"></script>
<script type="text/javascript" src=" ../base/wcm.compare.js"></script>
<script type="text/javascript" src=" ../base/wcm.diff.js"></script>
<script type="text/javascript" src=" ../sitestudio/wcm.idc.js"></script>
<script type="text/javascript"
src=" ../sitestudio/wcm.contentserver.popup.js"></script>
<script type="text/javascript" src=" ../sitestudio/wcm.helpfiles.js"></script>
<script type="text/javascript" src=" ../sitestudio/wcm.help.js"></script>
<script type="text/javascript" src=" ../form/wcm.form.js"></script>
<script type="text/javascript" src="wcm.sitestudio.form.js"></script>
<script type="text/javascript"
src=" ../custom/sitestudio/wcm.sitestudio.form.js"></script>
```

```
<!-- END:SCRIPT -->
```

are compressed, concatenated, and referenced as

```
<!-- BEGIN:SCRIPT -->  
<script type="text/javascript" src="../../wcm.sitestudio.form.composite.js"></script>  
<!-- END:SCRIPT -->
```

Similarly, the following CSS references within the following comment syntax:

```
<!-- BEGIN:CSS -->  
<link href="../../base/wcm.base.css" rel="stylesheet" type="text/css" />  
<link href="wcm.sitestudio.form.css" rel="stylesheet" type="text/css" />  
<link href="../../custom/sitestudio/wcm.sitestudio.form.css" rel="stylesheet"  
type="text/css" />  
<!-- END:CSS -->
```

are compressed, concatenated, and referenced as

```
<!-- BEGIN:CSS -->  
<link href="wcm.sitestudio.form.composite.css" rel="stylesheet" type="text/css" />  
<!-- END:CSS -->
```

13.4.3 Building the Optimized Code

To run the build process on Windows, double-click the shell-build shortcut in the following location:

```
\tools\optimize\shell-build
```

Use the following command line to run the build process on other Site Studio supported platforms, including Windows:

```
java.exe -cp ..\rhino\rhino1_7R2\js.jar org.mozilla.javascript.tools.shell.Main  
build.js
```

When running the build process from the command line, it is important to ensure that the execution working directory is that of the build script's directory. For instance, if executing the build from a shell, be sure to change directories to that of the build script (*\tools\optimize*) before executing the command line.

The process can also be run from the **Manage Fragment Libraries** page in Site Studio Administration. Click **Compress Contributor JavaScript** to start the process.

13.4.4 Debugging the Build Script

To run the visual debugger on Windows, double-click the shell-debug shortcut in the following location:

```
\tools\optimize\shell-debug
```

Use the following command line to run the visual debugger on other supported platforms, including Windows:

```
java.exe -cp ..\rhino\rhino1_7R2\js.jar org.mozilla.javascript.tools.debugger.Main  
build.js
```

When running the visual debugger from the command line, it is important to ensure that the debugger's execution working directory is that of the build script's directory. For example, if executing the debugger from a shell, be sure to change directories to that of the build script (*\tools\optimize*) before executing the command line.

13.4.5 Configuring Site Studio to Use Optimized Code

Once the optimized Contributor source code is built, Site Studio can be configured to use the optimized source code by changing the *SSContributorSourceDir* value. This value can be set in the following configuration file:

```
<CONTENT_SERVER_INSTALL_DIR>\custom\SiteStudio\SiteStudio.cfg
```

Within *SiteStudio.cfg*, set *SSContributorSourceDir* to the value *wcm_min*:

```
SSContributorSourceDir=wcm_min
```

13.4.6 Customizations and the Build Process

Customizations made to the Contributor source code will be picked up by the build process automatically as long as the JavaScript and CSS file references reside within the special comment syntax. In addition, if newly added HTML files contain the special comment syntax around the JavaScript and CSS file references, then the build process will perform optimizations on those files.

The build script can also be modified to suit a user's customization needs. For instance, the output directory name or location can be changed. In addition, there are regular expression filters to determine which files get copied, compressed, or concatenated. These filters can be updated to suit any customizations as needed.

13.5 Memory Usage

The use of memory in cached data can affect how responsive Site Studio is. One method of controlling this is done in the Site Studio program itself, where the memory requirements for the Site Studio structures were reduced by incorporating different changes.

Some of these changes include: disabling the Xerces deferred DOM loading and removing extraneous whitespace text nodes from the Project File DOM, among other changes.

The memory usage can also be controlled in the use of flags. This section covers the following topics:

13.5.1 Flags for Memory Size in XML DOMs

The Xerces parser's deferred node expansion feature is known to be a very inefficient feature for small DOMs. However, it can be controlled via a flag.

- **SSDisableDeferredNodeExpansion** - Enables or disables deferred node expansion. The default is true.

13.5.2 Flags for Size of Items in the DOC_INFO Cache

The algorithms that compute the size of the items in the *DOC_INFO* cache, can be fine-tuned in two ways. The first way is to use a multiplier on the file size. The second is to enumerate the DOM. The following configuration flags are available:

- **SSComputeDocInfoCacheSize** - A boolean value indicating if the size of the *ResultSet* should be computed or not. When the *ResultSet* size is not computed, the *SSDocInfoCacheLowerBound* value is used. The default is false.
- **SSDocInfoCacheLowerBound** - This places a lower bound on the size of the *ResultSet* reported to the cache. The default is 2000.

- **SSDocInfoCacheMultiplier** - A multiplier on the computed cache size to arrive at a final value reported to the cache. The reported value will be multiplied by 10 by the cache itself in its computations. The default is 0.1.
- **SSDocInfoCacheStringMultiplier** - A multiplier for Strings lengths to produce the final string size in memory. The default is 2.
- **SSDocInfoCacheStringOverhead** - The number of bytes to add per string. The default is 24.
- **SSDocInfoCacheColumnOverhead** - The number of bytes to add per column in the *ResultSet*. The default is 40.
- **SSDocInfoCacheRowOverhead** - The number of bytes to add per row in the *ResultSet*. The default is 24.
- **SSDocInfoCacheCellOverhead** - The number of bytes to add per cell in the *ResultSet*. The default is 12.

Note: These flags are used in Site Studio 10gR4, but not in Site Studio 11gR1.

13.5.3 Flags for Controlling the SSXPathCacheEntry Cache

The algorithm that computes the amount of space that a SSXPathCacheEntry consumes in the ResourceCache, can be fine-tuned in two ways. The first way is to use a multiplier on the file size. The second is to enumerate the DOM. The following configuration flags are available:

- **SSDomCacheUseFileSize** - Uses a cache size computation based on the size of the file. The default is true.
- **SSDomCacheUseDOM** - Uses a cache size computation based on an enumeration of the XML DOM. The default value is false.
- **SSDomCacheLowerBound** - Places a lower bound on the reported cache size. The default is 6000.
- **SSDomCacheMultiplier** - Multiplies the computed cache size by the given value to arrive at a final value. The default is 0.1 because we currently recommend that the total number be multiplied by 10 to calculate the realistic size of the cache.
- **SSDomCacheNodeMultiplier** - The number of bytes to count per DOM node. The default is 12.
- **SSDomCacheStringMultiplier** - The number of bytes to multiply string lengths in the DOM by to produce the string size. The default is 2.
- **SSDomCacheStringOverhead** - The number of bytes to add per string in the DOM. The default is 24.
- **SSDomCacheDefaultFileSizeFactor** - The default value by which to multiply the file size to obtain a cache size. The default is 2.0.
- **SSDomCacheFileSizeFactors** - A comma-separated list of file sizes and multipliers that control the computed cache size.

For example, the following string

```
1000, 6.0, 10000, 2.7, 50000, 2.1, 100000, 1.9, 300000, 1.6
```

multiplies files sized [0..999] by 6.0,

multiplies files sized [1000..9999] by 2.7,
multiplies files sized [10000..49999] by 2.1,
multiplies files sized [50000..99999] by 1.9,
multiplies files sized [100000..299999] by 1.6,
(files outside the range above are multiplied by the value of
SSDomCacheDefaultFileSizeFactor.)

JSON and Contributor

This section covers the following topics:

- ["About JSON"](#) on page 14-1
- ["Passing Configuration To and From Contributor"](#) on page 14-1

14.1 About JSON

JSON is the acronym for JavaScript Object Notation, which is a language that is used for light data exchange. This makes it easy for machines to parse and generate, but it is also easy for humans to read and write. JSON is language-independent, but uses conventions based in C and other programming languages in that family (for example, C++, Java, Python, Perl, and others).

JSON is used in Site Studio as an easy method to maintain passing the data to and from Contributor.

14.2 Passing Configuration To and From Contributor

When Contributor is opened, the JSON configuration object passes flags for reset, update, and preview.

Although JSON is the data transfer format; the data binder concepts of LocalData, ResultSets, ResultSet, fields and rows still exist. Also, some configuration information is passed to the Contributor without the data binder concepts.

The console will display all activity between the Contributor and the content server. Here is an example of the JSON passed requesting that a content item be checked out.

```
2009/7/18-15:13:34:0737 - REQUEST - [index.htm] HTTP Request (http_
05201522250333165144):
  "url": http://myvmware/stellent/idcplg,
  "contentType": application/json,
  "postdata": {
    "LocalData" : {
      "IdcService" : "SS_CHECKOUT_BY_NAME",
      "dDocName" : "loafer_frontpage"
    },
    "ResultSets" : {
  }
}

2009/7/18-15:13:34:0815 - RESPONSE - [index.htm] HTTP Response (http_
05201522250333165144):
  "status": 0,
```

