Oracle® Fusion Middleware

Idoc Script Reference Guide

11*g* Release 1 (11.1.1)

E10726-01

May 2010

Oracle Fusion Middleware Idoc Script Reference Guide, 11*g* Release 1 (11.1.1)

E10726-01

Primary Author: Karen Johnson

Contributing Author: Will Harris, Jean Wilson

Contributor: Sebastian Celis, Eva Cordes, Alec Kloss, Rick Petty, Sam White

Contents

3 Master List: Idoc Script by Type

5 Core Functions by Type and Usage

6 Core Content Server Variables and Functions

10 Feature and Product Variables and Functions

Preface

This document describes Idoc Script, a proprietary server-side scripting language.

Audience

This guide is intended for developers who use services to customize software and for application developers who access Content Server functions.

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

Further details about Idoc Script and its use can be found in the following books:

- *Oracle Fusion Middleware Services Reference Guide for Oracle Universal Content Management*
- *Oracle Fusion Middleware Developer's Guide for Content Server*
- *Oracle Fusion Middleware System Administrator's Guide for Content Server*
- *Oracle Fusion Middleware Application Administrator's Guide for Content Server*
- *The Definitive Guide to Content Server Development* by Brian Huff, © 2006, Apress, Berkeley, CA.

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.
italic	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.

1 Introduction

Idoc Script is a proprietary server-side scripting language. This guide describes Idoc Script usage and syntax, and provides detailed descriptions and examples of predefined Idoc Script functions, variables, and configuration settings.

This chapter contains these topics:

- "Document Organization" on page 1-1
- "What's New" on page 1-2

1.1 Document Organization

The information contained in this guide is based on Content Server 11gR1. The information is subject to change as the product technology evolves and as hardware and operating systems are created and modified. Due to the technical nature of browsers, databases, web servers, and operating systems, Oracle cannot warrant compatibility with all versions and features of third-party products.

This guide is divided into the following chapters:

- Chapter 2, "Idoc Script Application" describes Idoc script and how it is used.
- Chapter 3, "Master List: Idoc Script by Type" provides a listing of all variables and functions sorted by type (configuration variable, settable variable, and so on).
- Chapter 4, "Master List: Idoc Script by Usage" provides a listing of all variables and functions sorted by where they are used (in Workflows, or in the Debugger, for example).
- Chapter 5, "Core Functions by Type and Usage" provides a subset of information in Chapter 3, "Master List: Idoc Script by Type" and Chapter 4, "Master List: Idoc Script by Usage", listing only those variables and functions that relate to core Content Server functionality, which include:
  - "Content Items" on page 5-8
  - "Content Server" on page 5-8
  - "Database" on page 5-9
  - "Indexing" on page 5-10
  - "Schema" on page 5-10
  - "Searching" on page 5-11
  - "Security" on page 5-12
  - "Web Server" on page 5-13

  - "Workflow" on page 5-14
- Chapter 6, "Core Content Server Variables and Functions" contains each function and variable used in core Content Server processing with details on usage.
- Chapter 7, "Management and Configuration Options by Type and Usage" lists only those variables and functions used to manage Content Server, including:
  - "Archiver" on page 7-9
  - "Batch Loader" on page 7-9
  - "Clients" on page 7-10
  - "Content Profiles" on page 7-10
  - "Date and Time" on page 7-10
  - "Debugging" on page 7-11
  - "Directories and Paths" on page 7-11
  - "Dynamichtml" on page 7-12
  - "Field Display" on page 7-12
  - "Idoc Script" on page 7-13
  - "JSP" on page 7-14
  - "Localization" on page 7-14
  - "Page Display" on page 7-14
  - "Personalization" on page 7-15
  - "ResultSets" on page 7-15
  - "Strings" on page 7-16
  - "Templates" on page 7-17
  - "Users" on page 7-17
- Chapter 8, "Management and Configuration Variables and Functions" contains each function and variable used to configure and manage Content Server details on their usage.
- Chapter 9, "Feature and Product Variables and Functions by Type and Usage" lists variables and functions related to add-on products to Content Server, including:
  - "Conversion" on page 9-2
  - "Folders and WebDAV" on page 9-5
  - "HTML Preview" on page 9-5
  - "Pop-Up Calendar" on page 9-5
  - "Records Manager" on page 9-5
- Chapter 10, "Feature and Product Variables and Functions" contains only those variables and functions related to add-on products with details on their usage.

An index is provided at the end.

1.2 What's New

The following information has been added or modified for this release:

- The following variables or functions have been added:
  - "ddAppendIndexedColumnResultSet" on page 8-33
  - "ddAppendResultSet" on page 8-35
  - "ddApplyTableSortToResultSet" on page 8-36
  - "ddGetFieldList" on page 8-37
  - "ddIncludePreserveValues" on page 8-38
  - "ddLoadIndexedColumnResultSet" on page 8-40
  - "ddLoadResultSet" on page 8-41
  - "ddMergeIndexedColumnResultSet" on page 8-42
  - "ddMergeResultSet" on page 8-44
  - "ddMergeUsingIndexedKey" on page 8-46
  - "ddSetLocal" on page 8-48
  - "ddSetLocalByColumnsFromFirstRow" on page 8-49
  - "ddSetLocalByColumnsFromFirstRowIndexed" on page 8-50
  - "ddSetLocalEmpty" on page 8-51
  - "ddSetLocalEmptyByColumns" on page 8-52
  - "idocTestForInclude" on page 8-147
  - "rs" on page 8-245
  - "rsAddFieldsWithDefaults" on page 8-247
  - "rsCreateReference" on page 8-254
  - "rsLoopInclude" on page 8-265
  - "rsLoopSingleRowInclude" on page 8-266
- The following variables or functions have been modified:
  - "ServiceStartupTimeout" on page 6-238
  - "IdcHomeDir" on page 8-143
  - "IdcNativeDir" on page 10-48
  - "IdcResourcesDir" on page 10-49
  - "ShowContentForStorageBrowse" on page 10-119
- The following variables or functions have been removed:
  - AcadUseLISPInterface
  - AccountMapPrefix
  - AcctPermDelim
  - AcctPrefix
  - AdjustPrinterMargins
  - AdsSimpleAuth
  - AdsUserPassword
  - AllowIntranetUsers

- AllowPassThru
- AttributeMap
- AutoCad2000PlotterFilePath
- CGI_DEBUG
- CGI_RECEIVE_DUMP
- CGI_SEND_DUMP
- CgiFileName
- CLASSPATH
- ClusterBinDirRule
- ClusterNodeAddress,
- ClusterNodeName
- ComputerName
- ConnectionName
- CreatePDFThumbnails
- CustomConversionWaitTime
- CustomConverterPath
- DebugMode
- DebugStdConversion
- DefaultAuth
- DefaultMasterDomain
- DefaultNetworkAccounts
- DisableAmberLayouts
- DisableClassicLayoutPublishing
- DisableCollegiateSkinPublishing
- DisableStellent05SkinPublishing
- DisableStellentSkinPublishing
- DisableWindowsPublishing
- DisplayPSOrigin
- DistillerNormJobSetting
- DistillerOptJobSetting
- DocConverterEngineDir
- EnableCopyConvertCopy
- ExcludeConversionFilter
- ExclusiveIncludeConversionFilter
- FILTER_DEBUG
- IdcAdminServerPort
- ImageAlchemyExePath

- IncludeXmlFlexionDocType
- IndexerPath
- IndexVaultFile
- IsJdbc
- IsThumbnailPresent
- LocalGroupServer
- LogWithConnectionName
- MaxNumberConvertedHistoryShow
- MaxNumRecursiveStepDefinitions
- NetworkAdminGroup
- OptimizePDF
- PdfVersion
- PLUGIN_DEBUG
- ProfileCacheTimeoutInMins
- RolePrefix
- SetPdfVersion
- ThumbnailDPI
- ThumbnailJPEGQuality
- ThumbnailSizeLimit
- ThumbnailSizingMethod
- ThumbnailTemplate
- UseAdobeDistiller
- UseAlchemy111
- UseAutoCad2000
- UseAutocadModelSpace
- UseFullGroupName
- UseGroupFilter
- UseLightweightToPostscriptStep
- UseLocalGroups
- UseSearchML
- UseShortNamesAlways
- UseXmlEmbeddings
- UseXmlExtensibleDTD
- UseXmlNoDTD
- UseXmlNoReference
- UseXmlPStyleNames
- UseXmlSchema

  - UseXmlSchemaDTD
  - XmlTimeout
- Collaboration Manager is not supported. Core workflow variables are still supported.
- References to *install_dir* have been changed in support of Content Sever deployment and configuration on Oracle WebLogic Server to one of the following:
  - *UCM_ORACLE_HOME*: Refers to the *ORACLE_HOME* and Oracle WebLogic Server install directories for an Oracle UCM instance. Default directories are provided, however, sites can specify alternative directories.
  - *IdcHomeDir*: Refers to the directory in *UCM_ORACLE_HOME* where the Oracle UCM (`ucm`) server media is located. The server media can run Content Server, Inbound Refinery, or Universal Records Management.
  - *DomainHome*: Refers to the user-specified directory where an Oracle UCM server is deployed to run on an Oracle WebLogic Server application server. The *DomainHome*/ucm/*short-product-id*/bin directory contains the intradoc.cfg file and executables.
  - *short-product-id*: Refers to the type of Oracle UCM server deployed on an Oracle WebLogic Server, domain, used as the context root (default HttpRelativeWebRoot configuration value). Possible values include:
    - `cs` (Content Server)
    - `ibr` (Inbound Refinery)
    - `urm` (Universal Records Management)
  - *IntradocDir*: Refers to the root directory for configuration and data files specific to a Content Server instance deployed on an Oracle UCM domain on an Oracle WebLogic Server. This Idoc Script variable is configured for one type of Content Server instance: Content Server, Inbound Refinery, or Universal Records Management. This directory can be located elsewhere, but the default location is *DomainHome*/ucm/*short-product-id*/.

2 Idoc Script Application

Idoc Script is the server-side custom scripting language for Content Server. It enables you to reference variables, conditionally include content in HTML pages, and loop over results returned from queries. Because Idoc Script is evaluated on the server side rather than the client side, page elements are processed after the browser has made a request, but before the requested page is returned to the client.

Idoc Script is used primarily for the presentation of HTML templates and configuration settings.

This chapter includes the following sections:

2.1 Idoc Naming Conventions

Idoc variables (sometimes called *configuration variables* or *environment variables*) can be used in Idoc scripts and in configuration files.

In general, if the variable is part of a configuration, it begins with a capital letter. Those variables specified in the config.cfg file or intradoc.cfg file usually have an initial capital letter. Many parameters in service requests also begin with upper case letters.

Variables defined on a page, such as those that are derived and then used in files such as std_page.htm, begin with a lowercase letter. See "executeService" on page 8-73 for an example. The variables are calculated from environment variables or service variables then used for presentation.

If a variable is used to define an object, it begins with lowercase letters designating the type of object it is defining. See "wfSet" on page 6-301 for an example of a workflow-specific variable. In addition, all functions start with a lowercase letter and many start with a prefix to describe the type of function. For example, most string manipulation functions begin with `str`, or ResultSet functions begin with `rs`.

All database column names that are not custom metadata fields begin with the lower-case letter d. All custom metadata database column names create by the Content Server begin with the lower-case letter x.

2.2 Idoc Script Syntax

Idoc Script follows these basic syntax rules:

- Idoc Script Tags
- Idoc Script Comments

2.2.1 Idoc Script Tags

Idoc Script commands begin with <$ and end with $> delimiters. For example:

```
<$dDocTitle$>
<$if UseGuiWinLook and isTrue(UseGuiWinLook)$>
```

If you are using Idoc Script in an HCSP or HCSF page, you must use the syntax <!--$script--> for Idoc Script tags. For more information, see the *Oracle Fusion Middleware Developer's Guide for Content Server*.

2.2.2 Idoc Script Comments

You can use standard HTML comments or Idoc Script comments in Idoc Script code. An Idoc Script comment begins with [[% and closes with %]]. For example:

```
<!-- HTML Comment -->
[[%My Comment%]]
```

The comment syntax also can be used to comment out dynamichtml constructs and other resource specifiers, such as string resources, in Idoc Script resource files. For example:

```
[[% Commenting out resource includes
<@dynamichtml myinclude@>
<@end@>
End comment %]]
```

2.3 Idoc Script Uses

There are six basic uses for Idoc Script:

- Includes enable you to reuse pieces of Idoc Script and HTML code.
- Variables enable you to define and substitute variable values.
- Functions enable you to perform actions, including string comparison and manipulation routines, date formatting, and ResultSet manipulation.
- Conditionals enable you to evaluate *if* and *else* clauses to include or exclude code from an assembled page.
- Looping enables you to repeat code for each row in a ResultSet that is returned from a query.
- The Administration Interface enables you to use Idoc Script in Content Server applets and customizations.

2.3.1 Includes

An *include* defines pieces of code that are used to build the Content Server web pages. Includes are defined once in a resource file and then referenced by as many template files as necessary. The system leverages includes very heavily.

Includes make it easier for you to customize your instance using component architecture and dynamic server pages. For more information on includes and customization, see the *Oracle Fusion Middleware Developer's Guide for Content Server*.

- An include is defined in an HTM resource file using the following format:

  ```
  <@dynamichtml name@>
      code
  <@end@>
  ```

- An include is called from an HTM template file using the following Idoc Script format:

  ```
  <$include name$>
  ```

- Includes can contain Idoc Script and valid HTML code, including JavaScript, Java applets, cascading style sheets, and comments.
- Includes can be defined in the same file as they are called from, or they can be defined in a separate file.
- Standard includes are defined in the *ucm_home_dir*/idc/resources/core/idoc/std_page.idoc file.
- HDA and CFG files are not script enabled, therefore using an include statement in either of these types of files is not supported.

See the following sections for more information:

- "Include Example" on page 2-3
- "Super Tag" on page 2-4
- "Super Tag Example" on page 2-4
- "Special Keywords" on page 2-11
- "Keywords Versus Functions" on page 2-12

2.3.1.1 Include Example

One of the most common includes is the body definition element *<@dynamichtml body_def>*. This include sets the page background color, the color of hyperlinks, and the background image. The following is an example of code located in the *ucm_home_dir*/idc/resources/core/idoc/std_page.idoc file:

```
<@dynamichtml body_def@>
<!--Background image defined as part of body tag--->
<body
    <$if background_image$>
        background="<$HttpImagesRoot$><$background_image$>"
    <$elseif colorBackground$>
        bgcolor="<$colorBackground$>"
    <$endif$>
    <$if xpedioLook$>
        link="#663399" vlink="#CC9900"
    <$else$>
        link="#000000" vlink="#CE9A63" alink="#9C3000"
    <$endif$>
```

```
    marginwidth="0" marginheight="0" topmargin="0" leftmargin="0"
>
<@end@>
```

Most of the standard template resource files (for example, *ucm_home_dir*/idc/resources/core/templates/pne_home_page.htm) contain the following Idoc Script code near the top of the page:

```
<$include body_def$>
```

When the Content Server resolves a template page containing this code, it looks for the `<@dynamichtml body_def@>` definition and replaces the placeholder code with the code in the definition.

2.3.1.2 Super Tag

The *super* tag is used to define exceptions to an existing include. The *super* tag tells the include to start with an existing include and then add to it or modify using the specified code.

- The *super* tag uses the following syntax:

  ```
  <@dynamichtml my_resource@>
      <$include super.my_resource$>
      exception code
  <@end@>
  ```

- You can use the *super* tag to refer to a standard include or a custom include. The *super* tag incorporates the include that was loaded last.
- You can specify multiple *super* tags to call an include that was loaded earlier than the last version. For example, to make an exception to the standard *body_def* include in two different components, you can use the following syntax in the one that is loaded last:

  ```
  <$include super.super.body_def$>
  ```

  > **Caution:** If you use multiple *super* tags in one include, ensure that you know where the resources are loaded from and the order they are loaded in.

- The *super* tag is particularly useful when making small customizations to large includes or when you customize standard code that is likely to change from one software version to the next. When you upgrade to a new version of Content Server, the *super* tag ensures that your components are using the most recent version of the include, modifying only the specific code you require to customize your instance.

2.3.1.3 Super Tag Example

In this example, a component defines the *my_resource* include as follows:

```
<@dynamichtml my_resource@>
    <$a = 1, b = 2$>
<@end@>
```

Another component that is loaded later enhances the *my_resource* include using the *super* tag. The result of the following enhancement is that `a` is assigned the value 1 and `b` is assigned the value 3:

```
<@dynamichtml my_resource@>
    <$include super.my_resource$>
    <!--Change "b" but not "a" -->
    <$b = 3$>
<@end@>
```

2.3.2 Variables

A *variable* enables you to define and substitute variable values.

The following sections describe how to work with Idoc Script variables:

- "Creating a Variable" on page 2-5
- "Referencing a Variable" on page 2-5
- "Assigning a Value" on page 2-5
- "Using Commas as Separators" on page 2-6
- "Referencing a Variable in a Conditional" on page 2-6
- "Variable Substitution Order" on page 2-6
- "Regular Variables" on page 2-6

2.3.2.1 Creating a Variable

Idoc Script variables are created in one of the following ways:

- Many variables are predefined.
- You can define your own custom variables.
- Some variable values must be generated using queries and services. Some variable information is not automatically available from the database, so it must be asked for by defining a query and service.

See Chapter 3, "Master List: Idoc Script by Type" for more information on the types of Idoc Script variables.

2.3.2.2 Referencing a Variable

You can reference a variable in templates and other resource files with the following Idoc Script tag:

```
<$variable_name$>
```

2.3.2.3 Assigning a Value

- A value can be assigned to a variable using the structure:

  ```
  <$variable=value$>
  ```

  For example, `<$i=0$>` assigns the value of *0* to the variable *i*.

- Variable values can also be defined in an environment resource (CFG) file using the following name/value pair format:

  ```
  variable=value
  ```

  For example, standard configuration variables are defined in the *IntradocDir*/config/config.cfg file.

2.3.2.4 Using Commas as Separators

Idoc Script supports multiple clauses separated by commas in one script block.

For example, you can use `<$a=1,b=2$>` rather than two separate statements: `<$a=1$>` and `<$b=2$>`.

2.3.2.5 Referencing a Variable in a Conditional

The following structure can be used to evaluate the existence of a variable:

```
<$if variable_name$>
```

If the variable is defined, this conditional is evaluated as TRUE. If the variable is not defined or it is defined as an empty (null) string, it is evaluated as FALSE.

See "Conditional Example" on page 2-8 for an example of how this type of reference is typically used.

2.3.2.6 Variable Substitution Order

When a variable value is required to fulfill a service request, the data cached in the DataBinder is evaluated in the following default order:

1. LocalData
2. Active ResultSets
3. Non-active ResultSets
4. Environment

For more information on the DataBinder, see the *Oracle Fusion Middleware Developer's Guide for Content Server*.

2.3.2.7 Regular Variables

A regular variable that does not have special evaluation logic (such as Conditional Dynamic Variables) is equivalent to using the `#active` keyword prefix.

For example, the tag `<$variable$>` is equivalent to `<$#active.variable$>`. However, if *#active* is not explicitly stated and the variable is not found, an error report is printed to the debug output.

2.3.3 Functions

Idoc Script has many built-in global functions. Functions perform actions, including string comparison and manipulation routines, date formatting, and ResultSet manipulation. Some functions also return results, such as the results of calculations or comparisons.

Information is passed to functions by enclosing the information in parentheses after the name of the function. Pieces of information that are passed to a function are called *parameters*. Some functions do not take parameters; some functions take one parameter; some take several. There are also functions for which the number of parameters depends on how the function is being used.

See "Global Functions" on page 3-11 for a list of Idoc Script functions.

2.3.3.1 Personalization Functions

Personalization functions refer to user properties that are defined in personalization files, also called *user topic files*. Each user's User Profile settings, personal links in the left navigation bar, and workflow in queue information are all defined in user topic files, which are HDA files located in the *ucm_home_dir*/data/users/profiles/*us*/*username*/ directories.

The following global functions reference user topic files:

- "utGetValue" on page 8-353
- "utLoad" on page 8-354
- "utLoadResultSet" on page 8-356

For example, the Portal Design link in a user's left navigation bar is generated from the following code in the *pne_nav_userprofile_links* include (located in the *ucm_home_dir*/shared/config/resources/std_page.htm resource file). If the *portalDesignLink* property in the *ucm_home_dir*/data/users/profiles/*us*/*username*/pne_portal.hda file is TRUE, the link is displayed:

```
<$if utGetValue("pne_portal", "portalDesignLink") == 1$>
    <$hasUserProfileLinks=1$>
    <tr>
    <td colspan=2 nowrap align="left">
        <a class=pneLink href="<$HttpCgiPath$>?IdcService=GET_PORTAL_
PAGE&Action=GetTemplatePage&Page=PNE_PORTAL_DESIGN_PAGE">
        <$lc("wwPortalDesign")$></a>
    <td>
    </tr>
<$endif$>
```

2.3.4 Conditionals

A *conditional* enables you to use *if* and *else* clauses to include or exclude code from an assembled page.

- Use the following Idoc Script keywords to evaluate conditions:
  - `<$if condition$>`
  - `<$else$>`
  - `<$elseif condition$>`
  - `<$endif$>`
- Conditional clauses use this general structure:

  ```
  <$if conditionA$>
      <!--Code if conditionA is true-->
  <$elseif conditionB$>
      <!--Code if conditionB is true-->
  <$else$>
      <!--Code if neither conditionA nor conditionB are true-->
  <$endif$>
  ```

- A *condition* expression can be any Idoc Script function or variable.

  See "Referencing a Variable in a Conditional" on page 2-6 for more information.
- Boolean Operators can be used to combine conditional clauses. For example, you can use the *and* operator as follows:

  ```
  <$if UseBellevueLook and isTrue(UseBellevueLook)$>
  ```

- If the *condition* expression is the name of a ResultSet available for inclusion in the HTML page, the conditional clause returns *true* if the ResultSet has at least one row. This ensures that a template page presents information for a ResultSet only if there are rows in the ResultSet.

- A conditional clause that does not trigger special computation is evaluated using the XXXXXXXXXXXX_cannot_cross-reference to a marker on a para in a bable_ XXXXXXXXXXXXXX prefix. The result is *true* if the value is not null and is either a nonempty string or a nonzero integer.

See "Conditional Example" on page 2-8 for an example of conditional code.

2.3.4.1 Conditional Example

In this example, a table cell `<td>` is defined depending on the value of the variable *xDepartment*:

```
<$if xDepartment$>
    <td><$xDepartment$></td>
<$else$>
    <td>Department is not defined.</td>
<$endif$>
<$xDepartment=""$>
```

- If the value of *xDepartment* is defined, then the table cell contains the value of *xDepartment*.
- If the value of *xDepartment* is not defined or is an empty (null) string, a message is written as the content of the table cell.
- The last line of code clears the *xDepartment* variable by resetting it to an empty string.

2.3.5 Looping

Loop structures allow you to execute the same code a variable number of times. Looping can be accomplished in two ways with Idoc Script:

- ResultSet Looping
- While Looping

See "Ending a Loop" on page 2-10 for information on exiting and ending a loop structure.

2.3.5.1 ResultSet Looping

ResultSet looping repeats a set of code for each row in a ResultSet that is returned from a query. The name of the ResultSet to be looped is specified as a variable using the following syntax:

```
<$loop ResultSet_name$>
    code
<$endloop$>
```

- The code between the `<$loop$>` and `<$endloop$>` tags is repeated once for each row in the ResultSet.
- When inside a ResultSet loop, you can retrieve values from the ResultSet using the getValue function. Substitution of values depends on which row is currently being accessed in the loop.
- When inside a ResultSet loop, that ResultSet becomes active and has priority over other ResultSets when evaluating variables and conditional statements.

- You cannot use the `<$loop$>` tag to loop over a variable that points to a ResultSet. Instead you must loop over the ResultSet manually using the rsFirst and rsNext functions.

  For example, you cannot use the following code to loop over a ResultSet:

  ```
  <$name="SearchResults"$>
  <$loop name$>
      <!--output code-->
  <$endloop$>
  ```

  Instead, you must use the following code:

  ```
  <$name="SearchResults"$>
  <$rsFirst(name)$>
  <$loopwhile getValue(name, "#isRowPresent")$>
      <!--output code-->
      <$rsNext(name)$>
  <$endloop$>
  ```

2.3.5.2 ResultSet Looping Example

In this example, a search results table is created by looping over the *SearchResults* ResultSet, which was generated by the GET_SEARCH_RESULTS service.

```
<$QueryText="dDocType <matches> 'ADACCT'"$>
<$executeService("GET_SEARCH_RESULTS")$>
<table>
    <tr>
        <td>Title</td><td>Author</td>
    </tr>
<$loop SearchResults$>
    <tr>
        <td><a href="<$SearchResults.URL$>"><$SearchResults.dDocTitle$></a></td>
        <td><$SearchResults.dDocAuthor$></td>
    </tr>
<$endloop$>
</table>
```

2.3.5.3 While Looping

While looping enables you to create a conditional loop. The syntax for a while loop is:

```
<$loopwhile condition$>
    code
<$endloop$>
```

- If the result of the *condition* expression is *true*, the code between the `<$loopwhile$>` and `<$endloop$>` tags is executed.
- After all of the code in the loop has been executed, control returns to the top of the loop, where the *condition* expression is evaluated again.
  - If the result is *true*, the code is executed again.
  - If the code if the result is *false*, the loop is exited.