```

    "message": Succeeded,
    "status text": OK,
    "response text": {
"LocalData": {
"wfAction": "CHECKOUT",
"dReleaseState": "Y",
"isFinished": "0",
"dDocAccount": "",
"refreshSubjects": "",
"IsWorkflow": "",
"CurRevID": "615",
"dActionDate": "8/18/09 4:14 PM",
"refreshMonikers": "",
"dRevClassID": "71",
"dCheckoutUser": "sysadmin",
"IdcService": "SS_CHECKOUT_BY_NAME",
"changedSubjects": "documents,1250624531742",
"RedirectParams": "IdcService=CHECKOUT_
OK&dID=\u003c$dID$\u003e&CurRevCheckoutUser=\u003c$url (CurRevCheckoutUser)$\u003e&
CurRevID=\u003c$CurRevID$\u003e\u003c$if
IsWorkflowInfo$\u003e&IsWorkflowInfo=\u003c$IsWorkflowInfo$\u003e\u003c$endif$\u00
3e\u003c$if
ClientControlled$\u003e&ClientControlled=\u003c$ClientControlled$\u003e\u003c$endi
f$\u003e\u003c$if inQueueRedirect$\u003e&inQueueRedirect=1\u003c$endif$\u003e",
"changedMonikers": "",
"dSecurityGroup": "Public",
"IsNotLatestRev": "",
"dActionMillis": "36900553",
"dStatus": "RELEASED",
"dRevLabel": "9",
"dAction": "Check out",
"dID": "615",
"dWorkflowState": "",
"dPublishState": "",
"dIsCheckedOut": "1",
"dClbraName": "",
"prevReleaseState": "",
"refreshSubMonikers": "",
"isCurRevEmpty": "",
"latestID": "615",
"CurRevIsCheckedOut": "0",
"dUser": "sysadmin",
"XmlEncodingMode": "Full",
"CurRevCheckoutUser": "sysadmin",
"dDocName": "loafer_frontpage"
},
"ResultSets": {
}
}

```

For more information on the console, see [Chapter 15, "Contributor Console Window."](#)

Contributor Console Window

This section covers the following topics:

- ["About the Contributor Console Window"](#) on page 15-1
- ["Installing the Contributor Console Window"](#) on page 15-3
- ["Launching the Contributor Console Window"](#) on page 15-3
- ["Using the Contributor Console Window"](#) on page 15-4
- ["Logging Syntax"](#) on page 15-5
- ["Time Profiling"](#) on page 15-6
- ["Command Window Helper Functions"](#) on page 15-6
- ["Keyboard Commands"](#) on page 15-7

15.1 About the Contributor Console Window

The Contributor console window is the Contributor application's cross-browser and cross-platform logging mechanism. The Contributor console window is specially suited to accommodate logging and to facilitate JavaScript code execution across multiple HTML windows from within a single browser window.

All instructions logged to the Contributor console window's logging window (the top portion of the Contributor console window) display the time of execution and the context in which the instruction was executed. The lower portion of the window (the command window) is used to execute JavaScript.

Note: The Contributor console window needs to be the root browser window in order to accommodate logging and to facilitate JavaScript code execution across multiple contexts. See ["Using the Contributor Console Window"](#) on page 15-4 for more information.

This section covers the following topics:

- ["Logging Window"](#) on page 15-2
- ["Command Window"](#) on page 15-2
- ["Contributor Console Window User Interface"](#) on page 15-2

15.1.1 Logging Window

The upper portion of the Contributor console window is the logging window. Each instruction logged displays the time of execution and the context of the instruction. From the logging window's toolbar, you can clear the logging window and filter the list of logged items by selecting the different logging types.

15.1.2 Command Window

The lower portion of the Contributor console window is the command window. The command window is where arbitrary JavaScript code can be executed. Code can be executed within any context in the Contributor application by specifying a context ID in the command window's context field. A list of available contexts can be viewed by executing the following code snippet in the command window:

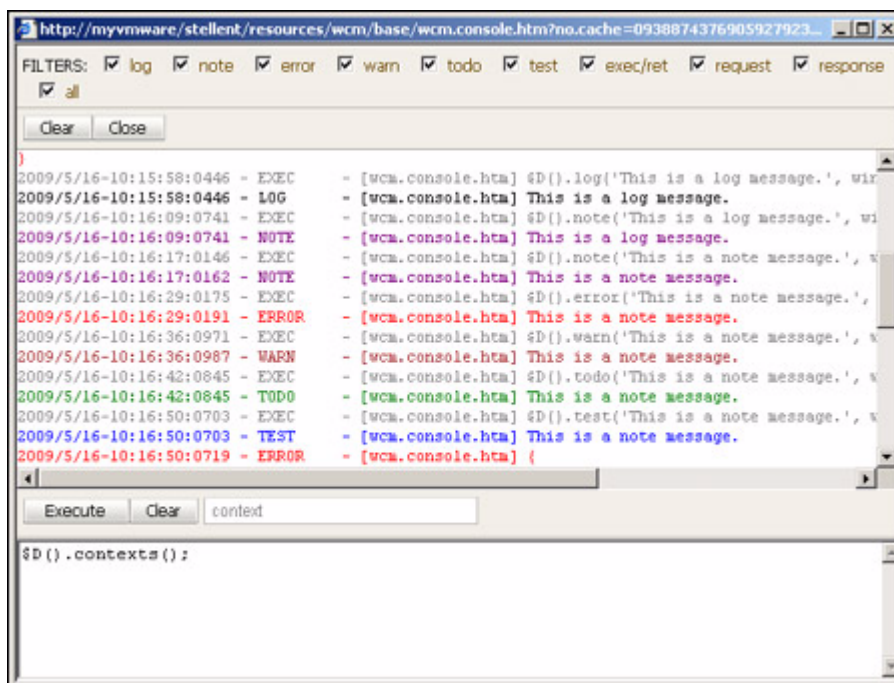
```
$D().contexts();
```

Either the context's file name or the context's ID can be used in the command window's context field. When the command window's toolbar context field is properly set, all code executed in the command window executes within the designated window object's context.

15.1.3 Contributor Console Window User Interface

The Contributor console window has a very simple interface.

Figure 15–1 Contributor Console Window



Element	Description
Filters	Select the message types to display.
Clear	Clears the logging window.

Element	Description
Close	Closes the Customize dialog.
Execute	Executes the code entered in the command window.
Clear	Clears the command window.
Context field	Enter the context (file name or ID) to execute within.
Command	Enter the command or scripts to run.

15.2 Installing the Contributor Console Window

The availability of the Contributor console window is optional and is not deployed to by default. To deploy the Contributor console window, you must do it from the General Component Information window of Site Studio Administrator, and click the (Enable) link on the Contributor Console Enabled line.

It's also possible to disable the console from the same location in Administrator.

For more information, see the *Administrator and Manager's Guide for Site Studio*.

15.3 Launching the Contributor Console Window

Once enabled, there are two ways to launch the Contributor console window: navigating to the Contributor console URL, or using the key commands.

Navigating to the Contributor Console Window's URL

Users can navigate directly to the Contributor console window if its URL is known. The following URL is where the Contributor console window usually exists in a Site Studio installation:

`http://<server_name>/<instance_name>/resources/wcm/base/wcm.console.htm`

Using Key Commands

From within Contributor, the key command Ctrl+Alt+Shift+C or Ctrl+Alt+Shift+E will launch the Contributor console window. The latter key command displays all the runtime errors in addition to opening the Contributor console window, if it is not already open.

Note: If the Contributor console window is not installed in the weblayout, then Ctrl+Alt+Shift+E opens a generic error dialog if there are runtime errors. The error dialog only displays runtime errors, not runtime logging.

Using a key command, a Contributor console window can be launched from any web page containing a reference to *wcm.js*. This means that you can launch the Contributor console window from any Contributor-enabled page that is in focus; for example, contribution mode, the contributor form, the static list's row editor, the link wizard, and so forth.

When the Contributor console window is launched via a key command, it iterates through the linked list of contexts looking for the root context. When the root context is located, the Contributor console window is launched from that page and all logging instructions iterate through the linked list of contexts and log to the open Contributor console window.

15.4 Using the Contributor Console Window

The Contributor console window must be the root browser window to accommodate logging and to facilitate JavaScript code execution across multiple contexts. If the Contributor console window is not the root browser window, or the chain of linked window objects is broken, the Contributor console window can become 'orphaned,' and will no longer work properly.

This section covers the following topics:

- ["Contributor Console Window Context"](#) on page 15-4
- ["Ensuring the Contributor Console Window is Root"](#) on page 15-4
- ["Practical Use of the Contributor Console Window"](#) on page 15-5
- ["Orphaned Console Windows"](#) on page 15-5

15.4.1 Contributor Console Window Context

Individual HTML window objects exist within every browser window, every HTML FRAME element, and every HTML IFRAME element. Site Studio hosts many contexts and many additional contexts are created and destroyed during the course of contribution.

In order for these windows to communicate with each other, the contexts must be from the same domain. Since Contributor's contexts are of the same domain, the application can dynamically iterate through a linked list of contexts to locate and log to an open Contributor console window.

15.4.2 Ensuring the Contributor Console Window is Root

You will not encounter an orphaned Contributor console window scenario if you establish the Contributor console window as the root browser window and then launch new browser windows from the Contributor console window. All new windows, as well as their spawned child windows, will log to the Contributor console window.

Establishing the Contributor Console Window as Root Browser Window

To establish the Contributor console window as the root browser window, you must navigate directly to the Contributor console window file on the server. Navigating to the Contributor console window will open a Contributor console window in a new browser window. Make sure you close the other window behind the Contributor console window if it is not needed.

Launching a New Browser Window from the Contributor Console Window

When you launch new browser windows from the Contributor console window, you ensure that the Contributor console window is the root window. To launch a new browser window from the Contributor console window, enter the following JavaScript code snippet in the command window and press **Execute**:

```
$D().launch('http://www.url.com');
```

The launched window, as well as its spawned children, will all log to the Contributor console window. All spawned windows will log to the Contributor console window if they are of the same domain.

15.4.3 Practical Use of the Contributor Console Window

There are some shortcuts to consider when establishing the Contributor console window as the root window is a re-occurring scenario. You can bookmark the URL of the Contributor console window for easy access. Once the Contributor console window is open, you can type the following in the command window to open a blank web page:

```
$D().launch();
```

Passing no parameter to the launch method will pop up a blank web page, then you can use the bookmark to navigate to the desired page.

15.4.4 Orphaned Console Windows

When you launch the Contributor console window from any Contributor-enabled page that is in focus, the web page in contribution mode is the root context, not the Contributor console window.

If the page in contributor mode is closed or refreshed, the Contributor console window is orphaned, meaning the Contributor console window no longer works. The link between the open Contributor console window and the root context (the contribution mode web page) is broken. In order to re-establish a link, you must close the orphaned Contributor console window and use the key command again to open a new Contributor console window.

For most logging and debugging purposes, you should not encounter an orphaned Contributor console window. However, if you want to avoid opening a new Contributor console window over the course of several contribution sessions, then the Contributor console window must be established as the root context.

15.5 Logging Syntax

The Contributor console window has the ability to log color-coded lines within the logging window to easily identify certain types of messages. Furthermore, these messages can be filtered by the Contributor console window.

The lines display in these colors:

Message Type	Display color
Log	Black
Note	Purple
Error	Red
Warning	Dark Red
Todo	Green
Test	Blue
Executed command	Grey
Request	Blue (italics)
Response	Dark red (italics)

Enter the following into the command window and click **Execute** to demonstrate the different logging types:

```
$D().log('This is a log message.', window);
```

```

$D().note('This is a note message.', window);
$D().error('This is an error message.', window);
$D().warn('This is a warning message.', window);
$D().todo('This is a todo message.', window);
$D().test("This is a test that succeeds." === 'This is a test that succeeds.',
window);
$D().test("This is a test that fails." !== 'This is a test that fails.',
window);
$D().request('This is a request message. A request message should be reserved for
logging server requests.', window);
$D().response('This is a response message. A response message should be reserved
for logging server responses.', window);
$D().log('The next line is an example of an execute message followed by a return
message.', window);
$D().exec("(function(){ return 'Returning this line of code.'; })() // A
self-executing function.", window);

```

15.6 Time Profiling

The time it takes to perform a certain action can be measured by using the Contributor console window's time profiling mechanism. Insert the following start and stop methods around the desired action and the minutes, seconds, and milliseconds will be logged to the open Contributor console window after the stop method executes.

```

$D().startProfiling('MY_PROFILE_ID'); // Start method
// TODO: perform some action here.
$D().stopProfiling('MY_PROFILE_ID'); // Stop method.

```

The start and end methods can reside in separate window contexts, as long as the same profile ID is passed as the first parameter.

15.7 Command Window Helper Functions

In addition to executing arbitrary JavaScript code in a given context and the previously mentioned logging types, the command window can execute known helper functions. The following is a list of built-in helper functions.

Type any of the following into the command window and click **Execute**:

Function Signature	Description
<code>\$D().clear();</code>	Clears the logging window.
<code>\$D().launch();</code>	Launches a new popup browser window with a blank page.
<code>\$D().launch(url);</code>	Launches a new popup browser window and navigates to the passed-in URL.
<code>\$D().contexts();</code>	Displays a list of available contexts in the logging window.
<code>\$D().setContext(context);</code>	Sets the command window's context field.
<code>\$D().data();</code>	Logs the runtime data of all the available contexts at a given moment.
<code>\$D().data(context);</code>	Logs the runtime data of the specified context at a given moment.

15.8 Keyboard Commands

The following is a list of available keyboard commands:

Function Signature	Description
Ctrl+Alt+Shift+C	Opens the Contributor console window if it is not already open.
Ctrl+Alt+Shift+E	Logs all errors to the command window. This command also opens the Contributor console window if it is not already open.
Ctrl+Enter	If the command window's text box is in focus, then this keyboard command executes the code within the command window in the specified context.
Ctrl+Alt+Shift+T	If the Contributor console window is in focus, this keyboard command shows (toggles) the raw log content in a text box. This allows you to easily copy and paste the content of the Contributor console's log window.

Manager Settings File

This section covers the following topics:

- ["About the Manager Settings File"](#) on page 16-1
- ["<ssm:settings> Tag"](#) on page 16-1
- ["<ssm:general> Tag"](#) on page 16-2
- ["<ssm:addSection> Tag"](#) on page 16-2
- ["<ssm:removeSection> Tag"](#) on page 16-3
- ["<ssm:moveSection> Tag"](#) on page 16-3
- ["<ssm:setErrorHandler> Tag"](#) on page 16-4
- ["<ssm:editProperties> Tag"](#) on page 16-4
- ["<ssm:editCustomProperties> Tag"](#) on page 16-5
- ["<ssm:primaryLayout> Tag"](#) on page 16-5
- ["<ssm:secondaryLayout> Tag"](#) on page 16-5
- ["<ssm:sectionOverride> Tag"](#) on page 16-6
- ["Example Manager Settings File"](#) on page 16-7

16.1 About the Manager Settings File

The manager settings file is an XML file that provides a number of configuration options that site designers can use to control the available features in Manager.

Designers can modify a number of settings in this file using a Form view that appears when the file is edited from the Site Assets pane. More advanced settings can be made when editing this file in Source view.

16.2 <ssm:settings> Tag

The `<ssm:settings>` tag is the root XML element within a Manager Settings file.

Parameters

- `xmlns`: the namespace for this XML file.

Child Tags

The `<ssm:settings>` tag can contain the following optional child tags:

- ["<ssm:general> Tag"](#) on page 16-2

- "[<ssm:addSection> Tag](#)" on page 16-2
- "[<ssm:removeSection> Tag](#)" on page 16-3
- "[<ssm:moveSection> Tag](#)" on page 16-3
- "[<ssm:setErrorHandler> Tag](#)" on page 16-4
- "[<ssm:editProperties> Tag](#)" on page 16-4
- "[<ssm:editCustomProperties> Tag](#)" on page 16-5
- "[<ssm:primaryLayout> Tag](#)" on page 16-5
- "[<ssm:secondaryLayout> Tag](#)" on page 16-5
- "[<ssm:sectionOverride> Tag](#)" on page 16-6

16.3 <ssm:general> Tag

The <ssm:general> tag contains general purpose configuration settings.

Parameters

- `contributorOnly`: true to indicate the Manager application should only be displayed when the browser is in Contribution Mode, false to indicate the Manager application can be displayed in both Contribution and Consumption Mode.
- `autoManage`: true to indicate the Manager application should automatically connect to the server and gather the information it needs to manage the site when first displayed, false to indicate the Manager application should wait for the user to click the "(manage site)" link before communicating with the server.
- `hierarchy`: controls the display of the hierarchy in the Manager application. Must be one of the following values:
 - `hide`: do not show the site hierarchy.
 - `showAll`: show the entire site hierarchy.
 - `showCurrentSectionOnly`: show only the current section (and its children)
 - `displayConsole`: true to display a console that lists all server communication between the Manager application and the Site Studio component. Used for debugging purposes only.
 - `updateTitle`: true to update the browser title bar with the currently selected section and action tab, false to leave title bar alone.
 - `updateUrl`: true to update the URL with the currently selected section and action tab state, false to leave URL alone.

Child Tags

The <ssm:general> tag contains no child tags.

16.4 <ssm:addSection> Tag

The <ssm:addSection> tag contains configuration settings for the "Add New Section" feature.

Parameters

- `hidden`: true to hide this feature from the manager.

Child Tags

The <ssm:addSection> tag can contain the following optional child tags to control the behavior of the "Add New Section" feature:

- <ssm:urlDirName> contains the following optional parameters:
 - hidden: true or false
 - disabled: true or false
- <ssm:id> contains the following optional parameters:
 - hidden: true or false
 - disabled: true or false
- <ssm:primaryUrl> contains the following optional parameters:
 - inherit: true or false for the new section to inherit the primary URL of its parent section
 - default: provides a default value for the primaryUrl for the new section.
- <ssm:secondaryUrl> contains the following optional parameters:
 - inherit: true or false for the new section to inherit the secondary URL of its parent section
 - default: provides a default value for the secondaryUrl for the new section.
- <ssm:secondaryUrlVariableField> contains the following optional parameters:
 - inherit: true or false for the new section to inherit the secondaryUrlVariableField of its parent section
 - default: provides a default value (true or false) for the secondaryUrlVariableField for the new section.

16.5 <ssm:removeSection> Tag

The <ssm:removeSection> tag contains configuration settings for the "Remove Section" feature.

Parameters

- hidden: true to hide this feature from the manager.

Child Tags

The <ssm:removeSection> tag contains no child tags.

16.6 <ssm:moveSection> Tag

The <ssm:moveSection> tag contains configuration settings for the "Move Section" feature (including drag & drop moves).

Parameters

- hidden: true to hide this feature from the manager.

Child Tags

The <ssm:moveSection> tag contains no child tags.

16.7 <ssm:setErrorHandler> Tag

The <ssm:setErrorHandler> tag contains configuration settings for the "Set Error Handler" feature.

Parameters

- `hidden`: true to hide this feature from the manager.

Child Tags

The <ssm:setErrorHandler> tag contains no child tags.

16.8 <ssm:editProperties> Tag

The <ssm:editProperties> tag contains configuration settings for editing the standard section properties.

Parameters

- `hidden`: true to hide this feature from the manager.

Child Tags

The <ssm:editProperties> tag can contain the following optional child tags:

- <ssm:id> contains the following optional parameters:
 - `hidden`: true or false
 - `disabled`: true or false
- <ssm:label> contains the following optional parameters:
 - `hidden`: true or false
 - `disabled`: true or false
- <ssm:active> contains the following optional parameters:
 - `hidden`: true or false
 - `disabled`: true or false
- <ssm:contributorOnly> contains the following optional parameters:
 - `hidden`: true or false
 - `disabled`: true or false
- <ssm:urlDirName> contains the following optional parameters:
 - `hidden`: true or false
 - `disabled`: true or false
- <ssm:urlPageName> contains the following optional parameters:
 - `hidden`: true or false
 - `disabled`: true or false
- <ssm:maxAge> contains the following optional parameters:
 - `hidden`: true or false
 - `disabled`: true or false

16.9 <ssm:editCustomProperties> Tag

The <ssm:editCustomProperties> tag contains configuration settings for the "Custom Properties" feature.

Parameters

- `hidden`: true to hide this feature from the manager.

Child Tags

The <ssm:editCustomProperties> tag contains no child tags.

16.10 <ssm:primaryLayout> Tag

The <ssm:primaryLayout> tag contains configuration settings to control the behavior of the "Choose Primary Layout" feature.

Parameters

- `hidden`: true to hide the "Layout" tab from the Manager application.
- `externalHidden`: true to hide the ability to choose an "External URL" as the primary layout for a section
- `previewHidden`: true to hide the preview IFrame on the "Layout" tab.

Child Tags

The <ssm:primaryLayout> tag can contain the following optional child tags.

- <ssm:presentation> contains a CDATA section that provides a simple HTML presentation string for displaying the layout information in the choose layout combo box. Simple inline HTML is allowed along with metadata field declarations marked as *\$field\$*. Some examples include:
 - `$dDocTitle$`
 - `$dDocName$`
 - `$dDocTitle$ (<i>$dDocName$</i>)`
 - `$dDocName$`
- <ssm:queryText> contains a CDATA section that provides the query string used for populating the choose layout combo box. In addition it can contain the following optional parameter:
 - `limitscope`: true to perform a search limited to items in the current Web site only.

16.11 <ssm:secondaryLayout> Tag

The <ssm:secondaryLayout> tag contains configuration settings to control the behavior of the "Choose Secondary Layout" feature.

Parameters

- `hidden`: true to hide the "Secondary Layout" tab from the Manager application.
- `previewHidden`: true to hide the preview IFrame on the "Secondary Layout" tab.

Child Tags

The <ssm:secondaryLayout> tag can contain the following optional child tags.

- <ssm:presentation> contains a CDATA section that provides a simple HTML presentation string for displaying the layout information in the choose layout combo box. Simple inline HTML is allowed along with metadata field declarations marked as *\$field\$*. Some examples include:
 - \$dDocTitle\$
 - \$dDocName\$
 - \$dDocTitle\$ (<i>\$dDocName\$</i>)
 - \$dDocName\$
- <ssm:queryText> contains a CDATA section that provides the query string used for populating the choose layout combo box. In addition it can contain the following optional parameter:
 - limitscope: true to perform a search limited to items in the current Web site only.

16.12 <ssm:sectionOverride> Tag

The <ssm:sectionOverride> tag contains configuration settings for a particular Web site section that override the default settings described earlier:

Parameters

- nodeId: the ID for the section that is being overridden.

Child Tags

The <ssm:sectionOverride> tag can contain any of the following optional child tags. The details of which have already been described in these sections:

- "[<ssm:addSection> Tag](#)" on page 16-2
- "[<ssm:removeSection> Tag](#)" on page 16-3
- "[<ssm:moveSection> Tag](#)" on page 16-3
- "[<ssm:setErrorHandler> Tag](#)" on page 16-4
- "[<ssm:editProperties> Tag](#)" on page 16-4
- "[<ssm:editCustomProperties> Tag](#)" on page 16-5
- "[<ssm:primaryLayout> Tag](#)" on page 16-5
- "[<ssm:secondaryLayout> Tag](#)" on page 16-5

The parameters and contents of the subsections within the <ssm:sectionOverride> are identical to their use in the general case with the following exception:

- The <ssm:moveSection> can contain 2 additional parameters that apply to the section being overridden only:
 - source: false if this section cannot be used as a source of a move action
 - target: false if this section cannot be used as a target of a move action

16.13 Example Manager Settings File

```

<?xml version='1.0' encoding='utf-8' ?>
<ssm:settings xmlns:ssm="http://www.stellent.com/sitestudio/managersettings/">
  <ssm:general contributorOnly="false"
    autoManage="true"
    hierarchy="showAll"
    displayConsole="false"
    updateTitle="true"
    updateUrl="true" />

  <ssm:addSection hidden="false">
    <ssm:urlDirName hidden="false" disabled="false" />
    <ssm:id hidden="false" disabled="false" />
    <ssm:primaryUrl inherit="true" default="" />
    <ssm:secondaryUrl inherit="true" default="" />
    <ssm:secondaryUrlVariableField inherit="true" default="" />
  </ssm:addSection>

  <ssm:removeSection hidden="false" />
  <ssm:moveSection hidden="false" />
  <ssm:setErrorHandler hidden="false" />

  <ssm:editProperties hidden="false" >
    <ssm:id hidden="false" disabled="true" />
    <ssm:label hidden="false" disabled="false" />
    <ssm:active hidden="false" disabled="false" />
    <ssm:contributorOnly hidden="false" disabled="false" />
    <ssm:urlDirName hidden="false" disabled="false" />
    <ssm:urlPageName hidden="false" disabled="false" />
    <ssm:maxAge hidden="false" disabled="false" />
  </ssm:editProperties>

  <ssm:editCustomProperties hidden="false" />

  <ssm:primaryLayout hidden="false" externalHidden="false"
    previewHidden="false">
    <ssm:presentation>
      <![CDATA[$dDocTitle (<i>$dDocName</i>)]>
    </ssm:presentation>
    <ssm:queryText limitScope="true">
      <![CDATA[]]>
    </ssm:queryText>
  </ssm:primaryLayout>

  <ssm:secondaryLayout hidden="false" previewHidden="false">
    <ssm:presentation>
      <![CDATA[$dDocTitle (<i>$dDocName</i>)]>
    </ssm:presentation>
    <ssm:queryText limitScope="true">
      <![CDATA[]]>
    </ssm:queryText>
  </ssm:secondaryLayout>

  <ssm:sectionOverride nodeId="42">
    <ssm:addSection hidden="true" />
    <ssm:removeSection hidden="true" />
    <ssm:moveSection hidden="true" />
    <ssm:setErrorHandler hidden="true" />
    <ssm:setErrorHandler hidden="true" />

```

```
<ssm:editProperties hidden="false" />
<ssm:editCustomProperties hidden="true" />
<ssm:primaryLayout hidden="true" />
<ssm:secondaryLayout hidden="true" />
</ssm:sectionOverride>