2.3.5.4 While Looping Example

In this example, a variable named *abc* is increased by 2 during each pass through the loop. On the sixth pass (when *abc* equals 10), the condition expression is no longer true, so the loop is exited.

```
<$abc=0$>
<$loopwhile abc<10$>
    <$abc=(abc+2)$>
<$endloop$>
```

2.3.5.5 Ending a Loop

There are two Idoc Script tags that will terminate a ResultSet loop or while loop:

- `<$endloop$>` returns control to the beginning of the loop for the next pass. All loops must be closed with an `<$endloop$>` tag.
- `<$break$>` causes the innermost loop to be exited. Control resumes with the first statement following the end of the loop.

2.3.6 Administration Interface

You can use Idoc Script in several areas of the administration interface, including:

- Workflow Admin
- Web Layout Editor
- Batch Loader
- Archiver
- System Properties
- E-mail

2.3.6.1 Workflow Admin

In the Workflow Admin tool, you can use Idoc Script to define the following:

- step events
- jump messages
- extra exit conditions
- tokens
- custom effects for jumps

For example, the following step entry script sends documents in the *Secure* security group to the next step in the workflow:

```
<$if dSecurityGroup like "Secure"$>
    <$wfSet("wfJumpName", "New")$>
    <$wfSet("wfJumpTargetStep", wfCurrentStep(1))$>
    <$wfSet("wfJumpEntryNotifyOff", "0")$>
<$endif$>
```

See "Workflows" on page 3-20 for more information.

2.3.6.2 Web Layout Editor

In the Web Layout Editor, you can use Idoc Script in the page titles, page descriptions, URL descriptions, query result pages, and content queries.

For example:

- You can use Idoc Script tags in the query results page definition to specify the contents of each row in a search results table.

- To set the search results to return all content items up to 7 days, you could define the search query to be:

  ```
  dInDate > '<$dateCurrent(-7)$>'
  ```

- To define a report that returns results based on the current user, you could define `User Name is <$UserName$>` as part of the report query expression.

See the *Oracle Fusion Middleware System Administrator's Guide for Content Server* for more information.

2.3.6.3 Batch Loader

In the Batch Loader, you can use Idoc Script in a mapping file, which tells the BatchBuilder utility how to determine the metadata for file records. See the *Oracle Fusion Middleware System Administrator's Guide for Content Server* for more information.

2.3.6.4 Archiver

In Archiver, you can use Idoc Script in the following areas:

- **Export query values.** For example, to archive content more than one year old, you could use `<$dateCurrent(-365)$>` as the Release Date value.
- **Value map output values.** For example, to set the expiration date one week in the future for all imported revisions, you could use `<dateCurrent(7)$>` as the Output Value.

See the *Oracle Fusion Middleware System Administrator's Guide for Content Server* for more information.

2.3.6.5 System Properties

When you set values in the System Properties utility, in the Admin Server, or in the Local Configuration or Shared Configuration screens of the Inbound Refinery, you are actually setting Idoc Script configuration variables. See the documentation for these applications for more information.

2.3.6.6 E-mail

You can use Idoc Script to customize the subject line and body of a notification e-mail. For example:

```
Hello, <$UserName$>. Content item <$dDocName$> requires your review.
```

2.4 Special Keywords

The following keywords have special meaning in Idoc Script:

Keyword	Example	Description
#active	<$#active.*variable*$>	Retrieves the value of the specified variable from the DataBinder, searching in the following default order: **1.** Active ResultSets **2.** Local data **3.** All other ResultSets **4.** Environment Does not send an error report to the debug output if the variable is not found.
#local	<$#local.*variable*$>	Retrieves the value of the specified variable from the local data. Does not send an error report to the debug output if the variable is not found.
#env	<$#env.*variable*$>	Retrieves the value of the specified variable from the environment settings. Does not send an error report to the debug output if the variable is not found.
exec	<$exec *expression*$>	Executes an expression and suppresses the output (does not display the expression on the page). In earlier versions of Idoc Script, the *exec* keyword was required to suppress the *value* of any variable from appearing in the output file. In the current version, the *exec* keyword is needed only to suppress an *expression* from appearing in the output file.
include	<$include *ResourceName*$>	Includes the code from the specified resource. See "Includes" on page 2-3 for more information.
super	<$include super.<*include*>$>	Starts with the existing version of the include code. See "Super Tag" on page 2-4 for more information.

2.4.1 Keywords Versus Functions

Content Server pages use the *include* and *exec* Special Keywords and the *inc* and *eval* Functions extensively. This section describes the differences between these commands and gives examples of how to use them.

The *include* and *exec* keywords are standalone commands that operate on defined parameters, but cannot take a variable as a parameter. The *inc* and *eval* functions have similar purposes, but they can take variables for parameters, which enables you to dynamically create Idoc Script code depending on the value of the variables.

The following sections describe these keywords and functions in detail:

- "exec Keyword" on page 2-12
- "eval Function" on page 2-13
- "include Keyword" on page 2-13
- "inc Function" on page 2-13

2.4.1.1 exec Keyword

The *exec* keyword executes an Idoc Script expression and suppresses the output (does not display the expression on the page). It is primarily used to set variables without writing anything to the page.

In earlier versions of Idoc Script, the *exec* keyword was required to suppress the *value* of any variable from appearing in the output file. In the current version, the *exec* keyword is needed only to suppress an *expression* from appearing in the output.

For example, the first line below is equivalent to the last two lines:

```
<$varA="stringA", varB ="stringB"$>
<$exec varA="stringA"$>
<$exec varB="stringB"$>
```

See exec for more information.

2.4.1.2 eval Function

The *eval* function evaluates an expression as if it were actual Idoc Script.

In the following example, a variable named *one* is assigned the string *Company Name*, and a variable named *two* is assigned a string that includes variable *one*.

```
<$one="Company Name"$>
<$two="Welcome to <$one$>"$>
<$one$><br>
<$two$><br>
<$eval(two)$>
```

In the page output, variable *one* presents the string `Company Name`, variable *two* presents the string `Welcome to <$one$>`, and the function *eval(two)* presents the string `Welcome to Company Name`.

Note that the string to be evaluated must have the Idoc Script delimiters <$ $> around it, otherwise it will not be evaluated as Idoc Script.

Also note that too much content generated dynamically in this manner can slow down page display. If the *eval* function is used frequently on a page, it may be more efficient to put the code in an include and use the inc Function in conjunction with the *eval* function.

See eval for more information.

2.4.1.3 include Keyword

The *include* keyword is the standard way in which chunks of code are incorporated into the current page. Because *include* is a keyword, it cannot take a variable as a parameter-the parameter must be the name of an include that already exists.

See "Includes" on page 2-3 and include for more information.

2.4.1.4 inc Function

The *inc* function does the same thing as the *include* keyword, except that it can take a variable as the parameter. This function is most useful for dynamically changing which include will be used depending on the current value of a variable.

For example, say you want to execute some Idoc Script for some, but not all, of your custom metadata fields. You could dynamically create includes based on the field names (such as *specific_include_xComments*) by executing this Idoc Script:

```
<$loop DocMetaDefinition$>
    <$myInclude = "specific_include_" & dName$>
    <$exec inc(myInclude)$>
<$endloop$>
```

Note the use of the exec Keyword, which suppresses the output of the include specified by the *inc* function. If you don't use *exec* before the *inc* function, the HTML inside the specified include will be displayed on the page.

Note that if the *specific_include_xComments* does not exist, this code will not throw an error because the output is not being displayed.

See inc for more information.

2.5 Operators

Idoc Script supports several operators. This section describes the following:

- "Comparison Operators" on page 2-14
- "Special String Operators" on page 2-14
- "Numeric Operators" on page 2-15
- "Boolean Operators" on page 2-16

2.5.1 Comparison Operators

Use the following comparison operators compare the value of two operands and return a *true* or *false* value based on the result of the comparison. These operators can be used to compare integers and Boolean values in Idoc Script.

If you are using Idoc Script in an HCSP or HCSF page, you must use special comparison operators. For more information, see the *Oracle Fusion Middleware Developer's Guide for Content Server*.

Operator	Description	Example
==	equality	`<$if 2 == 3$>` evaluates to false
!=	inequality	`<$if 2 != 3$>` evaluates to true
<	less than	`<$if 2 < 2$>` evaluates to false
<=	less than or equal	`<$if 2 <= 2$>` evaluates to true
>	greater than	`<$if 3 > 2$>` evaluates to true
>=	greater than or equal	`<$if 3 >= 2$>` evaluates to true

These are numeric operators that are useful with strings only in special cases where the string data has some valid numeric meaning, such as dates (which convert to milliseconds when used with the standard comparison operators).

- For string concatenation, string inclusion, and simple string comparison, use the Special String Operators.
- To perform advanced string operations, use strEquals, strReplace, or other string-related global functions.

2.5.2 Special String Operators

Use the following special string operators to concatenate and compare strings:

Operator	Description	Example

&	The _string join operator_ performs string concatenation. Use this operator to create script that produces Idoc Script for a resource include.	`<$"<$include " & VariableInclude & "$>"$>`
		evaluates to:
		`<$include VariableName$>`

like	The _string comparison operator_ compares two strings.
	▪ The first string is compared against the pattern of the second string. (The second string can use asterisk and question mark characters as Wildcards.)
	▪ This operator is not case sensitive.
		▪ Evaluates to FALSE:
		`<$if "cart" like "car"$>`
		▪ Evaluates to TRUE:
		`<$if "cart" like "car?"$>`
		▪ Evaluates to TRUE:
		`<$if "carton" like "car*"$>`
		▪ Evaluates to TRUE:
		`<$if "Carton" like "car*"$>`

|	The _string inclusion operator_ separates multiple options, performing a logical OR function.	Evaluates to TRUE: `<$if "car" like "car\|truck\|van"$>`

For example, to determine whether the variable a has the prefix _car_ or contains the substring _truck_, this expression could be used:

```
<$if a like "car*|*truck*"$>
```

Important: To perform advanced string operations, use strEquals, strReplace, or other string-related global functions. See "Strings" on page 4-19 for a list.

2.5.2.1 Wildcards

The _like_ operator recognizes the following wildcard symbols:

Wildcard	Description	Example
*	Matches 0 or more characters.	▪ _grow*_ matches _grow_, _grows_, _growth_, and _growing_ ▪ _*car_ matches _car_, _scar_, and _motorcar_ ▪ _s*o_ matches _so_, _solo_, and _soprano_
?	Matches exactly one character.	▪ _grow?_ matches _grows_ and _growl_ but not _growth_ ▪ _grow??_ matches _growth_ but not _grows_ or _growing_ ▪ _b?d_ matches _bad_, _bed_, _bid_, and _bud_

2.5.3 Numeric Operators

Use the following numeric operators to perform arithmetic operations. These operators are for use on integers evaluating to integers or on floats evaluating to floats:

Operator	Description	Example
+	Addition operator.	`<$a=(b+2)$>`

Operator	Description	Example
-	Subtraction operator.	`<$a=(b-2)$>`
*	Multiplication operator.	`<$a=(b*2)$>`
/	Division operator.	`<$a=(b/2)$>`
%	Modulus operator. Provides the remainder of two values divided into each other.	`<$a=(b%2)$>`

2.5.4 Boolean Operators

Use the following Boolean operators to perform logical evaluations:

Operator	Description	Example
and	■ If both operands have nonzero values or are true, the result is 1. ■ If either operand equals 0 or is false, the result is 0.	`<$if 3>2 and 4>3$>` evaluates to 1
or	■ If either operand has a nonzero value or is true, the result is 1. ■ If both operands equal 0 or are false, the result is 0.	`<$if 3>2 or 3>4$>` evaluates to 1
not	■ If the operand equals 0 or is false, the result is 1. ■ If the operand has a nonzero value or is true, the result is 0.	`<$if not 3=4$>` evaluates to 1

Boolean operators evaluate from left to right. If the value of the first operand is sufficient to determine the result of the operation, the second operand is not evaluated.

2.6 Metadata Fields

This section includes these topics:

- "Metadata Field Naming" on page 2-16
- "Standard Metadata Fields" on page 2-17
- "Option Lists" on page 2-20
- "Referencing Metadata in Dynamic Server Pages" on page 2-22

2.6.1 Metadata Field Naming

Each metadata field has an internal field name, which is used in code. In addition, many fields have descriptive captions which are shown on web pages.

- Use field captions when displaying metadata to the user.
- Use internal field names when batch loading files or scripting dynamic server pages (.hcst,.hcsp, and .hcsf pages).
- All internal metadata field names begin with either a d or an x:
  - *Predefined* field names begin with a d. For example, `dDocAuthor`.
  - *Custom* field names begin with an x. For example, `xDepartment`.

- When you create a custom metadata field in the Configuration Manager, the x is automatically added to the beginning of your field name.

Important: In all cases, internal metadata field names are case sensitive.

2.6.2 Standard Metadata Fields

This section describes the standard metadata fields that Content Server stores for each content item. The fields are grouped as follows:

- Common Metadata Fields
- Other Fields

2.6.2.1 Common Metadata Fields

The following metadata fields are the most commonly used in customizing the interface. These fields appear by default on checkin and search pages.