</ssm:settings>
```

Content Tracker Integration

Site Studio and Content Tracker can be used together to generate reports for a Site Studio web site. You can customize the integration of the two products using several configuration flags. The configuration flags control the data that is reported.

Using Content Tracker (the Data Engine Control Center), you can control the data that is tracked and where that data is tracked for each service call.

This section covers the following topics:

- ["Tracked Data"](#) on page 17-1
- ["Configuration Flags"](#) on page 17-2

17.1 Tracked Data

There are two types of site accesses that are tracked:

- **Hierarchy access:** Access is typically by section ID and is targeted at the section regardless of the actual content at that section.
- **Content access:** Access is directed specifically at a piece of content regardless of where that content actually appears.

Hierarchy Access	Content Access	SS Variable	CT Field Name
dDocName of layout page	dDocName of data file, native doc or fragment library	targetPage	sc_scs_dDocName
dID of layout page	dID of data file, native doc or fragment library	targetdID	sc_scs_dID
Site id	Site id	targetSiteId	extField_1
Node id	Node id	targetNodeId	extField_2
Is secondary page	Is secondary page	targetIsSecondary	extField_3
Website object type	Website object type	targetWebsiteObjectType	extField_4
contribution mode	contribution mode	SSContributor	extField_5
[Not used]	dDocName of layout that data file, native doc or fragment is being used on	targetContentId	extField_6
Site relative url	Site relative url	siteRelativeUrl	extField_7

If errors occur, then the following values are included in the Content Tracker database:

Error Type	SS Variable	CT Field Name
Error	SS Variable	CT Field Name
The original site relative URL that was received at the server and could not be resolved.	originalSiteRelativeUrl	extField_8
Whether or not the original URL was invalid	invalidSiteRelativeUrl	extField_9

If the site has an error handler page and that page gets accessed, the *siteRelativeUrl* will be set to the URL of the error page. The original URL that caused the error will be set as the *originalSiteRelativeUrl* value. The *invalidSiteRelativeUrl* value will be set to "1."

If the site has no error handler page and an error occurs, then the original URL will be recorded as the *siteRelativeUrl*, and the *invalidSiteRelativeUrl* value will be set to "1". In this case the *originalSiteRelativeUrl* value will be empty.

17.2 Configuration Flags

The following configuration flags (set in `config.cfg`) control the data that Site Studio tracks and how it tracks that information.

- **SSTrackContentAccess=[true | false]**

This specifies whether content access is enabled. The default setting is "true." If you set it to false, then only hierarchy access is tracked.
- **SSTrackFragmentAccess=[true | false]**

This specifies whether to record access to fragment libraries. The default setting is "false."
- **SSTrackerReportNumDaysBack=[n | 7]**

This specifies the number of days to go back for a content access type report that is accessed using a popup menu from the contribution icon. The default setting is "7" and "n" is a positive number (in days).

Upgrading Pre-7.5 Web Sites

This section covers the following topics:

- ["Introduction"](#) on page A-1
- ["What the Automated Upgrade Does"](#) on page A-2
- ["Upgrading Your Content Servers"](#) on page A-2
- ["Performing Additional Steps Manually"](#) on page A-6

A.1 Introduction

If you are upgrading from a Site Studio release before 7.5, you must upgrade your Web sites before you can use them with Site Studio 11gR1. This is because Site Studio versions 7.5 and 10gR3 have some important architectural changes, including:

- The site hierarchy is stored in a project file and no longer relies on folders. As a result, Oracle Content Server's folders features are no longer required.
- The Web site URLs display as a logical path and suffix instead of as a CGI-based address displaying the `SS_GET_PAGE` service (see ["SS_GET_PAGE"](#) on page 11-18 for more information). As a result, you see friendlier, path-based URLs.
- Layout pages no longer use the `<base>` tag. Therefore, hyperlinks and references that rely on the base tag must be modified.
- The *siteId* and *root node id* are no longer synonymous.

It is important to realize that, after upgrading, the upgraded projects will work as "legacy" projects in Site Studio 11gR1; that is, they will use the pre-10gR4 architecture and they will not take advantage of the new architecture and features in Site Studio 10gR4 and 11gR1. The same is true of projects that were created using Site Studio 7.5 or 10gR3. They do not need to be upgraded per se and you can use them with Site Studio 11gR1, but they will continue to function in "legacy" (that is, pre-10gR4) mode.

A.2 What the Automated Upgrade Does

When you upgrade your pre-7.5 Site Studio release and your Web sites, the following tasks are performed automatically:

Action	Description
Folders-based sites upgraded to project-based sites	The existing hierarchy in the folder structure is reproduced in the project file. The root "dCollectionName" is used as the "siteLabel," the root "dCollectionID" is used as the "siteId," the "originalCollectionID" project attribute is set, and the site type is transferred from the root section to the project.
Custom section properties in new sites updated	The custom section properties of type "siteid" and "url" are updated (adding friendly URLs, where necessary).
Fragment instance parameters in layout pages updated	Parameters of type "managedurl" and "url" are updated.
Metadata populated	If the Create Project Files option is enabled, the "xWebsiteSection" values are populated (derived from "xCollectionID").
Links in layout pages and data files updated	If the Upgrade Layouts and Upgrade Data Files options are enabled, the weblayout links in layout pages and contributor data files are updated to include the HttpRelativeWebRoot token; optionally, the javascript links are updated.
Navigation updated	The navigation files for the Web site are regenerated.

Note: Custom elements cannot be upgraded automatically. See ["Updating Your Custom Elements"](#) on page A-10 for more information.

A.3 Upgrading Your Content Servers

The task of site upgrade begins with upgrading the Site Studio component on each of the content servers you are using, and then upgrading the Web sites stored on the content servers:

- ["Upgrading Sites on a Single Content Server Instance"](#) on page A-3
- ["Upgrading Sites on Multiple Content Server Instances"](#) on page A-3
- ["Performing a Full Upgrade"](#) on page A-4
- ["Performing a Minimal Upgrade"](#) on page A-5

Although the Folders component is not used in Site Studio version 7.5 or later, you must retain folders during the upgrade of your Web sites so that each site can be migrated from a folders-based hierarchy to a project-based hierarchy.

You can then disable the Folders component. If you want to continue using Folders, you must configure them with the appropriate metadata (see ["Assigning a Web Site Section to Your Folders"](#) on page A-10).

Note: When you follow the upgrade steps, every Web site on the server is upgraded. If you want to upgrade only selected sites, then you must create a copy of the other sites on another server.

A.3.1 Upgrading Sites on a Single Content Server Instance

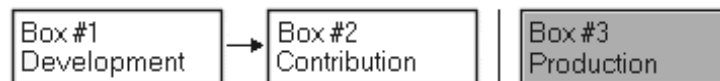
If your Web sites are stored on a single content server, the upgrade consists of:

1. Installing the new Site Studio component (having first uninstalled the old component).
2. Performing a full upgrade on the content server. For more information, see ["Performing a Full Upgrade"](#) on page A-4.
3. Performing additional steps manually that are not handled by the automated upgrade. For more information, see ["Performing Additional Steps Manually"](#) on page A-6.

A.3.2 Upgrading Sites on Multiple Content Server Instances

You may have sites on multiple content servers, each serving a different purpose, such as a development server, a contribution server, and a production server.

Figure A-1 Multiple Content Server Instance Illustration



The content on each server (source server) gets copied to the next server (target server) using the Archiver/Replicator utility. As such, it is important to carefully plan and upgrade the sites on each server without encountering replication problems.

On the First Two Instances of the Content Server

- Stop replication between the content servers
- Install the new Site Studio component.

On the Source Instance of the Content Server

- Perform a full upgrade of your sites. For more information, see ["Performing a Full Upgrade"](#) on page A-4.

Then:

- Perform additional steps manually (steps not handled by the automatic upgrade). For more information, see ["Performing Additional Steps Manually"](#) on page A-6.

On the Target Instance of the Content Server

- Perform a minimal upgrade of your sites. For more information, see ["Performing a Minimal Upgrade"](#) on page A-5.

On Both Instances of the Content Server

- Start replication again between the content servers.

Once the new component has been installed on all instances of the content server and Web sites have been upgraded as indicated above, you can begin replicating your sites again.

You can use the replication feature in Site Studio (see the *Administrator and Manager's Guide for Site Studio*). Or, if you have been using Archiver/Replicator and want to continue using it, you can do so as long as you modify the archive query to include Site Studio project files.

On the Next Target Content Server (Downstream in Replication)

- Stop replication between the source and target content servers.

Note: In this case, your source server (Box 2) was the target server in the previous steps, and your target server (Box 3) is the next server down the line (downstream) in your replication.

- Install the new Site Studio component.

Then:

- Perform a minimal upgrade of your sites. For more information, see "[Performing a Minimal Upgrade](#)" on page A-5.

Then:

- Start replication again between the source and target content servers.

Repeat this last procedure for each target instance of the content server downstream in your replication.

A.3.3 Performing a Full Upgrade

A full upgrade of the content server is required in the case of single-server setup. It is also required for the source server in the case of a multi-server setup. (All other servers in a multi-server setup require a minimal upgrade.)

When you upgrade your site, Site Studio turns your existing folder-based site into a project-based site. When it does this, it creates a project file as a managed item in the content server. As such, you must identify the metadata that you would like assigned to the project file that will represent each Web site.

During the upgrade process, the content server attempts to index content that gets changed, which could take considerable time and resources. You may want to temporarily disable automatic indexing before you begin the upgrade process and then re-enable it when you are done. (See the Oracle Content Server administration documentation for further details.)

You should have already installed and enabled the new Site Studio component on the server before you start a full upgrade.

To perform a full upgrade, perform these tasks:

1. Log on to the content server as an administrator, open the **Administration** page, and then the **Site Studio Administration** page.

2. Click **Set Project Default Document Information**.

The Set Project Default Document Information page is displayed. This is where you assign the default metadata for the new projects that you create in Site Studio.

3. Once you have specified the metadata values, click **Update**.

This returns you to the Site Studio Administration page, where you can begin the upgrade process.

4. Click **Manage Web Sites**.

5. Click **Go to Web Sites Update Page**. (This option displays only when older Web sites are detected.)

6. Click **Advanced Options** to specify site upgrade options.

Figure A-2 Advanced Upgrade Options Screen

7. Choose the following for a full upgrade:
 - Select **Create Project files**.
 - Select **Upgrade Layouts**.
 - Select **Upgrade Data Files**.
 - Select **Convert Hyperlinks** and choose a link format:
 - **To Server-Side Links:** Links contain the coded identity of the target location using server-side script.
 - **To Path-Based URLs:** Links contain the full path to the target location.
8. Click **Set Options** to return to the Upgrade Legacy Web Sites page.
9. Click **Start Upgrade**.

You see the individual files that must be upgraded on this page. Wait until you see the message that says that the upgrade process was completed.

Note: The site upgrade automatically updates the site hierarchy and its many links, and the Web Sites menu in the content server now lists your sites.

A.3.4 Performing a Minimal Upgrade

A minimal upgrade is required in the case of a multi-server setup and applies to all target servers; that is, the server that has Web sites being replicated to it.

You should have already installed and enabled the new Site Studio component on the server before you start a minimal upgrade.

To perform a minimal upgrade, perform these tasks:

1. Log onto the content server as an administrator, open the **Administration** page, and then the **Site Studio Administration** page.
2. Click **Set Default Project Document Information**.
The Set Project Default Document Information page is displayed. This is where you assign the default metadata for the new projects that you create in Site Studio.
3. Once you have specified the metadata values, click **Update**.
This returns you to the Site Studio Administration page, where you can begin the upgrade process.
4. Click **Manage Web Sites**.

5. Click **Go to Web Sites Update Page**. (This option displays only when older Web sites are detected.)
6. Click **Advanced Options** to specify site upgrade options (Figure A-2).
7. Select **Create Project files**.

Note: This upgrades project files and populates the "Web Site Section" metadata value.

8. Click **Set Options** to return to the Upgrade Legacy Web Sites page.
9. Click **Start Upgrade**.
 Wait until you see the message that says that the upgrade process was completed. The Web Sites menu in the content server now lists your sites.

A.4 Performing Additional Steps Manually

Once you have upgraded your Web sites, there are still several steps that you must perform manually, including the following:

- ["Updating the Site Navigation"](#) on page A-6
- ["Rebuilding the Content Server Index"](#) on page A-6
- ["Updating Your Custom Fragments"](#) on page A-7
- ["Updating Your Custom Elements"](#) on page A-10
- ["Assigning a Web Site Section to Your Folders"](#) on page A-10
- ["Updating JSP Code"](#) on page A-11

A.4.1 Updating the Site Navigation

After upgrading a pre-7.5 Web site, you must update its navigation files. You can do this in Designer (using the **Update Navigation** button) or on the Site Studio Administration page (specifically, the Manage Web Sites page). This step is necessary for Contributor to function correctly on the site.

A.4.2 Rebuilding the Content Server Index

After upgrading a pre-7.5 Web site, you may need to rebuild the content server index. If your content server has been set up to use database search and indexing (full-text or metadata-only), you do not need to rebuild the search index. If you are using a different search engine, you must rebuild the search index. This is necessary because Site Studio updates the xWebsiteSection metadata field for all content items residing in folders on your site.

Caution: Rebuilding the search index may be a very time-consuming process, depending on the number of content items managed by your Oracle Content Server instance. It is therefore recommended that you perform this rebuild during off-peak hours of Oracle Content Server use (typically at night or on the weekend).

See the Oracle Content Server administration documentation for more information on rebuilding the index.

A.4.3 Updating Your Custom Fragments

Most of the manual updates you must perform after upgrading your pre-7.5 Web site involve modifying your custom fragments. If you are currently using the predefined fragments that came with Site Studio, you do not have to do this because an updated version of each fragment is included with each Site Studio release.

Most likely, you have customized the fragments or introduced new ones to meet a specific purpose for your organization. There are three things you must do with the fragments so that they work in the latest version:

- [Modifying Links That Rely on the <base> Tag](#)
- [Modifying Obsolete SS_GET_PAGE / JavaScript Links](#)
- [Updating GET_SEARCH_RESULTS](#)

A.4.3.1 Modifying Links That Rely on the <base> Tag

The <base> tag that points to the content server's web-accessible directory ('weblayout') is no longer used. During the site upgrade, Site Studio updates the necessary code in your layout pages and contributor data files, but you must perform this step manually in your custom fragments and scripts.

You can do this by re-authoring hand-coded links that are relative to the URL in the <base> tag and use the `HttpRelativeWebRoot` server-side variable instead.

Example

Say, you have a link to a graphic that looks something like this:

```

```

You must then replace it with the following:

```

```

A.4.3.2 Modifying Obsolete SS_GET_PAGE / JavaScript Links

If any existing fragments use `SS_GET_PAGE`, `javascript:link`, or `javascript:nodelink` style hyperlinks, you may want to change them to path-based URLs to take advantage of their many benefits. (For more information, see the *User's Guide for Site Studio Designer*.)

Example

Say, you have a link that looks something like this:

```
<a href="javascript:nodelink(42);">link</a>
```

You must then replace it with the following:

```
<a href="<!--ssServerRelativeSiteRoot-->products/servers/index.htm">link</a>
```

A.4.3.3 Updating GET_SEARCH_RESULTS

Any fragment that used the `GET_SEARCH_RESULTS` service will continue to work, but will not take advantage of the features provided by Site Studio 7.5 and 10gR3 until it is upgraded to use the `SS_GET_SEARCH_RESULTS` service (for more information, see "[SS_GET_SEARCH_RESULTS](#)" on page 11-22).

Using the `SS_GET_SEARCH_RESULTS` service has several advantages:

- **limitscope logic** (now provided by the service and not required in the fragment): This limits the search results to only those items within the current Web site.
- **dontshowinlists logic** (now provided by the service and not required in the fragment): This limits the search results to only those items that have not been removed from lists by contributors.
- **ssUrl**: This column provides a friendly URL for each row in the search results.

Fragments that use the `GET_SEARCH_RESULTS` service are typically dynamic list fragments and search results navigation fragments. The updates required differ depending on the version of the Site Studio release that you are upgrading from:

- If you have been using Site Studio version 6.5, and you have customized dynamic lists or search results fragments using that version (for example, by copying a Site Studio fragment and adding custom code to it), you will have used code that performs limitscope logic using the old `xWebsiteID` metadata field.
- If you have been using Site Studio version 7.2, and you have customized dynamic lists or search results fragments using that version (for example, by copying a Site Studio fragment and adding custom code to it), you will have used code that performs limitscope logic using the new `xWebsites` metadata field. In addition, you will have used code that performs *dontshowinlists* logic using the new `xDontShowInListsForWebsites` metadata field.

In both cases above, you must update those fragments to remove the old `limitscope` and `dontshowinlists` logic from them and to use the new `SS_GET_SEARCH_RESULTS` service, which now provides this functionality internally.

Example

In Site Studio 6.5, the Standard Dynamic List fragment includes the following code for the `SSLimitScope` parameter. This should be removed:

```
<!--$QueryText=eval(ssQueryText)-->
<!--$if sslimitScope like "true"-->
  <!--$if strEquals(QueryText, '')-->
    <!--$QueryText='xWebSiteID=' & siteId-->
  <!--$else-->
    <!--$QueryText=(' & QueryText & ') and (xWebSiteID=' & siteId & ')-->
  <!--$endif-->
<!--$endif-->
```

In Site Studio 7.2, the Standard Dynamic List fragment includes the following code for the `SSLimitScope` parameter. This should be removed:

```
<!--$QueryText=eval(ssQueryText)-->
<!--$if sslimitScope like "true"-->
  <!--$if strEquals(QueryText, '')-->
    <!--$QueryText='xWebsites &lt;contains&gt; ' & siteId-->
  <!--$else-->
    <!--$QueryText=(' & QueryText & ') and (xWebsites &lt;contains&gt;' &
siteId & ')-->
  <!--$endif-->
<!--$endif-->

<!--$if strEquals(QueryText, '')-->
  <!--$QueryText='not(xDontShowInListsForWebsites &lt;contains&gt; ' & siteId &
')-->
<!--$else-->
```

```

    <!--$QueryText=(' & QueryText & ') and not(xDontShowInListsForWebsites
    &lt;contains&gt; ' & siteId & ') '-->
    <!--$endif-->

```

Once the old limitscope logic is removed from the fragment, change the `GET_SEARCH_RESULTS` service call to use `SS_GET_SEARCH_RESULTS`. Before you invoke the `SS_GET_SEARCH_RESULTS` service, however, you should set the following parameter values:

Parameter	Description
<code>ssLimitScope</code>	Specifies that the limitscope logic should be applied by the <code>SS_GET_SEARCH_RESULTS</code> service. Typically this true/false value is supplied by a fragment parameter value.
<code>ssDontShowInLists</code>	Specifies that the dontshowinlists logic should be applied by the <code>SS_GET_SEARCH_RESULTS</code> service. Typically this true/false value is set to "true" in all fragments.
<code>ssTargetNodeId</code>	Specifies the node ID that is used to display the search results. The "ssTargetSiteId" can also be used to generate links to other Web sites on the content server. If the "ssTargetSiteId" is not specified, the generated link assumes the same site that originated the link.
<code>ssTargetSiteId</code>	Specifies the site ID that is used to display the search results. The "ssTargetNodeId" parameter must also be used to fully qualify the target node.
<code>ssSourceNodeId</code>	Indicates the node ID for the current page containing the link.
<code>ssSourceSiteId</code>	Indicates the site ID for the current page containing the link.
<code>ssWebsiteObjectType</code>	Specifies that the search results should be limited to a specific Website Object Type. Typically you leave this value empty.
<code>ssUserSearchText</code>	Specifies any user text to perform a full text search. Typically, this only applies to Search Results fragments where the value is provided by a consumer entering a value in a Search Box fragment.

When looping through the results of the `SS_GET_SEARCH_RESULTS` service call, you typically use the new `ssUrl` column of the result set if you want to create hyperlinks to that item. This ensures that full path-based URLs are used instead of cryptic ID-based URLs.

Additionally, these URLs should be appended with parameters that describe the source location of the link. This allows error pages to be generated properly when there are invalid links.

The following parameters should be affixed to the URLs.

Parameter	Description
<code>ssSourceNodeId</code>	Declares the source node ID. Used to generate friendly URLs if both <code>ssTargetNodeId</code> and <code>xWebsiteSection</code> are blank.
<code>ssSourceSiteId</code>	Declares the source site ID. This allows the error page to be displayed if the target page cannot be found.

Here is a simplified example using Idoc script:

```

<!-- New params for SS_GET_SEARCH_RESULTS -->
<!--$ssLimitScope="true"-->
<!--$ssDontShowInLists="true"-->
<!--$ssTargetNodeId=""-->

```

```

<!--$ssTargetSiteId=" "-->
<!--$ssSourceNodeId=nodeId-->
<!--$ssSourceSiteId=siteId-->
<!--$ssWebsiteObjectType=" "-->
<!--$ssUserText=" "-->

<!--$executeService("SS_GET_SEARCH_RESULTS")-->

<!--$loop SearchResults-->
  <a href="<!--$ssUrl-->?ssSourceSiteId=<!--$siteId-->&ssSourceNodeId=
<!--nodeId-->">
    <!--$dDocTitle-->
  </a><br><br>
<!--$endloop-->

```

For more details, refer to the dynamic list and search results fragments that are provided with the Site Studio product.

A.4.4 Updating Your Custom Elements

Any custom element forms created using a Site Studio release before 7.5 are not compatible with Site Studio 11gR1. They must be manually upgraded and re-authored. The primary reason for not maintaining backward compatibility is Site Studio's prior dependency on Internet Explorer's proprietary `window.external` functionality (due to the ActiveX control used for the Contributor). This functionality was removed from Site Studio as a result of the browser-independent, JavaScript-based Contributor application that is used in Site Studio 10gR3 (10.1.3.3.3) and higher.

A.4.5 Assigning a Web Site Section to Your Folders

Site Studio no longer uses the Oracle Content Server folder features (Folders component) to organize and manage your site hierarchy. If a Web site created in a Site Studio release before 7.5 is upgraded, content that resides in a folder has a new metadata value ("Web Site Section") assigned to it so that it is recognized as part of the upgraded site.

Any new content added to the folder, after the upgrade, will not receive this metadata value. As such, if you want to continue using folders to add content to your site, you must assign a "Web Site Section" value to each folder.

To assign a "Web Site Section" value, perform these tasks:

1. Log onto the content server as a user with write (RW) access to the folder you want to update.
2. Open the **Browse Content** tray or menu, and then open the **Web Sites** tree.
3. Select the Web site to want to update.
4. Click **Folder Information** for the folder you want to change.
5. Select the **Update** action.
6. Click **Browse** next to the Web Site Section field.
7. Choose the corresponding Web site section.
8. Click **OK**.
9. Click **Update**.
10. Repeat these steps for each folder that you want to map to a Web site section in Site Studio.

A.4.6 Updating JSP Code

If you have created JSP code based on `SiteStudio.SSNavigationBean` and `SiteStudio.SSNavigationNode` objects, references to these objects must be changed so that "sitestudio" is all lowercase, as follows:

- `sitestudio.SSNavigationBean`
- `sitestudio.SSNavigationNode`

Index

A

addNode() method, 5-2
angle bracket, 4-2
ASP node, 9-13
associating with a region definition, 7-1
asynchronous communication
 with custom elements, 8-3
automatic upgrade of pre-10gR4 projects, A-2
available contexts, 15-2

B

<base> tag, A-7
bracket
 angle, 4-2
 differences, 4-2
 square, 4-2
build process, 13-3
 and customizations, 13-5
 executing from a shell, 13-4
 HTML files, 13-3
 optimized code, 13-4
 requirements, 13-3
 running from the command line, 13-4
build script debugging, 13-4
build system, 13-3

C

Child Tags, 6-7
client-side JavaScript, 6-4
color-coding in console window, 15-5
command window, 15-2
 available contexts, 15-2
 context field, 15-2
 JavaScript execution, 15-2
 toolbar, 15-2
component resources, 5-3
configuration flags, 13-2, 17-2
 DisableSiteStudioContribution, 12-5
 ShowSiteStudioMissingDataFileErrors, 12-5
 SiteStudioValidateConversionsDefinitions, 12-6
 SiteStudioValidateDataFiles, 12-7
 SiteStudioValidateElementDefinitions, 12-6
 SiteStudioValidatePlaceholderDefinitions, 12-6

SiteStudioValidateProjects, 12-7
SiteStudioValidateRegionDefinitions, 12-6
SSAccessDeniedHeader, 12-7
SSAccessDeniedReplacementHeader, 12-7
SSAccessDeniedUserAgentExceptions, 12-8
SSAccommodateWelcomeFile, 12-8
SSAdditionalNavResultSetFields, 12-8
SSAddSecurityIDValues, 12-8
SSAfterProjectLoadedProperties, 12-9
SSAllowDynamicDefinitions, 12-9
SSAllowEmptyUrlPageName, 12-9
SSAllowNotModifiedHeader, 12-9
SSAltTagFieldName, 12-10
SSAlwaysRecordServerConfig, 12-10
SSAssumeXmllsUtf8, 12-10
SSAutoCheckinBusyTimeout, 12-10
SSBackupCollectionName, 12-11
SSCacheControlOverride, 12-11
SSCanGenerateUniqueDataFiles, 12-11
SSChangeAccessDeniedHeaders, 12-11
SSCheckAssignedContentAccess, 12-12
SSCheckBrowserForSiteRoot, 12-12
SSClearDefinitionArchiveWebsites, 12-12
SSCompressorArguments, 12-13
SSCompressorCommand, 12-13
SSCompressorDir, 12-13
SSCompressorJar, 12-13
SSCompressorMainClass, 12-14
SSCompressorTimeout, 12-14
SSCompressorTimerInterval, 12-14
SSCompressorWaitForever, 12-14
SSContributorSourceDir, 12-15
SSCustomNodePropertyDefsPermissions, 12-15
SSDefaultDocumentsFields, 12-15
SSDefaultEditor, 12-15
SSDefaultExternalDocNamePrefix, 12-16
SSDefaultExternalDocNameSuffix, 12-16
SSDefaultExternalServerRelativeSiteRoot, 12-16
SSDefaultExternalUrlPrefix, 12-16
SSDefaultExternalUrlSuffix, 12-17
SSDefaultPlaceholderDefinition, 12-17
SSDefaultRegionTemplate, 12-17
SSDefaultUrlPageName, 12-17
SSDetectIncludeFileEncoding, 12-18
SSDICPlaceholderDefinition, 12-18
SSDirectDeliveryExtensions, 12-18

- SSDirectDeliveryOverrideProperty, 12-18
- SSDirectDeliveryProperty, 12-19
- SSDirectDeliveryRequiredExtensions, 12-19
- SSDisableDeferredNodeExpansion, 12-19
- SSDisableIncludeXmlCache, 12-19
- SSDisableLinkResolutionSiteLocking, 12-20
- SSDisableProjectDeferredNodeExpansion, 12-20
- SSDomCacheDefaultFileSizeFactor, 12-20
- SSDomCacheFileSizeFactors, 12-20
- SSDomCacheLowerBound, 12-21
- SSDomCacheMultiplier, 12-21
- SSDomCacheNodeMultiplier, 12-21
- SSDomCacheStringMultiplier, 12-22
- SSDomCacheStringOverhead, 12-22
- SSDomCacheUseDOM, 12-22
- SSDomCacheUseFileSize, 12-22
- SSEditorDebugLevel, 12-22
- SSEnableASPSupport, 12-23
- SSEnableDirectDelivery, 12-23
- SSEnableExtranetLookCompatibility, 12-23
- SSEnableFolioEditing, 12-23
- SSEnableFormEditing, 12-24
- SSEnableJavaScriptCompressor, 12-24
- SSEnableUpperCaseColumnsCheck, 12-24
- SSGenerateUniqueNodeIds, 12-24
- SSHidePrimaryFileInContributor, 12-25
- SSHttpAbsoluteHelpRoot, 12-25
- SSHttpLayerManager, 12-25
- SSIdocMarker, 12-25
- SSIgnoreMaxAgeNodeProperties, 12-25
- SSIgnoreNoProjectDefaultMetadataMessage, 12-26
- SSIgnoreReadyToReplicate, 12-26
- SSImportOnlyLatestRevs, 12-26
- SSIncludeInactiveNodesInNavResultSet, 12-26
- SSIncludeInactiveNodesInNavXML, 12-27
- SSIncludeRegionTemplatesInDefinitionBundles, 12-27
- SSIncludeXmlTransformFormat, 12-27
- SSIncludeXmlTransformIndent, 12-27
- SSJavaExecutablePath, 12-27
- SSJSONContentType, 12-28
- SSLoadCustomElementsWithOnDemandEditors, 12-28
- SSLoadProjectsAtStartup, 12-28
- SSLoadUncompressedFckSource, 12-29
- SSManuallyValidateNodeIdUniqueness, 12-29
- SSMaxNodeIdLength, 12-29
- SSMaxSiteIdLength, 12-29
- SSMaxSitesMenuItems, 12-30
- SSMaxTemplateEvaluationStack, 12-30
- SSMigrationCollectionName, 12-30
- SSOmitFragmentLibrariesInArchiverQueries, 12-30
- SSOnDemandEditorsThresholdCount, 12-31
- SSPreFillUrlDirNamesDuringUpgrade, 12-31
- SSProjectAutoCheckinInterval, 12-31
- SSProjectLoadFailureTracingSection, 12-31
- SSProjectReleaseSleepTime, 12-32
- SSProjectReleaseWaitTime, 12-32
- SSQuickDiffDefaultRegionTemplate, 12-32
- SSShowAssignmentTooltips, 12-32
- SSSQLUseContains, 12-32
- SSStoppedSiteResponsePageDocName, 12-33
- SSSuppressAddToWebsite, 12-33
- SSSuppressLargeCssOptimization, 12-33
- SSTempProjectLifetime, 12-34
- SSTitleTagFieldName, 12-34
- SSTrackContentAccess, 12-34
- SSTrackFragmentAccess, 12-34
- SSUrlFieldName, 12-35
- SSUrlFixupExceptions, 12-35
- SSUrlPageNames, 12-35
- SSUseAbsoluteRedirects, 12-35
- SSUseCallbackTrackingForASP, 12-36
- SSUseDefaultDocNamePrefix, 12-36
- SSUseDefaultServerRelativeSiteRoot, 12-36
- SSUseDefaultUrlPrefix, 12-36
- SSUseMissingLinkTargetFallback, 12-36
- SSUseOnDemandContributionModeMenus, 12-36
- SSUseUrlSegmentSessionInfo, 12-37
- SSValidateCustomElements, 12-37
- SSWebFilterIgnoreList, 12-38
- SSWebLayoutUrlUsesDocNames, 12-38
- SSWelcomeFile, 12-38
- SSWelcomeFileReplacement, 12-38
- console window, 14-2
 - available contexts, 15-2
 - children, 15-4
 - color-coding, 15-5
 - command window, 15-2
 - context, 15-2
 - establishing as root window, 15-5
 - helper functions, 15-6
 - installation, 15-3
 - keyboard commands, 15-7
 - launching from Contributor, 15-3
 - logging types, 15-2, 15-5, 15-6
 - logging window, 15-2
 - orphans, 15-5
 - root context, 15-5
 - root window, 15-4
 - URL, 15-5
- content access, 17-1, 17-2
- content reuse
 - rules, 11-19
- Content Server
 - folders functionality, A-2
 - minimal site upgrade, A-5
- content server services
 - SS_ADD_NODE, 11-7
 - SS_ADD_WEBSITE_ID, 11-7
 - SS_BATCH_DECODE_LINK, 11-7
 - SS_CHECKIN_FRAGMENT_LIBRARY, 11-8
 - SS_CHOOSE_WEBSITE_SECTION, 11-8
 - SS_CHOOSE_WEBSITES, 11-8
 - SS_CLEAR_PREVIEW, 11-8
 - SS_CLEAR_REGION_ASSOCIATIONS, 11-9
 - SS_CLEAR_WEBSITE_ID, 11-9

SS_COMMIT_SITE_CHANGES, 11-9
 SS_CREATE_NEW_SITE_EX2, 11-9
 SS_CREATE_SITE_NAV_JS, 11-10
 SS_DECODE_LINK, 11-10
 SS_DELETE_NODE, 11-11
 SS_DOC_INFO_LATEST, 11-11
 SS_EDIT_NATIVE_DOCUMENT, 11-11
 SS_GET_ADMIN_PAGE, 11-12
 SS_GET_ALL_CUSTOM_NODE_PROP_DEFS, 11-12
 SS_GET_ALL_NODE_PROPERTIES, 11-12
 SS_GET_ALL_SITE_DOMAINS, 11-12
 SS_GET_ALL_SITE_PROPERTIES, 11-13
 SS_GET_ALL_SITES_EX2, 11-13
 SS_GET_CONFIG_INFO, 11-13
 SS_GET_CONTRIBUTOR_CONFIG, 11-13
 SS_GET_CONTRIBUTOR_STRINGS, 11-14
 SS_GET_DC_RULES, 11-14
 SS_GET_DOCUMENT_LABELS, 11-14
 SS_GET_DOCUMENT_USAGE, 11-14
 SS_GET_ENVIRONMENT_PROPERTY_NAMES, 11-15
 SS_GET_FIRST_NODE_ID, 11-15
 SS_GET_FRIENDLY_URL, 11-15
 SS_GET_LINK, 11-16
 SS_GET_LINK_MANAGEMENT_REPORT, 11-16
 SS_GET_LINK_WIZARD_CONFIG, 11-16
 SS_GET_LINK_WIZARD_CONFIG_WITH_SITE, 11-17
 SS_GET_NODE_LINK, 11-17
 SS_GET_NODE_PROPERTY, 11-18
 SS_GET_PAGE, 11-18
 SS_GET_PLACEHOLDER_SWITCH_CONTENT_CONFIG, 11-21
 SS_GET_REGION_ASSOCIATIONS, 11-22
 SS_GET_REGION_DEFINITION_ELEMENTS, 11-22
 SS_GET_RELATIVE_NODE_ID, 11-22
 SS_GET_SEARCH_RESULTS, 11-22
 SS_GET_SITE_AS_XML_EX2, 11-23
 SS_GET_SITE_ASSET_CATEGORIES, 11-24
 SS_GET_SITE_CHANGE_MONITOR_TOKEN, 11-24
 SS_GET_SITE_DEFINITION, 11-24
 SS_GET_SITE_DEFINITION_FOR_USER, 11-24
 SS_GET_SITE_DOMAINS, 11-25
 SS_GET_SITE_FRAGMENT_ASSET_REPORT, 11-25
 SS_GET_SITE_INFO, 11-25
 SS_GET_SITE_PROPERTY, 11-25
 SS_GET_SITE_PUBLISH_REPORT, 11-26
 SS_GET_SITE_REPORT, 11-26
 SS_GET_SWITCH_CONTENT_CONFIG, 11-26
 SS_GET_UNIQUE_NODE_SITE_ID, 11-26
 SS_GET_VERSION, 11-27
 SS_GET_WEBLAYOUT_URL, 11-27
 SS_IS_JS_NAV_OUT_OF_DATE, 11-27
 SS_MAP_FRIENDLY_NAME, 11-27
 SS_MOVE_NODE, 11-28
 SS_PARSE_FRIENDLY_URL, 11-28
 SS_PREPARE_PREVIEW, 11-28
 SS_PUBLISH_THIS_PAGE, 11-29
 SS_REMOVE_WEBSITE_ID, 11-29
 SS_SET_ALL_CUSTOM_NODE_PROP_DEFS, 11-29
 SS_SET_ELEMENT_DATA, 11-29
 SS_SET_ENVIRONMENT_PROPERTY_NAMES, 11-30
 SS_SET_NODE_PROPERTY, 11-30
 SS_SET_NODES_PROPERTIES, 11-30
 SS_SET_PREVIEW_ELEMENT_DATA, 11-31
 SS_SET_SITE_ASSET_CATEGORIES, 11-31
 SS_SET_SITE_DOMAINS, 11-31
 SS_SET_SITE_PROPERTIES, 11-32
 SS_SET_SITE_PROPERTY, 11-32
 SS_SWITCH_REGION_ASSOCIATION, 11-32
 SS_VALIDATE_WEBSITE_OBJECT, 11-33
 WCM_BEGIN_EDIT_SESSION, 11-34
 WCM_EDIT_DATA_FILE, 11-34
 WCM_PLACEHOLDER, 11-33
 Content Tracker, 17-1
 and the Data Engine Control Center, 17-1
 content tracker
 tracked data, 17-1
 context, 15-4
 file name, 15-2
 context field, 15-2
 command window, 15-2
 executing code, 15-2
 in the toolbar, 15-2
 context ID
 command window, 15-2
 contributino mode
 key command, 2-7
 contribution mode
 query string, 2-8
 session cookie, 2-8
 contribution region
 fragment definition file, 6-5
 Contributor
 and JSON, 14-1
 context, 15-4
 enabling on-demand editors, 13-2
 executing code in context, 15-2
 using build system, 13-3
 Contributor console window, 15-1
 command window, 15-1
 context, 15-4
 contexts, 15-3
 establishing the root browser window, 15-4
 key commands, 15-3
 launching new browser windows, 15-4
 logging window, 15-1
 orphaned windows, 15-4
 root browser window, 15-4
 root content, 15-3
 URL, 15-3
 user interface, 15-2
 contributor form, 8-1, 8-3

- communication with custom element, 8-3
 - registered functions, 8-3
- Contributor mode
 - SSContributor parameter, 11-20
- Contributor services
 - SS_ADD_WEBSITE_ID, 11-7
 - SS_CHOOSE_WEBSITE_SECTION, 11-8
 - SS_CHOOSE_WEBSITES, 11-8
 - SS_CLEAR_PREVIEW, 11-8
 - SS_DECODE_LINK, 11-10
 - SS_EDIT_NATIVE_DOCUMENT, 11-11
 - SS_GET_CONFIG_INFO, 11-13
 - SS_GET_CONTRIBUTOR_CONFIG, 11-13
 - SS_GET_CONTRIBUTOR_STRINGS, 11-14
 - SS_GET_DOCUMENT_USAGE, 11-14
 - SS_GET_FRIENDLY_URL, 11-15
 - SS_GET_PLACEHOLDER_SWITCH_CONTENT_CONFIG, 11-21
 - SS_GET_SEARCH_RESULTS, 11-22
 - SS_GET_SWITCH_CONTENT_CONFIG, 11-26
 - SS_PUBLISH_THIS_PAGE, 11-29
 - SS_REMOVE_WEBSITE_ID, 11-29
 - SS_SET_ELEMENT_DATA, 11-29
 - SS_SET_PREVIEW_ELEMENT_DATA, 11-31
- conversions definition
 - xWebsiteObjectType value, 3-3
- Conversions Definition property, 7-2
- conversions rule
 - and wcmDynamicConversion, 7-1
- <convert> tag, 6-10
 - language attribute, 6-10
- CSS
 - metadata, 3-2
 - optimization, 13-3
 - optimizing, 13-2
- custom configuration script
 - xWebsiteObjectType value, 3-2
- custom element, 8-1, 12-37
 - detecting legacy forms, 8-5
 - ElementAPI, 8-1
 - in the Contributor form, 8-1
 - legacy compatibility, 8-5
 - metadata, 3-2
 - upgrading legacy custom elements, 8-6
 - validation, 12-37
- custom element form, 12-37
 - xWebsiteObjectType value, 3-2
- custom elements, A-10
 - asynchronous communication, 8-3
 - dependent scripts, 8-2
 - ElementAPI, 8-2
- custom fragments, A-7
- custom node properties, 9-12
- custom section properties
 - sitenavigation.xml file, 6-5
- custom section property, 6-4
 - accessing their values, 6-4
 - using client-side JavaScript to access, 6-4
 - using server-side JavaScript to access, 6-5
- <customgui> tag, 6-11

- JavaScript methods, 6-11
 - window.external.GetValue() method, 6-11
 - window.external.OnCancel() method, 6-11
 - window.external.OnOK() method, 6-11
 - window.external.SetValue() method, 6-11

D

- data binder concept, 14-1
- data binder formats
 - and JSON, 14-1
- Data Engine Control Center, 17-1
- data file
 - xWebsiteObjectType value, 3-1
- data transfer
 - and JSON, 14-1
- debugging
 - build script, 13-4
- definition
 - element definition, 2-6
- definitions
 - region definition, 2-4
- dependent scripts
 - ElementAPI, 8-2
- Designer services
 - SS_ADD_NODE, 11-7
 - SS_ADD_WEBSITE_ID, 11-7
 - SS_CHECKIN_FRAGMENT_LIBRARY, 11-8
 - SS_CLEAR_PREVIEW, 11-8
 - SS_COMMIT_SITE_CHANGES, 11-9
 - SS_CREATE_NEW_SITE_EX2, 11-9
 - SS_CREATE_SITE_NAV_JS, 11-10
 - SS_DELETE_NODE, 11-11
 - SS_DOC_INFO_LATEST, 11-11
 - SS_EDIT_NATIVE_DOCUMENT, 11-11
 - SS_GET_ADMIN_PAGE, 11-12
 - SS_GET_ALL_CUSTOM_NODE_PROP_DEFS, 11-12
 - SS_GET_ALL_SITE_DOMAINS, 11-12
 - SS_GET_ALL_SITE_PROPERTIES, 11-13
 - SS_GET_ALL_SITES_EX2, 11-13
 - SS_GET_CONFIG_INFO, 11-13
 - SS_GET_DC_RULES, 11-14
 - SS_GET_DOCUMENT_USAGE, 11-14
 - SS_GET_ENVIRONMENT_PROPERTY_NAMES, 11-15
 - SS_GET_FRIENDLY_URL, 11-15
 - SS_GET_NODE_PROPERTY, 11-18
 - SS_GET_PAGE, 11-18
 - SS_GET_REGION_DEFINITION_ELEMENTS, 11-22
 - SS_GET_SEARCH_RESULTS, 11-22
 - SS_GET_SITE_AS_XML_EX2, 11-23
 - SS_GET_SITE_ASSET_CATEGORIES, 11-24
 - SS_GET_SITE_CHANGE_MONITOR_TOKEN, 11-24
 - SS_GET_SITE_DEFINITION, 11-24
 - SS_GET_SITE_DEFINITION_FOR_USER, 11-24
 - SS_GET_SITE_DOMAINS, 11-25
 - SS_GET_SITE_FRAGMENT_ASSET_

- REPORT, 11-25
- SS_GET_SITE_INFO, 11-25
- SS_GET_SITE_PROPERTY, 11-25
- SS_GET_SITE_REPORT, 11-26
- SS_GET_UNIQUE_NODE_SITE_ID, 11-26
- SS_GET_VERSION, 11-27
- SS_IS_JS_NAV_OUT_OF_DATE, 11-27
- SS_MAP_FRIENDLY_NAME, 11-27
- SS_MOVE_NODE, 11-28
- SS_PARSE_FRIENDLY_URL, 11-28
- SS_PREPARE_PREVIEW, 11-28
- SS_REMOVE_WEBSITE_ID, 11-29
- SS_SET_ALL_CUSTOM_NODE_PROP_DEFS, 11-29
- SS_SET_ENVIRONMENT_PROPERTY_NAMES, 11-30
- SS_SET_NODE_PROPERTY, 11-30
- SS_SET_NODES_PROPERTIES, 11-30
- SS_SET_SITE_ASSET_CATEGORIES, 11-31
- SS_SET_SITE_DOMAINS, 11-31
- SS_SET_SITE_PROPERTIES, 11-32
- SS_SET_SITE_PROPERTY, 11-32
- SS_SWITCH_REGION_ASSOCIATION, 11-32
- SS_VALIDATE_WEBSITE_OBJECT, 11-33
- <designview> tag, 6-13
- determining managed file web addresses, 9-19
- determining native documents, 9-11
- dialog
 - Enable Metadata Modification, 2-5
 - Placeholder Definition, 2-3
 - Region Content Options, 2-5
 - Switch Region Content, 2-5
- DisableSiteStudioContribution, 12-5
- displaying runtime errors, 15-3
- DOC_INFO cache, 13-5
- DOM, 13-5, 13-6
- dynamic conversion, 7-1
 - inline, 9-11
 - ssIncludeDynamicConversion, 9-10
 - using ssInclInlineDynamicConversion, 9-10, 9-11
 - using ssIsNativeDoc, 9-11
- Dynamic Converter
 - converting a native document, 7-1

E

- Edge Side Includes, 11-34
- element
 - extracting from a managed XML file, 9-8
- element context, 9-6
- element definition, 2-6
 - xWebsiteObjectType value, 3-2
- <element> tag, 6-13
- ElementAPI
 - dependent scripts, 8-2
 - loading, 8-1
- ElementAPI object, 8-1
- elements, updating custom --, A-10
- Enable Metadata Modification dialog, 2-5
- error handler page

- siteRelativeUrl, 17-2
- error handler section, 11-20
- escaping problems
 - angle vs square bracket, 4-2
 - avoiding, 4-2, 4-3
 - single vs. double quote, 4-3
- establishing the root browser window, 15-4
- example of Manager Settings File, 16-7
- extracting elements, 9-8

F

- file size multiplier, 13-5, 13-6
- filter, 15-2
- flag
 - controlling the SSXPathCacheEntry cache, 13-6
 - for the DOC_INFO cache, 13-5
 - SSComputeDocInfoCacheSize, 13-5
 - SSContributorSourceDir, 13-5
 - SSDisableDeferredNodeExpansion, 13-5
 - SSDocInfoCacheCellOverhead, 13-6
 - SSDocInfoCacheColumnOverhead, 13-6
 - SSDocInfoCacheLowerBound, 13-5
 - SSDocInfoCacheMultiplier, 13-6
 - SSDocInfoCacheRowOverhead, 13-6
 - SSDocInfoCacheStringMultiplier, 13-6
 - SSDocInfoCacheStringOverhead, 13-6
 - SSDomCacheDefaultFileSizeFactor, 13-6
 - SSDomCacheFileSizeFactors, 13-6
 - SSDomCacheLowerBound, 13-6
 - SSDomCacheMultiplier, 13-6
 - SSDomCacheNodeMultiplier, 13-6
 - SSDomCacheStringMultiplier, 13-6
 - SSDomCacheStringOverhead, 13-6
 - SSDomCacheUseDOM, 13-6
 - SSDomCacheUseFileSize, 13-6
 - SSOnDemandEditorsThresholdCount, 13-2
 - SSTrackContentAccess, 17-2
 - SSTrackerReportNumDaysBack, 17-2
 - SSTrackFragmentAccess, 17-2
- flags, 13-2
 - for content tracker, 17-2
- folders on content server, A-1, A-2
- folders on Oracle Content Server, A-10
- forcing data file association., 9-6
- fragment
 - assets, 6-1, 6-2
 - complex, 6-1
 - contents of, 6-2
 - explained, 6-1
 - library, 6-2
 - parameter definitions, 6-2
 - simple, 6-1
 - snippets, 6-2, 6-4
 - structure of, 6-2
 - wcmFragment, 9-7
 - xWebsiteObjectType value, 3-2
- fragment definition file, 6-2, 6-5
- <fragment> element, 6-2
- <fragments> root element, 6-2

- <fragments> tag, 6-6
- fragment libraries, 17-2
- fragment library, 6-2, 6-6
 - adding to content server, 6-3
 - fragment definition file, 6-2
- fragment parameters, 6-2
- fragment snippet, 6-1, 6-4
 - adding by reference to a layout page, 6-4
 - adding directly to a layout page with special markup, 6-4
 - optional design-time view, 6-13
- <fragment> tag, 6-6
 - attributes, 6-6
 - child tags, 6-7
 - <element> child tag, 6-13
 - <parameter> child tag, 6-8
 - <snippet> child tag, 6-12
- <fragmentinstance> tag, 6-13
- fragments
 - folder location, 10-2
- <fragments> tag, 6-6
- <fragment> child tag, 6-6
- fragments, updating custom --, A-7
- full site upgrade, A-4

G

- GET_SEARCH_RESULTS service, A-7, A-9
- graphic files
 - and metadata values, 3-2