Do not confuse the Content ID (dDocName) with the dID. The dID is an internally generated integer that refers to a specific revision of a content item.

Internal Field Name	Standard Field Caption	Description
dDocAccount	Account	Security account.
dDocAuthor	Author	User who checked in the revision.
xComments	Comments	Explanatory comments.
dDocName	Content ID	Unique content item identifier.
dOutDate	Expiration Date	Date the revision becomes unavailable for searching or viewing.
dInDate	Release Date	Date the revision is scheduled to become available for searching and viewing (see also dCreateDate and dReleaseDate).
dRevLabel	Revision	Revision label (see also dRevisionID).
dSecurityGroup	Security Group	Security group.
dDocTitle	Title	Descriptive title.
dDocType	Type	Content type.

2.6.2.2 Other Fields

In addition to the Common Metadata Fields, the following metadata is stored for content items:

Internal Field Name	Standard Field Caption	Description
dCheckoutUser	Checked Out By (Content Information page)	User who checked out the revision.
dCreateDate	None	Date the revision was checked in.
dDocFormats	Formats (Content Information page)	File formats of the primary and alternate files.

Internal Field Name	Standard Field Caption	Description
dDocID	None	Unique rendition identifier.
dExtension	None	File extension of the primary file.
dFileSize	None	File size of the primary file (in kilobytes).
dFlag1	None	Not used.
dFormat	Format (checkin page, *Allow override format on checkin* enabled)	File format of the primary file.
dID	None	Unique revision identifier.
dIndexerState	None	State of the revision in an Indexer cycle. Possible values are: **X**: The revision has been processed by the rebuild cycle. **Y**: The revision has been processed by the rebuild cycle. **A, B, C, or D** : Values generated at run time that can be assigned to any of the following states: ■ Loading the revision for the active update cycle. ■ Indexing the revision for the active update cycle. ■ Loading the revision for the rebuild cycle. ■ Indexing the revision for the rebuild cycle. The specific definitions of these values are stored in the *DomainHome*/ucm/*cs*/search/cyclesymbols.hda file.
dIsCheckedOut	None	Indicates whether the revision is checked out. **0**: Not checked out **1**: Checked out
dIsPrimary	None	Indicates the type of file, primary or alternate. **0**: Alternate file **1**: Primary file
dIsWebFormat	None	Indicates whether the file is the web-viewable file in the *weblayout* directory. **0**: Not web-viewable file **1**: Web-viewable file
dLocation	None	Not used.
dMessage	None (Content Information page)	Indicates the success or reason for failure for indexing or conversion.
dOriginalName	Get Native File (Content Information page) Original File (revision checkin page)	Original file name of the native file.

Internal Field Name	Standard Field Caption	Description
dProcessingState	None	Conversion status of the revision. Possible values are: **I**: Incomplete Conversion; an error occurred in the conversion after a valid web-viewable file was produced and the file was full-text indexed. **Y**: Converted; the revision was converted successfully and the web-viewable file is available. **P**: Refinery PassThru; Inbound Refinery failed to convert the revision and passed the native file through to the web. **F**: Failed; the revision is deleted, locked, or corrupted, or an indexing error occurred. **C** : Processing; the revision is being converted by the Inbound Refinery. **M** : MetaData Only; full-text indexing was bypassed and only the revision's metadata was indexed.
dPublishState	None	Publish state of a revision that is used with Content Publisher. Possible values are: **P**: Published **S**: Staged **W**: Workflow **null**: Not staged, published, or in a workflow
dPublishType	None	Content type for a revision that is used with Content Publisher. Possible values are: **H**: Home **N**: Navigation **S**: Query results pages **P**: Pages **G**: Gallery Graphics **C**: Contributor embedded graphics **O**: All others
dReleaseDate	None	Date that the revision was actually released.
dReleaseState	None	Release status of a revision. **N**: New, not yet indexed **E**: In a workflow **R**: Processing, preparing for indexing **I**: Currently being indexed; the file has been renamed to the released name **Y**: Released **U**: Released, but needs to be updated in the index **O**: Old revision

Internal Field Name	Standard Field Caption	Description
dRendition1	None	Indicates whether the file is a thumbnail rendition. Possible values are: **null**: File is not a thumbnail rendition **T**: File is a thumbnail rendition
dRendition2	None	Not used.
dRevClassID	None	Internal integer that corresponds to the Content ID (dDocName). Used to enhance query response times.
dRevisionID	None	Internal revision number that increments by 1 for each revision of a content item, regardless of the value of dRevLabel.
dStatus	Status (Content Information page)	State of a revision in the system. Possible values are: **GENWWW**: The file is being converted to web-viewable format or is being indexed, or has failed conversion or indexing. **DONE**: The file is waiting to be released on its specified Release Date. **RELEASED**: The revision is available. **REVIEW**: The revision is in a workflow and is being reviewed. **EDIT**: The revision is at the initial contribution step of a workflow. **PENDING**: The revision is in a Basic workflow and is waiting for approval of all revisions in the workflow. **EXPIRED**: The revision is no longer available for viewing. The revision was not deleted, but it can be accessed only by an administrator. **DELETED**: The revision has been deleted and is waiting to be completely removed during the next indexing cycle.
dWebExtension	None	File extension of the web-viewable file.

2.6.3 Option Lists

An option list is a set of values that can be selected for a metadata field. The following topics describe the use of option lists:

- "Internal Option Lists" on page 2-20
- "Option List Script" on page 2-21
- "Creating an Option List" on page 2-21

2.6.3.1 Internal Option Lists

Content Server maintains the following internal option lists by default:

Metadata Field	Option List
Author (dDocAuthor)	docAuthors

Metadata Field	Option List
Security Group (dSecurityGroup)	securityGroups
Type (dDocType)	docTypes
Account (dDocAccount)	docAccounts
Role (dRole)	roles

The *securityGroups* and *docAccounts* option lists are filtered according to the current user's permissions.

2.6.3.2 Option List Script

The following Idoc Script variables and functions are used to generate and enable option lists:

Variable or Function	Description
optList function	Generates the option list for a metadata field.
optionListName variable	Specifies the name of an option list.
fieldIsOptionList variable	Specifies that a metadata field has an option list.
fieldOptionListType variable	Specifies the type of option list (strict, combo, multi, or access).
hasOptionList variable	Set to the value of the *fieldIsOptionList* variable. This variable is used in conditional statements.
defaultOptionListScript variable	Defines a piece of Idoc Script that displays a standard option list field.
optionListScript variable	Overrides the standard implementation of option list fields (as defined by the *defaultOptionListScript* variable).
optionsAllowPreselect variable	Specifies that a metadata field option list can be prefilled with its last value.
addEmptyOption variable	Specifies that the first value in the option list is blank.
optionListResultSet variable	Specifies a ResultSet that contains option list values.
optionListKey variable	Specifies the name of a ResultSet column that contains option list values.
optionListValueInclude variable	Specifies an include that defines the values for an option list.

2.6.3.3 Creating an Option List

To create an option list, you can use one of the following methods:

- Use the optList function to generate a basic option list. Note that this only produces output when used with a service that calls "loadMetaOptionsList".

  For example, this code displays a list of possible authors as an HTML option list:

  ```
  <select name="dDocAuthors">
      <$optList docAuthors$>
  </select>
  ```

- Use the rsMakeFromList function to turn the option list into a ResultSet, and then loop over the ResultSet.

  For example, this code creates a ResultSet called *Authors* from the *docAuthors* option list, and loops over the ResultSet to create an HTML option list. (Because the column name is not specified as a parameter for *rsMakeFromList*, the column name defaults to `row`.)

  ```
  <$rsMakeFromList("Authors","docAuthors")$>
  <select name="dDocAuthors">
      <$loop Authors$>
          <option><$row$>
  <$endloop$>
  </select>
  ```

  These code samples are equivalent. Typically, you would use the *rsMakeFromList* function when you want to parse or evaluate the list options.

2.6.4 Referencing Metadata in Dynamic Server Pages

For dynamic server pages, several metadata values are stored with a *ref:* prefix, which makes them available to the page but does not replace ResultSet values. (This prevents pollution of ResultSets by dynamic server pages.)

When you reference any of the following metadata values on a dynamic server page, you must include the *ref:* prefix:

- hasDocInfo
- dDocName
- dExtension
- dSecurityGroup
- isLatestRevision
- dDocType
- dID

For example, the following statement determines if the document type is *Page*:

```
<$if strEquals(ref:dDocType,"Page"))$>
```

See the *Oracle Fusion Middleware Developer's Guide for Content Server* for more information.

2.7 Using Merge Includes to Format Responses

You can use a MergeInclude to format your results from a Content Server request based on an IdocScript include, rather than an entire template page.

A MergeInclude is a feature often used to integrate ASP pages using the IdcCommandX ActiveX module. The Content Server architecture is essentially a modular, secure, service-based application with multiple interfaces, although its architecture was designed to optimize the web interface. Services such as 'GET_SEARCH_RESULTS' will generate response data based on the 'QueryString' passed, and the user's security credentials. This response data is internally represented in the form of a HDA file. To see this in action, simply perform a search then add 'IsJava=1' to the URL. You will now see how data is internally represented for the response.

Because this HDA representation is not particularly useful for web-based users, so we use IdocScript includes and templates to format the response into a readable HTML page. A user can modify how this HTML is displayed by changing the template or a few resource includes with a component.

However, to retrieve only a small portion of this search result (for example, to display it on an ASP, JSP, or PHP page where the majority of the code is not IdocScript), or have an IFRAME or DIV element pop up and display the results, or to dynamically change how to display the results, you can simply add these parameters to your URL:

```
MergeInclude=my_custom_include&IsJava=1
```

This will cause the Content Server to bypass formatting the response according to the template that is specified in the service. It will instead format the response based on the IdocScript in 'my_custom_include'. For example, if you executed a search, then added the above line to the URL, and the include looked like this in your component:

```
<@dynamichtml my_custom_include@>
<html>
<table width=300>
<tr>
    <td><b>Name</b></td>
    <td><b>Title (Author)</b></td>
</tr>
<$loop SearchResults$>
<tr><td><a href="<$URL$>"><$dDocName$></a></td>
    <td><$dDocTitle$> (<$dDocAuthor$>)</td></tr>
<$endloop$>
</table>
</html>
<@end@>
```

This would display a search result page devoid of all images and formatting that you may not need. Consequently, you can format any Content Server response with any IdocScript include that you want. In theory, the IdocScript include can contain any kind of formatting that you want: XML, WML, or simply plain text.

For example, if you wanted to return search results in a format that can be read in an Excel Spreadsheet, you could create a resource include that returns a comma-delimited list of entries. You could then save the returned file to your hard drive, and then open it up in Excel. Another useful trick would be to create a resource include that formats the response into a record set that can be read in as a file by the IdcCommandX utility, or the BatchLoader. Such an include could be used with a search result, or an Active Report created with the Weblayout Editor, to build up batch files specific to arbitrary queries against the database or against the search index.

- MergeInclude variables are cached differently than normal resource includes. Therefore, you must restart the Content Server if you make changes to the resource include. This can be bypassed if you execute a 'docLoadResourceInclude()' function to dynamically load different includes from within the MergeInclude.
- The content type of the returned data is 'text/plain' and not 'text/html' for data returned by a MergeInclude. Some clients (such as IE and many versions of Netscape) still display plain text as html if you have valid HTML in the response, others clients may not. If you experience problems, you may need to manually set the content type when you link to it.

2.8 Scoped Local Variables

Scoped local variables are a special kind of local variable, used to override how metadata is drawn to the page. These variables are scoped to a specific metadata field by separating them with a colon.

For example, to hide the title and comments fields, you would set the following flags:

```
dDocTitle:isHidden=1
xComments:isHidden=1
```

These flags must be set early in the page in the URL or by overriding the include `std_doc_page_definitions`.

In the following list, all flags affect the display of the field `xFieldName`:

- `xFieldName:groupHeader`: This is set in Content Profiles if this field is the first field in a group. It contains the HTML and IdocScript to use for the group header.
- `xFieldName:hasOptionList`: Allows the field to contain a custom option list, instead of using the default option list. Must be used with the `xFieldName:optionListName` variable or `xFieldName:optionListScript` variable.
- `xFieldName:include:` Used to set the value for `fieldInclude` to the name of a custom resource include. This resource will be used throughout the page, including the JavaScript and the HTML. This flag is used rarely. If needed, use the `std_namevalue_field` include file as a guide for making a custom include.
- `xFieldName:isExcluded`: Set to true to exclude a field from the page completely. It will not be displayed as a field, or as a hidden input field. The field will be completely absent from the page.
- `xFieldName:isHidden`: Set to TRUE to hide a field on the page. On pages with form posts, the field will still be present. However, it will only exist as a hidden INPUT field. The value of the field will be blank, unless `xFieldName` or `fieldValue` is defined. This will enable you to create pages with default values that cannot be changed.
- `xFieldName:isInfoOnly`: Set to TRUE to display only the value of a field. This is used instead of `xFieldName:isHidden` to show the user what default values are being submitted.
- `xFieldName:isRelocated`: Set to TRUE to stop the automatic display of a field on the HTML page. By default, all fields on the page have a specific order. To reorder them, you must set this flag, then display the field manually.

  ```
  <!-- hide the comments field -->
  <$xComments:isRelocated = 1$>
  <$loop DocMetaDefinition$>
  <$strTrimWs(inc("std_meta_field_display"))$>
  <$endloop$>
  <!-- now turn off relocation, and display it -->
  <$xComments:isRelocated = ""$>
  <$fieldName="xComments", fieldCaption="Comments", fieldType="Memo"$>
  <$include std_display_field$>
  ```

- `xFieldName:isRequired`: Set to TRUE to turn this field into a required field. This flag must be set in `std_doc_page_definitions`, before the JavaScript validation code is drawn to the page.