H

- helper functions in the console window, 15-6
- hierarchy
 - in the Web site, 9-15
- hierarchy access, 17-2
 - and content tracker, 17-1
- HttpASPPath, 10-2
- HttpFragmentsRoot, 6-3, 10-2
- HttpRelativeFragmentsRoot, 6-3, 10-2
- HttpRelativeWebsitesRoot, 10-1
- HttpWebsitesRoot, 10-1

I

- Idoc Script, 9-2
 - for accessing custom section property values, 6-5
 - server-side, 6-5
 - SS_GET_ALL_NODE_PROPERTIES service, 6-5
 - ssGetNodeProperty (name), 6-5
 - ssGetNodeProperty (nodeId, name), 6-5
 - wcmPlaceholder, 9-2
- Idoc script
 - and dynamic conversion, 7-2
 - ssGetAllSites, 9-18
 - ssGetCoreMajorVersion, 9-13
 - ssGetDocInfo, 9-9
 - ssGetFirstNodeId, 9-15
 - ssGetNodeLabel, 9-17
 - ssGetNodeLabelPath, 9-17

- ssGetNodeProperty, 9-12
- ssGetRelativeNodeId, 9-15
- ssGetServerRelativePath, 9-16
- ssGetServerRelativeUrl, 9-16
- ssGetSiteProperty, 9-14
- ssGetUrlPageName, 9-17
- ssGetWebsiteName, 9-14
- ssGetWebsiteNodeType, 9-13
- ssGetXmlNodeCount, 9-9
- ssInclInlineDynamicConversion, 9-10, 9-11
- ssIncludeDynamicConversion, 9-10
- ssIncludeXml, 9-8
- ssIsNativeDoc, 9-11
- ssLink, 9-18
- ssLoadSiteNavResultSet, 9-15
- ssNodeLink, 9-18
- ssRandom, 9-12
- ssSplitString, 9-13
- ssWeblayoutUrl, 9-19
- wcmDynamicConversion, 9-6
- wcmDynamicList, 9-5
- wcmElement, 9-3
- wcmFragment, 9-7
- wcmIncludeElement, 9-6
- wcmListElement, 9-4
- wcmListEnd, 9-4
- wcmListRowCount, 9-5
- wcmListStart, 9-4
- wcmUrl, 9-7

- Idoc Script extensions, 9-1

- Idoc variable

- for displaying error information, 11-20

- Idoc variables, 10-1

- HttpASPPath, 10-2

- HttpFragmentsRoot, 10-2

- HttpRelativeFragmentsRoot, 10-2

- HttpRelativeWebsitesRoot, 10-1

- HttpWebsitesRoot, 10-1

- SS_SERVER_NAME, 10-2

- ssServerRelativeSiteRoot, 10-2

- image

- xWebsiteObjectType value, 3-2

- installing the console window, 15-3

- internal web site information structures, 12-20

- invalidSiteRelativeUrl, 17-2

- IsDirty() function, 8-3

J

- JavaScript

- auto-generated files, 5-1

- client-side, 6-4

- code execution, 15-4

- code execution across multiple windows, 15-1

- ElementAPI, 8-4

- for accessing custom section property values, 6-4

- methods, 6-5

- methods for providing contribution

- functionality, 5-4

- optimization, 13-3

- optimizing, 13-2
- sitenavigation.js, 5-1, 5-3
- wcm.toggle.js, 5-4
- JavaScript code
 - in the command window, 15-2
- JavaScript interpreter, 13-3
- JavaScript Object Notation, 14-1
- JavaScript WCM library, 8-2
- JavaServer Pages (JSP), A-11
- JDK, 13-3
- JSON
 - example code passed, 14-1
- JSP, A-11

K

- key command
 - contribution mode, 2-7
 - session cookie, 2-8
- key commands, 15-3

L

- layout page
 - adding a fragment to with special markup, 6-4
 - adding a snippet directly to, 6-4
 - complex fragment for, 6-1
 - <fragmentinstance> tag, 6-13
 - referencing a fragment snippet in, 6-4
 - servicing up the correct one, 11-18
 - simple fragment for, 6-1
- legacy projects, A-1
- link formats
 - using wcmUrl, 9-7
- Link Wizard services
 - SS_ADD_WEBSITE_ID, 11-7
 - SS_DECODE_LINK, 11-10
 - SS_GET_CONTRIBUTOR_STRINGS, 11-14
 - SS_GET_FRIENDLY_URL, 11-15
 - SS_GET_LINK_WIZARD_CONFIG, 11-16
 - SS_GET_LINK_WIZARD_CONFIG_WITH_SITE, 11-17
 - SS_GET_SITE_DEFINITION_FOR_USER, 11-24
- link() method in sitenavigation.js, 5-3
- links
 - <base> tag, A-7
 - SS_GET_PAGE, A-7
- loading the ElementAPI, 8-1
- lock contention, 12-20
- logging instructions
 - contexts, 15-3
- logging types, 15-2, 15-5, 15-6
- logging window, 15-2
 - filter, 15-2
 - logging types, 15-2
 - toolbar, 15-2

M

- m_href node property, 5-2
- m_id node property, 5-2

- m_label node property, 5-2
- m_level node property, 5-2
- m_parent node property, 5-2
- m_subNodes node property, 5-2
- managed XML file, extracting an element, 9-8
- Manager services
 - SS_ADD_NODE, 11-7
 - SS_DELETE_NODE, 11-11
 - SS_GET_ALL_CUSTOM_NODE_PROP_DEFS, 11-12
 - SS_GET_SEARCH_RESULTS, 11-22
 - SS_GET_SITE_DEFINITION_FOR_USER, 11-24
 - SS_GET_SITE_PROPERTY, 11-25
 - SS_MOVE_NODE, 11-28
 - SS_SET_NODE_PROPERTY, 11-30
 - SS_SET_SITE_PROPERTY, 11-32
- manager settings
 - xWebsiteObjectType, 3-2
- Manager Settings File
 - example, 16-7
- manager settings file, 16-1
- manager settings file child tags
 - <ssm:addSection>, 16-2
 - <ssm:editCustomProperties>, 16-5
 - <ssm:editProperties>, 16-4
 - <ssm:general>, 16-2
 - <ssm:moveSection>, 16-3
 - <ssm:primaryLayout>, 16-5
 - <ssm:removeSection>, 16-3
 - <ssm:secondaryLayout>, 16-5
 - <ssm:sectionOverride>, 16-6
 - <ssm:setErrorHandler>, 16-4
- memory usage, 13-5
- metadata
 - conversions definition, 3-3
 - CSS, 3-2
 - custom configuration script, 3-2
 - custom element form, 3-2
 - custom fields, 3-1
 - data file, 3-1
 - dataProperty, 2-5
 - element definition, 3-2
 - fragment, 3-2
 - image, 3-2
 - manager settings, 3-2
 - native document, 3-2
 - other, 3-3
 - page template, 3-1
 - placeholder definition, 3-2
 - project file, 3-2
 - region definition, 3-2
 - region template, 3-2
 - script, 3-2
 - stylesheet, 3-2
 - subtemplate, 3-2
 - validation script, 3-2
 - xRegionDefinition, 2-4
- metadata field
 - xDontShowListForWebsites, 3-4
 - xWebsiteObjectType, 3-1

- xWebsites, 3-3
- metadata fields
 - Region Definition, 3-4
 - Web Site Section, A-10
- minimal site upgrade, A-5
- minimal upgrade, A-5
- multiple content servers, upgrading sites on --, A-3

N

- native document
 - dynamic conversion of, 9-10
 - xWebsiteObjectType value, 3-2
- native documents
 - and Dynamic Converter, 7-1
 - common errors, 7-2
 - conversions definition, 7-2
 - dynamic conversion, 7-1
 - implementation, 7-2
 - using Idoc script in conversion, 7-2
 - using ssInchlineDynamicConversion, 9-10, 9-11
 - using ssIncludeDynamicConversion, 9-10
 - using ssIsNativeDoc, 9-11
 - using wcmDynamicConversion, 7-1
 - wcmDynamicConversion, 9-6
- navigation links, SS_GET_PAGE service, 11-18
- navigation on sites, A-6
- NavNode object
 - addNode(), 5-2
 - definition, 5-1
 - m_href, 5-2
 - m_id, 5-2
 - m_label, 5-2
 - m_level, 5-2
 - m_parent, 5-2
 - m_subNodes, 5-2
- node property, 9-12
- node, see 'section'
- nodeId
 - parameter, 11-18
- nodelink(), 5-3

O

- on-demand editors, 13-2
- on-demand Web sites, 13-1
- optimization
 - build process, 13-4
 - build system, 13-3
 - requirements, 13-3
- optimizing
 - Contributor code, 13-2, 13-3
- optimizing CSS and JavaScript, 13-2
- optimizing JavaScript and CSS, 13-3
- <option> tag, 6-9
 - value attribute, 6-9
- Oracle Content Server
 - folders functionality, A-1, A-10
 - full site upgrade, A-4
 - search index, A-6

- upgrading for legacy projects, A-2
- upgrading sites, A-3
- originalSiteRelativeUrl, 17-2
- orphaned console windows, 15-5
- orphaned Contributor console window, 15-4
- other
 - xWebsiteObjectType, 3-3

P

- page template
 - xWebsiteObjectType value, 3-1
- page templates, 2-2
- parameter
 - ssDocName, 11-19
 - ssSourceNodeId, 11-19
 - ssTargetNodeId, 11-19
- <parameter> tag, 6-8
 - <convert> child tag, 6-10
 - cssstyle parameter type, 6-8
 - <customgui> child tag, 6-11
 - default value, 6-9
 - <option> child tag, 6-9
 - <querytext> child tag, 6-9
 - <validate> child tag, 6-9
- path-based URL, 10-2
 - using ssLink, 9-18
 - using ssNodeLink, 9-18
- path-based URLs, A-7
- placeholder definition
 - definitions
 - placeholder definition, 2-3
 - example code, 2-3
 - xWebsiteObjectType value, 3-2
 - Placeholder Definition dialog, 2-3
- placeholders, 2-2
- PreviewId parameter, 11-20
- primary page, 2-2
- project file
 - xWebsiteObjectType value, 3-2
- projects
 - upgrading pre-7.5 projects, A-1
 - property, custom section, 6-4

Q

- query string
 - setting contribution mode, 2-8
- <querytext> tag, 6-9

R

- random number generator, 9-12
- rebuilding search index, A-6
- Region Content Options dialog, 2-5
- region definition, 2-4
 - element context, 9-6
 - example code, 2-4
 - xWebsiteObjectType value, 3-2
- Region Definition metadata field, 3-4
- region template

- xWebsiteObjectType value, 3-2
- region templates, 2-3
- ResourceCache, 13-6
- reusing content, rules for, 11-19
- Rhino, 13-3
- root browser window, 15-4
- root content
 - for launching the Contributor console window, 15-3
- runtime errors
 - display, 15-3

S

- script
 - custom, 3-2
 - validation, 3-2
 - xWebsiteObjectType value, 3-2
- script extension
 - ssIncludeXml(), 6-4
- script extensions, 9-2
- script links, 9-18
- search index, A-6
- search results, A-7, A-9
- secondary page, 2-2
- secondaryUrl property, parsing, 11-18
- section
 - for handling errors, 11-20
 - previously called "node", 11-19
- section node, 9-13
- section properties
 - primary page, 2-2
 - secondary page, 2-2
- <section> tag, 5-3
- server relative URL, 9-16
- server-relative URL, 9-16
- server-side script links
 - using ssLink, 9-18
 - using ssNodeLink, 9-18
- services
 - Contributor, 11-1
 - Designer, 11-2
 - link wizard, 11-4
 - Manager, 11-3
 - switch content, 11-4
- services for search results, A-7, A-9
- services, see 'content server services'
- session cookie
 - contribution mode, 2-8
 - key command, 2-8
- ShowSiteStudioMissingDataFileErrors, 12-5
- single content server, upgrading sites on --, A-3
- site hierarchy
 - declaring NavNode objects, 5-2
 - XML definition of, 5-3, 5-4
- site navigation, A-6
- site node, 9-13
- site property
 - using ssGetSiteProperty, 9-14
- site sections and folders, A-10

- Site Studio
 - websites folder, 10-1
- Site Studio
 - component resources, 5-3
 - pre-10gR4 projects, A-1
 - using JSON, 14-1
 - websites folder, 10-1
- Site Studio metadata fields
 - Region Definition, 3-4
 - Web Site Section, A-10
- <site> tag, 5-3
- sitenavigation.js, 5-1, 5-3
 - declaration of NavNode objects, 5-2
 - link() method, 5-3
 - NavNode object definition, 5-1
 - nodelink() method, 5-3
- sitenavigation.xml file, 5-3, 5-4, 6-5
- siteRelativeUrl
 - and the error handler page, 17-2
- sites, see 'Web sites'
- SiteStudioValidateConversionsDefinitions, 12-6
- SiteStudioValidateDataFiles, 12-7
- SiteStudioValidateElementDefinitions, 12-6
- SiteStudioValidatePlaceholderDefinitions, 12-6
- SiteStudioValidateProjects, 12-7
- SiteStudioValidateRegionDefinitions, 12-6
- <snippet> tag, 6-12
 - <designview> child tag, 6-13
- square bracket, 4-2
- SS_GET_ALL_NODE_PROPERTIES service0, 6-5
- SS_GET_PAGE, A-7
- SS_GET_PAGE service, 11-18
 - error during call, 11-20
 - nodeId parameter, 11-18
 - optional URL parameters, 11-19
 - parameters required, 11-18
 - PreviewId parameter, 11-20
 - ssContributor parameter, 11-20
 - ssDocName parameter, 11-18
- SS_GET_SEARCH_RESULTS service, A-7, A-9
- SS_SERVER_NAME, 10-2
- SSAccessDeniedHeader, 12-7
- SSAccessDeniedReplacementHeader, 12-7
- SSAccessDeniedUserAgentExceptions, 12-8
- SSAccommodateWelcomeFile, 12-8
- SSAdditionalNavResultSetFields, 12-8
- SSAddSecurityIDValues, 12-8
- SSAfterProjectLoadedProperties, 12-9
- SSAllowDynamicDefinitions, 12-9
- SSAllowEmptyUriPageName, 12-9
- SSAllowNotModifiedHeader, 12-9
- SSAltTagFieldName, 12-10
- SSAlwaysRecordServerConfig, 12-10
- SSAssumeXmlIsUtf8, 12-10
- SSAutoCheckinBusyTimeout, 12-10
- SSBackupCollectionName, 12-11
- SSCacheControlOverride, 12-11
- SSCanGenerateUniqueDataFiles, 12-11
- SSChangeAccessDeniedHeaders, 12-11
- SSCheckAssignedContentAccess, 12-12

SSCheckBrowserForSiteRoot, 12-12
SSCheckWebsiteObjectSecurity, 12-12
SSClearDefinitionArchiveWebsites, 12-12
SSCompressorArguments, 12-13
SSCompressorCommand, 12-13
SSCompressorDir, 12-13
SSCompressorJar, 12-13
SSCompressorMainClass, 12-14
SSCompressorTimeout, 12-14
SSCompressorTimerInterval, 12-14
SSCompressorWaitForever, 12-14
SSComputeDocInfoCacheSize, 13-5
SSContributor parameter, 11-20
SSContributorSourceDir, 12-15, 13-5
SSCustomNodePropertyDefsPermissions, 12-15
SSDefaultDocumentsFields, 12-15
SSDefaultEditor, 12-15
SSDefaultExternalDocNamePrefix, 12-16
SSDefaultExternalDocNameSuffix, 12-16
SSDefaultExternalServerRelativeSiteRoot, 12-16
SSDefaultExternalUrlPrefix, 12-16
SSDefaultExternalUrlSuffix, 12-17
SSDefaultPlaceholderDefinition, 12-17
SSDefaultRegionTemplate, 12-17
SSDefaultUrlPageName, 12-17
SSDetectIncludeFileEncoding, 12-18
SSDICPlaceholderDefinition, 12-18
SSDirectDeliveryExtensions, 12-18
SSDirectDeliveryOverrideProperty, 12-18
SSDirectDeliveryProperty, 12-19
SSDirectDeliveryRequiredExtensions, 12-19
SSDisableDeferredNodeExpansion, 12-19, 13-5
SSDisableIncludeXmlCache, 12-19
SSDisableLinkResolutionSiteLocking, 12-20
SSDisableProjectDeferredNodeExpansion, 12-20
SSDocInfoCacheCellOverhead, 13-6
SSDocInfoCacheColumnOverhead, 13-6
SSDocInfoCacheLowerBound, 13-5
SSDocInfoCacheMultiplier, 13-6
SSDocInfoCacheRowOverhead, 13-6
SSDocInfoCacheStringMultiplier, 13-6
SSDocInfoCacheStringOverhead, 13-6
ssDocName, 11-19
ssDocName parameter, 11-18
SSDomCacheDefaultFileSizeFactor, 12-20, 13-6
SSDomCacheFileSizeFactors, 12-20, 13-6
SSDomCacheLowerBound, 12-21, 13-6
SSDomCacheMultiplier, 12-21, 13-6
SSDomCacheNodeMultiplier, 12-21, 13-6
SSDomCacheStringMultiplier, 12-22, 13-6
SSDomCacheStringOverhead, 12-22, 13-6
SSDomCacheUseDOM, 12-22, 13-6
SSDomCacheUseFileSize, 12-22, 13-6
SSEditorDebugLevel, 12-22
SSEnableASPSupport, 12-23
SSEnableDirectDelivery, 12-23
SSEnableExtranetLookCompatibility, 12-23
SSEnableFolioEditing, 12-23
SSEnableFormEditing, 12-24
SSEnableJavaScriptCompressor, 12-24
SSEnableUpperCaseColumnsCheck, 12-24
ssErrorCode, 11-20
ssErrorMessage, 11-20
SSGenerateUniqueNodeIds, 12-24
ssGetAllSites, 9-18
ssGetCoreMajorVersion, 9-13
ssGetDocInfo, 9-9
ssGetFirstNodeId, 9-15
ssGetNodeLabel, 9-17
ssGetNodeLabelPath, 9-17
ssGetNodeProperty, 9-12
ssGetNodeProperty (name), 6-5
ssGetNodeProperty (nodeId, name), 6-5
ssGetRelativeNodeId, 9-15
ssGetServerRelativePath, 9-16
ssGetServerRelativeUrl, 9-16
ssGetSiteProperty, 9-14
ssGetUrlPageName, 9-17
ssGetWebsiteName, 9-14
ssGetWebsiteNodeType, 9-13
ssGetXmlNodeCount, 9-9
SSHidePrimaryFileInContributor, 12-25
SSHttpAbsoluteHelpRoot, 12-25
SSHttpLayerManager, 12-25
SSIdocMarker, 12-25
SSIgnoreMaxAgeNodeProperties, 12-25
SSIgnoreNoProjectDefaultMetadataMessage, 12-26
SSIgnoreReadyToReplicate, 12-26
SSImportOnlyLatestRevs, 12-26
ssIncDynamicConversion, 9-10, 9-11
ssIncInlineDynamicConversion, 9-10, 9-11
ssIncludeDynamicConversion, 9-10
SSIncludeInactiveNodesInNavResultSet, 12-26
SSIncludeInactiveNodesInNavXML, 12-27
SSIncludeRegionTemplatesInDefinitionBundles, 12-27
ssIncludeXml, 9-8
ssIncludeXml(), 6-4
 parameters, 6-4
SSIncludeXmlTransformFormat, 12-27
SSIncludeXmlTransformIndent, 12-27
<ssinfo> XML data island
 <fragmentinstance> tag, 6-13
 structure for fragment instance and
 parameters, 6-13
ssIsNativeDoc, 9-11
SSJavaExecutablePath, 12-27
SSJSONContentType, 12-28
ssLink, 9-18, 12-20
SSLoadCustomElementsWithOnDemandEditors, 12-28
SSLoadProjectsAtStartup, 12-28
ssLoadSiteNavResultSet, 9-15
SSLoadUncompressedFckSource, 12-29
SSManuallyValidateNodeIdUniqueness, 12-29
SSMaxNodeIdLength, 12-29
SSMaxSiteIdLength, 12-29
SSMaxSitesMenuItems, 12-30
SSMaxTemplateEvaluationStack, 12-30
SSMigrationCollectionName, 12-30

- <ssm:settings>, 16-1
- SSNavigationBean object, A-11
- SSNavigationNode object, A-11
- ssNodeLink, 9-18, 12-20
- SSOmitFragmentLibrariesInArchiverQueries, 12-30
- SSOnDemandEditorsThresholdCount, 12-31, 13-2
- SSPrefillUrlDirNamesDuringUpgrade, 12-31
- SSProjectAutoCheckinInterval, 12-31
- SSProjectLoadFailureTracingSection, 12-31
- SSProjectReleaseSleepTime, 12-32
- SSProjectReleaseWaitTime, 12-32
- SSQuickDiffDefaultRegionTemplate, 12-32
- ssRandom, 9-12
- ssServerRelativeSiteRoot, 10-2
- SSShowAssignmentTooltips, 12-32
- ssSourceNodeId, 11-19
- ssSplitString, 9-13
- SSSQLUseContains, 12-32
- SSStoppedSiteResponsePageDocName, 12-33
- SSSuppressAddToWebsite, 12-33
- SSSuppressLargeCssOptimization, 12-33
- ssTargetNodeId, 11-19
- SSTempProjectLifetime, 12-34
- SSTitleTagName, 12-34
- SSTrackContentAccess, 12-34, 17-2
- SSTrackerReportNumDaysBack, 17-2
- SSTrackFragmentAccess, 12-34, 17-2
- SSUrlFieldName, 12-35
- SSUrlFixupExceptions, 12-35
- SSUrlPageNames, 12-35
- SSUseAbsoluteRedirects, 12-35
- SSUseCallbackTrackingForASP, 12-36
- SSUseDefaultDocNamePrefix, 12-36
- SSUseDefaultServerRelativeSiteRoot, 12-36
- SSUseDefaultUrlPrefix, 12-36
- SSUseMissingLinkTargetFallback, 12-36
- SSUseOnDemandContributionModeMenus, 12-37
- SSUseUrlSegmentSessionInfo, 12-37
- SSValidateCustomElements, 12-37
- SSWebFilterIgnoreList, 12-38
- ssWeblayoutUrl, 9-19
- SSWeblayoutUrlUsesDocNames, 12-38
- SSWelcomeFile, 12-38
- SSWelcomeFileReplacement, 12-38
- SSXPathCacheEntry, 13-6
- static list
 - simple element in fragment definition, 6-13
- stylesheet
 - xWebsiteObjectType value, 3-2
- subtemplate
 - xWebsiteObjectType value, 3-2
- subtemplates, 2-2
- Switch Content services
 - SS_ADD_WEBSITE_ID, 11-7
 - SS_GET_ALL_NODE_PROPERTIES, 11-12
 - SS_GET_CONFIG_INFO, 11-13
 - SS_GET_CONTRIBUTOR_STRINGS, 11-14
 - SS_GET_DOCUMENT_LABELS, 11-14
 - SS_GET_LINK, 11-16
 - SS_GET_SEARCH_RESULTS, 11-22

- SS_GET_SITE_DEFINITION_FOR_USER, 11-24
- SS_GET_SWITCH_CONTENT_CONFIG, 11-26
- SS_SET_NODE_PROPERTY, 11-30
- SS_SET_NODES_PROPERTIES, 11-30
- SS_SWITCH_REGION_ASSOCIATION, 11-32
- Switch Region Content dialog, 2-5

T

- tags
 - <convert>, 6-10
 - <customgui>, 6-11
 - <designview>, 6-13
 - <element>, 6-13
 - <fragment>, 6-6
 - <fragments>, 6-6
 - <option>, 6-9
 - <parameter>, 6-8
 - <querytext>, 6-9
 - <snippet>, 6-12
 - <validate>, 6-9
- template
 - fragment for, 6-1
 - including a snippet, 6-12
- templates
 - page templates, 2-2
 - region templates, 2-3
 - subtemplates, 2-2
- toolbar
 - command window, 15-2
 - context field, 15-2
 - logging window, 15-2

U

- updating custom elements, A-10
- upgrading legacy sites, A-2
- upgrading pre-7.5 projects
 - automated upgrade, A-2
 - <base>tag, A-7
 - custom elements, A-10
 - custom fragments, A-7
 - folders, A-10
 - full upgrade, A-4
 - GET_SEARCH_RESULTS service, A-7, A-9
 - JSP code, A-11
 - minimal upgrade, A-5
 - multiple content server, A-3
 - search index, A-6
 - single content server, A-3
 - site navigation, A-6
 - site sections, A-10
 - SS_GET_PAGE, A-7
 - upgrading your content servers, A-2
- upgrading pre-7.5 sites, A-2
- Upload Fragment Library, 6-3
- URLs
 - path-based, A-7
- useSecondary parameter, 11-18
- Using JSON for data exchange, 14-1

using server-side script links, 9-18

V

<validate> tag, 6-9

language attribute, 6-10

validation script

xWebsiteObjectType, 3-2

virtual key codes, 5-4

visual debugger, 13-4

W

WCM_PLACEHOLDER, 7-1, 7-2

wcmDynamicConversion, 7-1, 7-2, 9-6

and the properties panel, 7-1

wcmDynamicList, 9-5

wcmElement, 9-3, 9-4, 9-5

wcmFragment, 9-7

wcmIncludeElement, 9-6

wcm.js, 15-3

wcmListElement, 9-4

wcmListEnd, 9-4

wcmListRowCount, 9-5

wcmListStart, 9-4

wcmPlaceholder, 2-2, 2-4, 9-2

wcm.toggle.js, 2-7, 5-4

wcmUrl, 9-7

example, 4-2

web page

for handling errors, 11-20

in Contributor mode, 11-20

temporary preview version of, 11-20

web site

contribution functionality for, 5-4

defining the hierarchy, 6-4

run-time files, 5-1

Web site hierarchy, 9-15

web site hierarchy

sitenavigation.js, 5-1, 5-3

Web Site Section metadata field, A-10

Web sites, A-2

websites folder, 10-1

window.external functionality, 8-5, 8-6, A-10

X

xCollectionID, replaced by xWebsiteSection, 3-3

xDontShowInListsForWebsites, 3-4

Xerces parser, 13-5

Xerces parser, 12-19, 13-5

XML definition

<section> tag, 5-3

<site> tag, 5-3

XML files, auto-generated, 5-1

XML node, extracting, 6-4

XPath expression

as ssIncludeXml() parameter, 6-4

xRegionDefinition, 2-4

xRegionDefinition metadata field, 3-4

xWebsiteID, replaced by xWebsites, 3-3

xWebsiteObjectType, 3-1

xWebsites, 3-3

xWebsiteSection

replaces xCollectionID, 3-3

xWebsiteSection metadata field, A-10