- xFieldName:maxLength: Similar to fieldWidth, this sets the maximum length of a text input field. This is usually greater than fieldWidth, and must be less than the width of the field in the database.
- xFieldName:noSchema: Set to TRUE to disable a schema option list for a field. Required if you want to generate option lists in a custom, dynamic way.
- xFieldName:optionListName: This flag can only be set if a field is an option list. You can override which option list to use to display values:

```
<$xCountry:hasOptionList = 1$>
<$xCountry:noSchema = 1$>
<$xCountry:optionListName = "securityGroups"$>
<$loop DocMetaDefinition$>
<$strTrimWs(inc("std_meta_field_display"))$>
<$endloop$>
```

- xFieldName:optionListScript: Similar to optionListName, except it can be used to render IdocScript instead of explicitly defined option lists. This allows the option list to be drawn with a ResultSet instead:

```
<$xCountry:hasOptionList = 1$>
<$xCountry:noSchema = 1$>
<$xCountry:optionListScript =
"<$rsMakeFromList('GROUPS', 'securityGroups')$>" &
"<select>\n" &
"<$loop GROUPS$>" &
" <option><$row$>" &
"<$endloop$>\n" &
"</select>"$>
<$loop DocMetaDefinition$>
<$strTrimWs(inc("std_meta_field_display"))$>
<$endloop$>
```

- xFieldName:rowClass: Used in std_nameentry_row. It sets a Cascading Style Sheet class for the table row that contains this field.

```
<$xComments:rowClass="xuiPageTitleText"$>
<$loop DocMetaDefinition$>
<$strTrimWs(inc("std_meta_field_display"))$>
<$endloop$>
```

- xFieldName:rowStyle: Same as rowClass, but this can be used to create inline styles. For example, to hide the Comments field with DHTML, use the following code:

```
<$xComments:rowStyle="display:none"$>
<$loop DocMetaDefinition$>
<$strTrimWs(inc("std_meta_field_display"))$>
<$endloop$>
```

  This is useful when you want to hide and display fields dynamically without a page reload.

3

Master List: Idoc Script by Type

This chapter describes the different types of Idoc Script variables and functions, and groups the variables and functions by type.

3.1 Conditional Dynamic Variables

Some dynamic variables are conditional and can only be used within a conditional statement such as *if*, *while*, *elseif*, or *loop*. These variables have the following special features:

- Conditional variables are internal flags that are *gettable* but not *settable*.
- Conditional variables will only provide a Boolean response and do not return a value such as a string or integer.
- Conditional variables will not accept the #*active* keyword prefix. Thus, an error report is printed to the debug output if the variable is not found.

The following is a list of Idoc Script conditional dynamic variables:

"AdminAtLeastOneGroup" on page 6-5
"AfterLogin" on page 8-5
"AllowCheckin" on page 6-6
"AllowCheckout" on page 6-7
"AllowReview" on page 6-12
"dcShowExportLink" on page 10-23
"EmptyAccountCheckinAllowed" on page 6-44
"HasPredefinedAccounts" on page 6-70
"HasUrl" on page 6-71
"HeavyClient" on page 8-121
"IsCheckinPreAuthed" on page 6-107

"isComponentEnabled" on page 6-108
"IsContributor" on page 6-109
"IsCriteriaSubscription" on page 6-110
"IsCurrentNav" on page 8-156
"IsDynamic" on page 8-159
"IsExternalUser" on page 8-162
"IsFilePresent" on page 6-113
"IsFullTextIndexed" on page 6-115
"IsLocalSearchCollectionID" on page 6-119
"IsLoggedIn" on page 6-120
"IsMac" on page 8-174
"IsMaxRows" on page 8-175
"IsMultiPage" on page 8-177
"IsNotSyncRev" on page 6-122
"IsPromptingForLogin" on page 6-124
"IsRequestError" on page 8-183
"IsSubAdmin" on page 6-127
"IsSun" on page 8-188
"IsSysManager" on page 6-128
"IsUploadSockets" on page 6-129
"IsUserEmailPresent" on page 8-192
"IsWindows" on page 8-194
"IsWorkflow" on page 6-133
"SingleGroup" on page 6-242
"UserIsAdmin" on page 6-269
"UserLanguageID" on page 8-347
"UserLocaleId" on page 8-348
"UseXmlUrl" on page 6-273

3.2 Configuration Variables

Configuration variables are predefined Idoc Script variables that are generally used as settings in configuration files. These variables can be used within Idoc Script to detect whether a configuration setting is enabled or to return the value of the configuration setting.

- Configuration variables that pass a Boolean value can be set only to the value TRUE (1) or FALSE (0).
- Unless otherwise specified, Boolean-type configuration values will default to FALSE (0), and string-type configuration values will default to an empty string.

This section lists the configuration variables by the configuration file (typically *.cfg* or *.hda*) in which they are most commonly located. Your actual application will vary depending on the criteria and selections made during installation.

- "Content Server Configuration Variables" on page 3-2
- "Inbound Refinery Configuration Variables" on page 3-9
- "Pop-Up Calendar Configuration Variables" on page 3-10
- "Record Manager and Retention Manager Configuration Variables" on page 3-10

3.2.1 Content Server Configuration Variables

The following configuration files are located in these configuration directories:

"DomainHome/ucm/cs/bin/intradoc.cfg" on page 3-3

"IntradocDir/config/config.cfg" on page 3-3
"IdcHomeDir/components/folders_g/folders_environment.cfg" on page 3-8
"IntradocDir/data/providers/provider_name/provider.hda" on page 3-8

3.2.1.1 *DomainHome*/ucm/cs/bin/intradoc.cfg

The following configuration variables are located in the *DomainHome*/ucm/cs/bin/intradoc.cfg file. These variables define directory paths and global settings for the Content Server and Admin Server.

"AllowConcurrentUpdate" on page 6-10
"ARCHIVE_COLLECTION" on page 8-7
"BatchLoaderPath" on page 8-13
"BatchLoaderUserName" on page 8-14
"CleanUp" on page 8-23
"CollectionLocation" on page 8-25
"DataDir" on page 8-30
"EnableErrorFile" on page 8-70
"FileEncoding" on page 8-92
"HTMLEditorPath" on page 8-122
"IdcAnalyzeLogDir" on page 8-142
"IdcCommandFile" on page 6-84
"IdcCommandLog" on page 6-85
"IdcCommandUser" on page 6-87
"IdcNativeDir" on page 10-48
"IntradocDir" on page 8-152
"isPhysicallySplitDir" on page 8-180
"JvmCommandLine" on page 8-200
"LogMessagePrefix" on page 8-209
"MaxErrorsAllowed" on page 8-213
"PreviewOutputExtension" on page 10-81
"PreviewPath" on page 10-82
"VaultDir" on page 8-358
"WebBrowserPath" on page 8-360
"WeblayoutDir" on page 8-361

3.2.1.2 *IntradocDir*/config/config.cfg

The following configuration variables are located in the *IntradocDir*/config/config.cfg file. These variables define instance settings for the Admin Server and most of the configuration settings for the Content Server.

"AccessListPrivilegesGrantedWhenEmpty" on page 8-3
"AdditionalIndexBuildParams" on page 6-2
"AdditionalSubscriptionCollatedFieldList" on page 6-4
"AllowAllRedirectUrls" on page 6-8
"AllowAlternateMetaFile" on page 6-9
"AllowMatchesInDateCheck" on page 8-6
"AllowPrimaryMetaFile" on page 6-11
"AllowSamePrimaryAlternateExtensions" on page 6-13
"AppletChunkSize" on page 6-14
"AppletChunkThreshold" on page 6-15
"ARCHIVE_COLLECTION" on page 8-7
"ArchiverDoLocks" on page 8-8
"ArchiverTouchMonitorInterval" on page 8-9
"ArchiverTraceLockMaintenance" on page 8-10
"ArchiverTraceLocks" on page 8-11

"AuthorDelete" on page 6-16
"AutoNumberPrefix" on page 6-17
"AutoNumberWidth" on page 6-18
"BinDir" on page 8-15
"CachedResultRowCount" on page 6-19
"ChunkedRequestTrace" on page 8-22
"CollectionLocation" on page 8-25
"ColumnMapFile" on page 6-21
"ConfigDir" on page 8-26
"ConversionCacheExpirationPeriodInDays" on page 10-20
"CookieAuthTimeoutInDays" on page 6-25
"CookieProxyPassword" on page 6-26
"createAlternateMetaFile" on page 6-27
"createPrimaryMetaFile" on page 6-28
"DatabasePreserveCase" on page 6-29
"DatabaseWildcards" on page 6-30
"DatedCacheIntervalDays" on page 10-21
"DateOutputFormat" on page 8-32
"DCMaxFileSize" on page 10-22
"DCTimeOut" on page 10-24
"DCViewFormat" on page 10-25
"Default Accounts" on page 6-31
"DefaultFilterInputFormat" on page 10-27
"DefaultFilterOutputFormat" on page 10-28
"DefaultHtmlConversion" on page 10-30
"DefaultHtmlRefreshTimeoutInSeconds" on page 6-32
"DefaultPasswordEncoding" on page 6-33
"DirectoryLockingLogPath" on page 8-56

"DisableContentProfiles" on page 8-57
"DisableDBLongIntegerField" on page 6-34
"DisableForcedConversions" on page 10-33
"DisableHttpUploadChunking" on page 6-36
"DisableKeepingForcedConversions" on page 10-34
"DisableOptimizedLatestReleaseQuerying" on page 10-35
"DisableSharedCacheChecking" on page 8-59
"DoDocNameOrder" on page 6-40
"DomainControllerName" on page 6-41
"DownloadApplet" on page 8-63
"DtmPort" on page 10-37
"EnableDocumentHighlight" on page 6-45
"EnableEnvironmentPackager" on page 8-69
"EnableExpirationNotifier" on page 6-46
"EnableOptimizedLatestReleaseQuerying" on page 10-39
"EnableRebuildForcesReconversion" on page 10-40
"EnableSchemaPublish" on page 6-48
"ExclusiveCheckout" on page 6-51
"ExpireRevisionOnly" on page 6-52
"ForcedConversionRules" on page 10-41
"ForceDistinctRevLabels" on page 6-55
"ForceDocTypeChoice" on page 8-93
"ForceJdbcSync" on page 6-56
"ForceSecurityGroupChoice" on page 8-96
"FormInheritMetaDataFieldList" on page 6-58
"GetCopyAccess" on page 6-61

"HasExternalUsers" on page 6-68
"HighlightBegin" on page 10-46
"HtmlDataInputFilterLevel" on page 6-72
"HighlightEnd" on page 10-47
"HTMLEditorPath" on page 8-122
"HttpRelativeWebRoot" on page 8-138
"HttpServerAddress" on page 8-139
"IdcAuthExtraRequestParams" on page 6-83
"IdcCommandServerHost" on page 6-86
"IdcHttpHeaderVariables" on page 6-88
"IdcHomeDir" on page 8-143
"IdcLocale" on page 8-144
"IDC_Name" on page 6-82
"IdcServerBindAddress" on page 8-145
"IdcServerSocketQueueDepth" on page 8-146
"IndexableRenditions" on page 6-89
"IndexerAutoWorkInterval" on page 6-90
"IndexerLargeFileSize" on page 6-91
"IndexerRebuildStagingManualCopy" on page 6-92
"IndexerRebuildStagingPath" on page 6-93
"IndexerTouchMonitorInterval" on page 6-95
"IndexerTraceCycleListChanges" on page 6-96
"IndexerTraceLockFileChanges" on page 6-97
"IndexerTraceStartStop" on page 6-98
"InstanceDescription" on page 6-99
"InstanceMenuLabel" on page 6-100
"IntradocRealm" on page 6-101
"IntradocServerHostName" on page 6-102
"IntradocServerPort" on page 6-103
"IsAllowIndexRenditions" on page 6-104
"IsAutoArchiver" on page 8-154
"IsAutoNumber" on page 6-105
"IsAutoQueue" on page 10-52
"IsAutoSearch" on page 6-106
"IsContentRefineryPresent" on page 10-53
"IsDpSubmitErrorFatal" on page 8-158
"IsWebServerPagesOnly" on page 6-132
"IsDynamicConverterEnabled" on page 10-54
"IsFormsPresent" on page 6-114
"rule:IsFragmentOnly" on page 10-117
"IsJdbcLockTrace" on page 6-116
"IsJdbcQueryTrace" on page 6-117
"IsJspServerEnabled" on page 8-173
"IsOverrideFormat" on page 8-179
"IsSavedWfCompanionFile" on page 6-126
"JdbcConnectionString" on page 6-135
"JdbcDriver" on page 6-136
"JdbcPassword" on page 6-137
"JdbcPasswordEncoding" on page 6-138
"JdbcUser" on page 6-139
"JspAdminQuery" on page 8-197
"JspDefaultIndexPage" on page 8-198
"JspEnabledGroups" on page 8-199
"LatestReleasedTimestampTimeout" on page 10-56
"LmDefaultLayout" on page 6-156

"LmDefaultSkin" on page 6-157
"LocaleArchiverDateFormat" on page 8-207
"LogName" on page 8-210
"MacSupportsSignedApplets" on page 8-211
"MailServer" on page 6-162
"MajorRevSeq" on page 6-163
"MaxArchiveErrorsAllowed" on page 8-212
"MaxCollectionSize" on page 6-164
"MaxConversionCacheSizeInMegs" on page 10-66
"MaxDocIndexErrors" on page 6-165
"MaxHitCount" on page 6-166
"MaxIntegerAllowed" on page 6-167
"MaxQueryRows" on page 6-168
"MaxRequestThreadCount" on page 6-169
"MaxResults" on page 6-170
"MaxSavedSearchResults" on page 6-171
"MaxSearchConnections" on page 6-172
"MaxSizeResourceFileCacheMillions" on page 6-173
"MemoFieldSize" on page 6-175
"MinIntegerAllowed" on page 6-176
"MinMemoFieldSize" on page 6-177
"MinorRevSeq" on page 6-178
"MultiUpload" on page 8-216
"NeverTimeoutForcedConversionsTemplateChange" on page 10-69
"NoAutomation" on page 6-179
"noInfoColumn" on page 8-217
"noRevisionsColumn" on page 8-219
"NoSearchConnectionCloseOnDocChange" on page 6-181
"NotificationIntervalInDays" on page 6-182
"NotificationMaximum" on page 6-183
"NotificationQuery" on page 6-184
"NotifyExtras" on page 6-185
"NotifyTime" on page 6-186
"NtlmSecurityEnabled" on page 6-187
"NumAdditionalRenditions" on page 10-70
"NumConnections" on page 188
"OdbcDateFormat" on page 8-221
"PrimaryWorkQueueTimeout" on page 6-193
"ProviderDir" on page 8-236
"QueueDir" on page 10-109
"RedirectHostsFilter" on page 6-200
"RequestAuditAdditionalVerboseFieldsList" on page 8-237
"RequestAuditIntervalSeconds1" on page 8-238
"RequestAuditIntervalSeconds2" on page 8-239
"RequestAuditListDepth1" on page 8-240
"RequestAuditListDepth2" on page 8-241
"RequestAuditSubClassificationField" on page 8-242
"SchemaAllowRecursiveFileMove" on page 6-208
"SchemaBuildDigestFile" on page 6-209
"SchemaPublishCheckInterval" on page 6-210
"SchemaPublishInterval" on page 6-211
"SchemaPublishMaximumInterval" on page 6-212
"SchemaPublishMinimumInterval" on page 6-213
"SchemaPublishRenameRetryInitialInterval" on page 6-214
"SchemaPublishRetryInterval" on page 6-216

"SchemaPublishRetryIntervalMax" on page 6-217
"SchemaPublishRenameRetryIterations" on page 6-215
"SchemaVerifyPublishing" on page 6-218
"SchemaWin32RenameDelayInterval" on page 6-219
"SearchCacheCleanerInterval" on page 6-221
"SearchCacheHardLimit" on page 6-222
"SearchCacheMaxItemAgeInMinutes" on page 6-223
"SearchCacheSoftLimit" on page 6-224
"SearchCacheTrace" on page 6-225
"SearchConnectionWaitTimeout" on page 6-226
"SearchDebugLevel" on page 6-227
"SearchDir" on page 6-228
"SearchIndexerEngineName" on page 6-229
"SearchQueryFormat" on page 6-230
"SearchQueryTrace" on page 6-231
"SelfRegisteredAccounts" on page 6-232
"SelfRegisteredRoles" on page 6-233
"ServiceStartupTimeout" on page 6-238
"ShowCreateExternalMenu" on page 6-240
"showDocActions" on page 8-292
"ShowOnlyKnownAccounts" on page 6-241
"ShowRevisionsColumn" on page 8-293
"showSearchResultsCheckoutColumn" on page 8-294
"SkipMozillaMailToWorkaround" on page 6-243
"SmtpPort" on page 6-244
"SocketHostAddressSecurityFilter" on page 6-245
"SocketHostNameSecurityFilter" on page 6-246
"StdPageWidth" on page 8-295
"StrConfineOverflowChars" on page 8-299
"SubjectsDir" on page 8-317
"SubscriptionMethod" on page 6-253
"SubscriptionCollatedFieldList" on page 6-252
"SysAdminAddress" on page 8-318
"SystemDateFormat" on page 8-319
"SystemLocale" on page 8-320

"SystemTimeZone" on page 8-321
"TempDir" on page 8-322
"TimeoutChangedTemplateForcedConversionsInMins" on page 10-121
"TimeoutMismatchedForcedConversionsInMins" on page 10-122
"TimeoutPerOneMegInSec" on page 6-255
"TraceConversionCacheCleanup" on page 10-123
"TraceResourceConflict" on page 8-330
"TraceResourceLoad" on page 8-331
"TraceResourceOverride" on page 8-332
"UniversalSearchAllowFullText" on page 6-257
"UploadApplet" on page 8-334
"UseAccounts" on page 6-258
"UseAdsi" on page 6-259
"UseBellevueLook" on page 8-337
"UseCascadingStyleSheets" on page 10-124
"UseFourDigitYear" on page 8-338
"UseNtlm" on page 6-263
"UserCache" on page 8-342
"UserCacheTimeout" on page 8-343

"UseRedirectedOutput" on page 8-345
"UseRevisionCreateDate" on page 6-264
"UserFilteredSubjects" on page 6-265
"UserPublishCacheDir" on page 8-350
"UseSearchCache" on page 6-270
"UseSSL" on page 6-272
"UseXpedioLook" on page 8-352
"WarnAboutTimeZone" on page 8-359
"WebServerAuthOnly" on page 6-274
"WorkflowDir" on page 6-305
"WorkflowIntervalHours" on page 6-306
"xml" on page 8-362
"XMLEncodingMode" on page 6-307

3.2.1.3 *IdcHomeDir*/components/folders_g/folders_environment.cfg

The following configuration variables are located in the folders_environment.cfg file in the *IdcHomeDir*/components/folders_g/ directory. These variables define settings for Content Server folders.

"CollectionDeleteEnabled" on page 10-6
"CollectionHiddenMeta" on page 10-7
"CollectionInhibitUpdateMeta" on page 10-8
"CollectionMeta" on page 10-9
"CollectionMoveEnabled" on page 10-10
"CollectionReadOnlyMarkedFolders" on page 10-11
"CollectionReadOnlyMeta" on page 10-12
"CollectionReleasedOnly" on page 10-13
"CollectionSecurityReadOnly" on page 10-14
"CollectionTrashDeleteDate" on page 10-15
"CollectionTrashDeleteLocation" on page 10-16
"CollectionTrashDeleteOldName" on page 10-17
"CollectionTrashDeleter" on page 10-18
"CollectionWebDAVServer" on page 10-19
"InitialColID" on page 10-51

3.2.1.4 *IntradocDir*/data/providers/provider_name/provider.hda

The following configuration variables are located in the provider.hda file in the *IntradocDir*/data/providers/*provider_name*/ directory. These variables define settings for Content Server providers.

"IsPrimaryUserBase" on page 6-123
"LdapAdminDN" on page 6-142
"LdapAdminPassword" on page 6-143
"LdapPort" on page 6-144
"LdapServer" on page 6-145
"LdapSuffix" on page 6-146
"NumConnections" on page 6-188
"Priority" on page 6-194
"ProviderClass" on page 6-195
"ProviderConfig" on page 6-196
"ProviderConnection" on page 6-197
"ProviderType" on page 6-198
"SourcePath" on page 6-248
"UseNetscape" on page 6-262
"UseSecureLdap" on page 6-271

3.2.2 Inbound Refinery Configuration Variables

The following configuration files are located in the Inbound Refinery installation directory (noted by *IntradocDir*/config/config.cfg).

Configuration variables returning a string that defines a directory or file path related to the Inbound Refinery will return that path relative to the Inbound Refinery installation directory. These paths are generally not recognizable to the Content Server nor usable to the user if passed as information within an HTM page.

"AllowSecondaryConversion" on page 10-4
"AllowSkippedHyperlinktoCauseIncomplete" on page 10-5
"DefaultGraphicTimeout" on page 10-29
"DefaultNativeTimeout" on page 10-31
"DefaultPostscriptTimeout" on page 10-32
"DistillerPath" on page 10-36
"FileEncoding" on page 8-92
"FrameMakerCyclesAllowed" on page 10-42
"FrameMakerTimeout" on page 10-43
"IntradocDir" on page 8-152
"JvmCommandLine" on page 8-200
"MaxNumRecursiveStepDefinitions" on page 10-67
"MSPubexePath" on page 10-68
"OpenOfficeListenerPort" on page 10-71
"OpenOfficeListenerURL" on page 10-72
"PageMakerExePath" on page 10-74
"PostConversionTimeOut" on page 10-75
"PostprocessPDFPath" on page 10-76
"PostscriptPrinterName" on page 10-77
"PowerPointPrintOutput" on page 10-78
"PreconversionPath" on page 10-79
"PreconversionWaitTime" on page 10-80
"PrinterPortPath" on page 10-83
"ProcessExcelUrlHyperlinks" on page 10-84
"ProcessInDesignBookMarks" on page 10-85
"ProcessInDesignHyperlinks" on page 10-86
"ProcessInDesignInternalPDFThumbnail" on page 10-87
"ProcessInDesignLinkUpdate" on page 10-88
"ProcessInDesignPDFCompatibility50" on page 10-89
"ProcessPowerpointHyperlinks" on page 10-90
"ProcessVisioInternalHyperlinks" on page 10-91
"ProcessVisioUrlHyperlinks" on page 10-92
"ProcessWordBookMarks" on page 10-93
"ProcessWordComments" on page 10-94
"ProcessWordEmailLinks" on page 10-95
"ProcessWordFootEndNote" on page 10-96
"ProcessWordHeadings" on page 10-97
"ProcessWordHyperlinks" on page 10-98
"ProcessWordToc" on page 10-99
"ProcessWordUrlHyperlinks" on page 10-100
"SecondaryConversionStepName" on page 10-118
"VaultDir" on page 8-358
"VerboseMode" on page 10-125
"WeblayoutDir" on page 8-361

3.2.3 Link Manager Configuration Variables

The following variables are used to configure the Link Manager application. These variables are set in the *IntradocDir*/ucm/cs/bin/config.cfg file.

"AllowForceDelete" on page 10-2
"HasSiteStudio" on page 10-44
"LkDisableOnRebuild" on page 10-57
"LkDisableReferenceInfo" on page 10-58
"LkDisallowConversionFormats" on page 10-59
"LkIsSecureSearch" on page 10-60
"LkReExtractOnRebuild" on page 10-61
"LkRefreshBatchSize" on page 10-62
"LkRefreshErrorsPercent" on page 10-63
"LkRefreshErrorsAllowed" on page 10-64
"LkRefreshErrorThreshold" on page 10-65

3.2.4 Pop-Up Calendar Configuration Variables

The following variables are used to configure the Pop-Up Calendar application. These variables are set in the *IntradocDir*/config/config.cfg file.

"pucDefaultTime" on page 10-101
"pucEnableOnSearch" on page 10-102
"pucHideTimeInDateFields" on page 10-103
"pucUseMonthLocale" on page 10-104
"pucUseWeekdayLocale" on page 10-105
"pucXOffset" on page 10-106
"pucYOffset" on page 10-107
"pucZIndex" on page 10-108

3.2.5 Record Manager and Retention Manager Configuration Variables

"AllowRetentionPeriodWithoutCutoff" on page 10-3
"HideVitalReview" on page 10-45
"RecordsManagementNumberOverwriteOnDelete" on page 10-111
"RecordsManagementDenyAuthorFreePassOnRMSecurity" on page 10-110
"RmaAddDocWhereClauseForScreening" on page 10-112
"RMAHideExternalFieldsFromCheckInUpdate" on page 10-113
"RMAHideExternalFieldsFromSearchInfo" on page 10-114
"RmaNotifyDispReviewerAndCatAuthor" on page 10-115
"RmaNotifyReviewerAndAlternateReviewer" on page 10-116
"ShowContentForStorageBrowse" on page 10-119

3.3 Dynamic Variables

A *dynamic* variable is evaluated on each occurrence of the variable. Thus, each time the variable is encountered the value is recalculated from code. (In contrast, a *value* variable is evaluated once at the beginning of the service call and that value is used throughout the service call. See "Value Variables" on page 3-21.) Dynamic variables generally return a value such as a string or an integer.

The following is a list of Idoc Script dynamic variables:

"AfterLogin" on page 8-5
"HttpIgnoreWebServerInternalPortNumber" on page 6-81

3.4 Environment Variables

Web server variables are the CGI environment variables that are set when the server executes the gateway program. In order to pass data about the information request from the server to the script, the server uses command-line arguments and environment variables. These environment variables can be used to output information to a log file or can be used within Idoc Script statements and as part of evaluations.

For example, this Idoc Script statement evaluates whether the remote host address matches a specific string:

```
<$if strEquals("207.0.0.1",REMOTE_HOST)$>
```

This HTML and Idoc Script markup displays a list of web server environment information on the page:

```
<P>HTTP_INTERNETUSER=<$HTTP_INTERNETUSER$></P>
<P>REMOTE_HOST=<$REMOTE_HOST$></P>
<P>SCRIPT_NAME=<$SCRIPT_NAME$></P>
```

The following is a list of Web server variables:

"CONTENT_LENGTH" on page 6-24
"GATEWAY_INTERFACE" on page 6-59
"HTTP_ACCEPT" on page 6-73
"HTTP_ACCEPT_ENCODING" on page 6-74
"HTTP_ACCEPT_LANGUAGE" on page 6-75
"HTTP_COOKIE" on page 6-76
"HTTP_HOST" on page 6-77
"HTTP_INTERNETUSER" on page 6-78
"HTTP_REFERER" on page 6-79
"HTTP_USER_AGENT" on page 6-80
"PATH_INFO" on page 6-190
"PATH_INFO" on page 6-190
"QUERY_STRING" on page 6-199
"REMOTE_ADDR" on page 6-204
"REMOTE_HOST" on page 6-205
"REQUEST_METHOD" on page 6-206
"SCRIPT_NAME" on page 6-220
"SERVER_NAME" on page 6-234
"SERVER_PORT" on page 6-235
"SERVER_PROTOCOL" on page 6-236
"SERVER_SOFTWARE" on page 6-237

3.5 Global Functions

Idoc Script has many built-in global functions. Functions perform actions, including string comparison and manipulation routines, date formatting, and ResultSet manipulation. Some functions also return results, such as the results of calculations or comparisons.

Information is passed to functions by enclosing the information in parentheses after the name of the function. Pieces of information that are passed to a function are called *parameters*. Some functions do not take parameters; some functions take one parameter; some take several. There are also functions for which the number of parameters depends on how the function is being used.

The following is a list of Idoc Script global functions:

"abortToErrorPage" on page 8-2
"break" on page 8-16
"cacheInclude" on page 8-19
"clearSchemaData" on page 6-20
"computeDocUrl" on page 6-22
"computeRenditionUrl" on page 6-23
"dateCurrent" on page 8-31
"ddAppendIndexedColumnResultSet" on page 8-33
"ddAppendResultSet" on page 8-35
"ddApplyTableSortToResultSet" on page 8-36
"ddGetFieldList" on page 8-37
"ddIncludePreserveValues" on page 8-38
"ddLoadIndexedColumnResultSet" on page 8-40
"ddLoadResultSet" on page 8-41
"ddMergeIndexedColumnResultSet" on page 8-42
"ddMergeResultSet" on page 8-44
"ddMergeUsingIndexedKey" on page 8-46
"ddSetLocal" on page 8-48
"ddSetLocalByColumnsFromFirstRow" on page 8-49
"ddSetLocalByColumnsFromFirstRowIndexed" on page 8-50
"ddSetLocalEmpty" on page 8-51
"ddSetLocalEmptyByColumns" on page 8-52
"docLoadResourceIncludes" on page 8-60
"docRootFilename" on page 8-62
"docUrlAllowDisclosure" on page 6-39
"dpGet" on page 8-65
"dpPromote" on page 8-66
"dpPromoteRs" on page 8-67
"dpSet" on page 8-68
"encodeHtml" on page 6-49
"eval" on page 8-71
"executeService" on page 8-73
"forceExpire" on page 8-94
"formatDate" on page 8-97
"formatDateDatabase" on page 6-57
"formatDateDisplay" on page 8-98
"formatDateOnly" on page 8-99
"formatDateOnlyDisplay" on page 8-100
"formatDateOnlyFull" on page 8-101
"formatDateWithPattern" on page 8-102
"formatTimeOnly" on page 8-103
"formatTimeOnlyDisplay" on page 8-104
"getDebugTrace" on page 8-106
"getErrorTrace" on page 8-107
"getFieldViewDisplayValue" on page 6-63
"getFieldViewValue" on page 8-108
"getFreeMemory" on page 8-109
"getParentValue" on page 6-64
"getRequiredMsg" on page 8-112
"getTextFile" on page 8-113
"getTotalMemory" on page 8-114
"getUserValue" on page 8-115
"getValue" on page 8-116
"getValueForSpecifiedUser" on page 8-118
"getViewValue" on page 6-65

"getViewValueResultSet" on page 6-66
"hasAppRights" on page 6-67
"idocTestForInclude" on page 8-147
"inc" on page 8-148
"incDynamicConversionByRule" on page 10-50
"incGlobal" on page 8-149
"incTemplate" on page 8-151
"indexerSetCollectionValue" on page 6-94
"isActiveTrace" on page 8-153
"isFalse" on page 8-163
"isLayoutEnabled" on page 6-118
"isTrue" on page 8-189
"isUserOverrideSet" on page 6-130
"isValidateFile" on page 6-131
"js" on page 8-196
"jsFilename" on page 6-140
"lc" on page 8-201
"lcCaption" on page 8-202
"LmDefaultLayout" on page 6-156
"LmDefaultSkin" on page 6-157
"lmGetLayout" on page 6-158
"lmGetSkin" on page 6-159
"loadCollectionInfo" on page 6-160
"loadDocMetaDefinition" on page 8-203
"loadDocumentProfile" on page 8-204
"loadSchemaData" on page 6-161
"loadSearchOperatorTables" on page 8-205
"loadUserMetaDefinition" on page 8-206
"optList" on page 8-228
"parseDataEntryDate" on page 8-231
"parseDateWithPattern" on page 8-234
"pneNavigation" on page 8-235
"regexMatches" on page 6-201
"regexReplaceAll" on page 6-202
"regexReplaceFirst" on page 6-203
"rptDisplayMapValue" on page 8-244
"rs" on page 8-245
"rsAddFields" on page 8-246
"rsAddFieldsWithDefaults" on page 8-247
"rsAddRowCountColumn" on page 8-249
"rsAppend" on page 8-250
"rsAppendNewRow" on page 8-251
"rsAppendRowvalues" on page 8-252
"rsCopyFiltered" on page 8-253
"rsCreateReference" on page 8-254
"rsCreateResultSet" on page 8-255
"rsDeleteRow" on page 8-256
"rsDocInfoRowAllowDisclosure" on page 8-257
"rsExists" on page 8-258
"rsFieldByIndex" on page 8-259
"rsFieldExists" on page 8-260
"rsFindRowPrimary" on page 8-261
"rsFirst" on page 8-262
"rsInsertNewRow" on page 8-263
"rsIsRowPresent" on page 8-264

"rsLoopInclude" on page 8-265

"rsLoopSingleRowInclude" on page 8-266

"rsMakeFromList" on page 8-267

"rsMakeFromString" on page 8-269

"rsMerge" on page 8-271

"rsMergeDelete" on page 8-272

"rsMergeReplaceOnly" on page 8-273

"rsNext" on page 8-274

"rsNumFields" on page 8-275

"rsNumRows" on page 8-276

"rsRemove" on page 8-277

"rsRename" on page 8-278

"rsRenameField" on page 8-279

"rsSetRow" on page 8-280

"rsSort" on page 8-281

"rsSortTree" on page 8-282

"setContentType" on page 8-286

"setExpires" on page 8-287

"setHttpHeader" on page 8-288

"setMaxAge" on page 8-289

"setValue" on page 8-291

"setResourceInclude" on page 8-290

"stdSecurityCheck" on page 6-251

"strCenterPad" on page 8-296

"strCommaAppendNoDuplicates" on page 8-297

"strConfine" on page 8-298

"strEquals" on page 8-300

"strEqualsIgnoreCase" on page 8-301

"strGenerateRandom" on page 8-302

"strIndexOf" on page 8-303

"strLeftFill" on page 8-304

"strLeftPad" on page 8-305

"strLength" on page 8-306

"strLower" on page 8-307

"strRemoveWs" on page 8-308

"strReplace" on page 8-309

"strReplaceIgnoreCase" on page 8-310

"strRightFill" on page 8-311

"strRightPad" on page 8-312

"strSubstring" on page 8-313

"strTrimWs" on page 8-314

"strUpper" on page 8-315

"toInteger" on page 8-327

"trace" on page 8-328

"url" on page 8-335

"urlEscape7Bit" on page 8-336

"userHasAccessToAccount" on page 6-266

"userHasGroupPrivilege" on page 6-267

"userHasRole" on page 6-268

"utGetValue" on page 8-353

"utLoad" on page 8-354

"utLoadDocumentProfiles" on page 8-355

"utLoadResultSet" on page 8-356

"xml" on page 8-362

3.6 Page Variables

Page variables are set on a particular web page to enable specific page attributes or functionality. A page variable applies only to the page on which it is set.

This section includes these topics:

- "Page Display Variables" on page 3-15
- "Field Display Variables" on page 3-15

3.6.1 Page Display Variables

Page variables that affect page display are typically set near the top of the page. Page display variables should be used as read-only variables; setting or changing the value of any of these variables will typically change the way all metadata is displayed on the page, which in most cases is not the desired effect.

The following is a list of Idoc Script page display variables:

"generateUniqueId" on page 8-105
"isCheckin" on page 8-155
"isDocPage" on page 8-157
"isEditMode" on page 8-160
"isFormSubmit" on page 8-168
"isInfo" on page 8-170
"isQuery" on page 8-181
"isUpdate" on page 8-190
"isUploadFieldScript" on page 8-191
"localPageType" on page 8-208
"noMCPrefill" on page 8-218

3.6.2 Field Display Variables

Field display variables can be grouped into the following types:

- "Field Information Variables" on page 3-15
- "Common Field Display Variables" on page 3-16
- "Other Field Display Variables" on page 3-17

3.6.2.1 Field Information Variables

The following variables define information about a metadata field. The variable values are loaded or computed for each metadata field.

"fieldCaption" on page 8-76
"fieldDefault" on page 8-79
"fieldIsOptionList" on page 8-84
"fieldName" on page 8-86
"fieldOptionListType" on page 8-87
"fieldType" on page 8-88
"fieldValue" on page 8-89
"fieldWidth" on page 8-91
"isRequired" on page 8-184
"requiredMsg" on page 8-243
"valueStyle" on page 8-357

3.6.2.1.1 Field Information Variables

The `std_prepare_metafield_include` include in the resource file named *IdcHomeDir*/resources/core/std_page.htm loads a number of field information variables from the local data in preparation for displaying the current metadata field:

```
<@dynamichtml std_prepare_metafield_include@>
<!--Prepare for presenting field-->
<$fieldName=dName, fieldCaption=dCaption, fieldDefault=dDefaultValue$>
<$fieldType=dType, fieldIsOptionList=dIsOptionList,
fieldOptionListType=dOptionListType$>
<@end@>
```

3.6.2.2 Common Field Display Variables

There are several commonly used page variables that affect the display of metadata fields. These variables can be set using different syntaxes at different places on a page, depending on how they are being used.

The following formats can be used to set a special field display variable:

- **Name/Value pair:** The variable is set using the standard `variable=value` format. For example, `isHidden=1`. This format is typically used to set the display of the current metadata field at the point in the page where the field is being generated by looped code.
- **FieldName:Variable format:** The variable is set by defining it as a parameter for the metadata field it applies to. For example, `myMetadata:isHidden`. This format is typically used at the top of a page to set the global display of a particular metadata field.

If a common field display variable is set at the top of a template page, it should be placed before the <HEAD> tag. Placing the variable in or after the <HEAD></HEAD> section will result in the field being displayed (or not displayed) as you intended, but the JavaScript validation code in the header will still be evaluated, so an … is not an object error will be thrown when you attempt to display a checkin page.

The following is a list of the common field display variables:

"isExcluded" on page 8-161
"isHidden" on page 8-169
"isInfoOnly" on page 8-171
"isRelocated" on page 8-182
"maxLength" on page 8-214
"optionListScript" on page 8-225

If these common field display variables are not sufficient to provide the required flexibility, the entire implementation of a metadata field can be replaced by setting the field variable to the name of a resource include that should be used instead (for example, `myField:include=customInclude`).

The standard implementation is referred to by the variable *defaultFieldInclude* , whose value is different depending on whether the field is being generated on a checkin/update, query, or info page. It also varies considerably based on the type of field being displayed. If the standard field include is overridden, then the new implementation must take into consideration all the issues of the different pages, including JavaScript validation and the Upload applet.

Use this approach only as a last resort. It is preferable to extend existing functionality and set local variables to have custom functionality.

If you use the *include* tag in this way to insert custom HTML code for a special metadata field, you must place the include statement after the </HEAD> tag on the page. If you place it before the </HEAD> tag, the system will insert your custom HTML code into the header and attempt to read it as JavaScript.

3.6.2.2.1 Other Field Display Variables A number of other variables are available to affect the display of metadata fields. Generally, these are used to define the display of a metadata field depending on which field is currently being generated and the value of related common field display variables.

The following is a list of other field display variables:

"addEmptyOption" on page 8-4
"captionEntryWidth" on page 8-20
"captionFieldWidth" on page 8-21
"defaultFieldInclude" on page 8-53
"defaultOptionListScript" on page 8-54
"fieldCaptionInclude" on page 8-77
"fieldCaptionStyle" on page 8-78
"fieldEditWidth" on page 8-80
"fieldEntryInclude" on page 8-81
"fieldExtraScriptInclude" on page 8-82
"fieldInclude" on page 8-83
"fieldMaxLength" on page 8-85
"fieldValueStyle" on page 8-90
"hasOptionList" on page 8-120
"isFieldExcluded" on page 8-164
"isFieldHidden" on page 8-165
"isFieldInfoOnly" on page 8-166
"isFieldMemo" on page 8-167
"isMultiOption" on page 8-176
"isStrictList" on page 8-187
"NotationForRequiredFields" on page 8-220
"optionListKey" on page 8-222
"optionListName" on page 8-223
"optionListResultSet" on page 8-224
"optionListValueInclude" on page 8-226
"optionsAllowPreselect" on page 8-227
"StyleForRequiredFields" on page 8-316

3.6.2.2.2 Field Display Variables This example shows how the `compute_std_field_overrides` include in the *IdcHomeDir*/resources/core/templates/std_page.htm resource file determines if the field currently being generated is hidden, information only, excluded, and/or relocated. This code is looped over during generation of each metadata field on a page:

```
<@dynamichtml compute_std_field_overrides@>
<$isCustomHidden = getValue("#active", fieldName & ":isHidden")$>
<$if isHidden or isCustomHidden$>
    <$isFieldHidden = 1$>
<$else$>
    <$isFieldHidden = ""$>
<$endif$>
<$isCustomInfo = getValue("#active", fieldName & ":isInfoOnly")$>
<$if isInfo or isCustomInfo or isFieldHidden or isInfoOnly$>
    <$isFieldInfoOnly = 1$>
<$else$>
    <$isFieldInfoOnly = ""$>
```

```
<$endif$>
<$isCustomExcluded = getValue("#active", fieldName & ":isExcluded")$>
<$isCustomRelocated = getValue("#active", fieldName & ":isRelocated")$>
<$if isCustomExcluded or (isCustomRelocated and not isRelocated) or isExcluded or
(isFieldHidden and not isFormSubmit)$>
    <$isFieldExcluded = 1$>
<$endif$>
<@end@>
```

3.7 Read-Only Variables

Read-only variables can be used to gather information about the current template, the user who is currently logged in, or other current settings. These variables are read-only and cannot be assigned a value.

- "Template Read-Only Variables" on page 3-18
- "User Read-Only Variables" on page 3-19
- "Content Read-Only Variables" on page 3-19
- "Other Read-Only Variables" on page 19

3.7.1 Template Read-Only Variables

Template-related read-only variables make it possible to create conditional content in a template based on the identity of the template. These pre-defined variables allow you to display the class, file path, name, or type of any template on a Content Server web page. This is particularly useful while you are developing your Web site.

The following read-only variables are related to templates:

"TemplateClass" on page 8-323
"TemplateFilePath" on page 8-324
"TemplateName" on page 8-325
"TemplateType" on page 8-326

3.7.1.1 Template Read-Only Variable Example

In this example, the internal name of the template appears under the Administration link in the left sidebar of all Content Server web pages. To accomplish this change, the pre-defined *TemplateName* variable (shown in bold below) was added to the *pne_nav_admin_links* include that defines the Administration links.

The following is an example of using the *TemplateName* pre-defined variable to display the internal template name on a web page:

```
<$if IsSubAdmin$>
<tr>
    <td>
        <a href="<$HttpCgiPath$>?IdcService=GET_ADMIN_PAGE&Action=
           GetTemplatePage&Page=ADMIN_LINKS"
        OnMouseOver="imgAct('admin')"
        OnMouseOut="imgInact('admin')">
        <img src="<$HttpImagesRoot$>
           <$button_admin_grey_ish_image$>" width="<$navImageWidth$>"
        height="<$navImageHeight$>" name="admin" border="0"
        alt="<$lc("wwProductAdministration", ProductID)$>"></a>
    </td>
        <td>
        <a class=pneHeader href="<$HttpCgiPath$>?IdcService=GET_ADMIN_
```

```
            PAGE&Action=GetTemplatePage&Page=ADMIN_LINKS"
        OnMouseOver="imgAct('admin')"
        OnMouseOut="imgInact('admin')">
        <$lc("wwAdministration")$></a>
    </td>
</tr>
<tr>
    <td colspan=2><font color=#FFFFFF style="Arial"
        size="-1"><$TemplateName$></font></td>
</tr>
<$endif$>
```

3.7.2 User Read-Only Variables

User-related read-only variables make it possible to gather information about the current user.

The following read-only variables are related to users:

"DelimitedUserRoles" on page 8-55
"ExternalUserAccounts" on page 8-74
"ExternalUserRoles" on page 8-75
"UserAccounts" on page 8-339
"UserAddress" on page 8-340
"UserAppRights" on page 8-341
"UserDefaultAccount" on page 8-344
"UserFullName" on page 8-346
"UserName" on page 8-349
"UserRoles" on page 8-351

3.7.3 Content Read-Only Variables

One content-related read-only variable, "SourceID" on page 6-247, makes it possible to retrieve the Content ID of the current dynamic server page.

This variable returns the same value as `ref:dID`. See "Referencing Metadata in Dynamic Server Pages" on page 2-22.

3.7.4 Other Read-Only Variables

The following variable is set only as internal flags, so it can be retrieved but not set directly.

"SafeDir" on page 8-283

3.8 Setable Variables

Setable variables can be set within script or used within a CGI string. For example, the variable "ScriptDebugTrace" can be used as a parameter to a service call to display debug trace information on a page. Setting one of these variables can change the content of the page.

The following is a list of Idoc Script setable variables:

"ClientControlled" on page 8-24
"coreContentOnly" on page 8-27
"getCookie" on page 6-60
"HasLocalCopy" on page 8-119
"IsJava" on page 8-172

"IsSavedQuery" on page 8-185
"IsSoap" on page 8-186
"IsXml" on page 8-195
"isZoneSearchField" on page 6-134
"ScriptDebugTrace" on page 8-284
"ScriptErrorTrace" on page 8-285
"setCookie" on page 6-239

3.8.1 Workflows

Idoc Script includes predefined functions and variables that are used specifically for workflows.

- "Workflow Functions" on page 3-21 perform actions or return results relative to a workflow.
- "Workflow Variables" on page 3-21 enable you to set values for variables related to workflows.

For a detailed description of how workflows are implemented in Content Server, see details the *Oracle Fusion Middleware Application Administrator's Guide for Content Server*.

The following points summarize the use of Idoc Script in workflows:

- Workflow jumps are initiated through the evaluation of Idoc Script that is defined for a particular step event (entry, update, or exit).
- As a revision moves from step to step, the system creates a *companion file* that maintains information about the state of the revision in the workflow. Along with user-defined options, the system also maintains the history of what steps the revision has been to, the last entry time, and the number of times a revision has entered a particular workflow step.
  - Global state information is maintained as the revision moves from step to step.
  - Localized state information is stored with the step and becomes available when a revision is at that step.
- The companion file uses *keys* to keep track of workflow state information. The syntax for a key is:

  ```
  step_name@workflow_name.variable=value
  ```

  For example, the following keys define the value of the entry count and last entry variables for the *Editor* step of a workflow called *Marketing*:

  ```
  Editor@Marketing.entryCount=1
  Editor@Marketing.lastEntryTs={ts '2002-05-28 16:57:00'}
  ```

- All workflow script evaluation occurs inside a database transaction. The result is that any serious errors or aborts that are encountered cause no change to either the database or the companion file. This also means that no Idoc Script workflow function should take more than a negligible amount of time. Consequently, to trigger an external process, an Idoc Script function should be written to execute in a separate thread.

> **Caution:** If you are using Idoc Script or custom components to load workflow information into the local data, keep in mind that there is a risk of data pollution. This is particularly important if you are loading information for a different revision than the current one.

3.8.1.1 Workflow Functions

Workflow functions perform actions or return results relative to a workflow. The following is a list of workflow functions:

"wfAddActionHistoryEvent" on page 6-276
"wfAddUser" on page 6-278
"wfComputeStepUserList" on page 6-279
"wfCurrentGet" on page 6-280
"wfCurrentSet" on page 6-281
"wfCurrentStep" on page 6-282
"wfDisplayCondition" on page 6-283
"wfExit" on page 6-284
"wfGet" on page 6-285
"wfIsFinishedDocConversion" on page 6-287
"wfIsNotifyingUsers" on page 6-288
"wfIsReleasable" on page 6-289
"wfLoadDesign" on page 6-295
"wfNotify" on page 6-298
"wfSet" on page 6-301
"wfSetIsNotifyingUsers" on page 6-302
"wfUpdateMetaData" on page 6-304

3.8.1.2 Workflow Variables

Workflow variables are used to set values related to workflows. The following is a list of workflow variables:

"dWfName" on page 6-42
"dWfStepName" on page 6-43
"entryCount" on page 6-50
"lastEntryTs" on page 6-141
"wfAction" on page 6-275
"wfAdditionalExitCondition" on page 6-277
"wfGetStepTypeLabel" on page 6-286
"wfJumpEntryNotifyOff" on page 6-290
"wfJumpMessage" on page 6-291
"wfJumpName" on page 6-292
"wfJumpReturnStep" on page 6-293
"wfJumpTargetStep" on page 6-294
"wfMailSubject" on page 6-296
"wfMessage" on page 6-297
"wfParentList" on page 6-299
"wfReleaseDocument" on page 6-300
"WfStart" on page 6-303

3.9 Value Variables

A *value* variable is evaluated once at the beginning of a service call and that value is used throughout the service call. The variable is then reevaluated on each new service call. In contrast, a *dynamic* variable is evaluated on each occurrence of the variable. For example, the value variable *isNew* evaluates whether the content item is new or a revision when performing a check in. That evaluation is used throughout the call to the checkin service.

The following is a list of Idoc Script value variables:

"AuthorAddress" on page 8-12

"BrowserVersionNumber" on page 8-17
"CURRENT_DATE" on page 8-28
"CURRENT_ROW" on page 8-29
"DocTypeSelected" on page 6-37
"DocUrl" on page 6-38
"DownloadSuggestedName" on page 8-64
"fileUrl" on page 6-53
"FIRSTREV" on page 6-54
"HasOriginal" on page 6-69
"htmlRefreshTimeout" on page 8-123
"htmlRefreshUrl" on page 8-124
"HttpAbsoluteCgiPath" on page 8-125
"HttpAbsoluteCgiRoot" on page 8-126
"HttpAbsoluteWebRoot" on page 8-127
"HttpAdminCgiPath" on page 8-128
"HttpBrowserFullCgiPath" on page 8-129
"HttpCgiPath" on page 8-130
"HttpCommonRoot" on page 8-131
"HttpEnterpriseCgiPath" on page 8-132
"HttpHelpRoot" on page 8-133
"HttpImagesRoot" on page 8-134
"HttpLayoutRoot" on page 8-135
"HttpRelativeAdminRoot" on page 8-136
"HttpRelativeCgiRoot" on page 8-137
"HttpSystemHelpRoot" on page 8-140
"HttpWebRoot" on page 8-141
"IsEditRev" on page 6-111
"IsFailedConversion" on page 10-55
"IsFailedIndex" on page 6-112
"isNew" on page 8-178
"IsNotLatestRev" on page 6-121
"MSIE" on page 8-215
"NoMatches" on page 6-180
"OneMatch" on page 6-189
"PageParent" on page 8-230
"ResultsTitle" on page 6-207
"StatusCode" on page 6-249
"StatusMessage" on page 6-250
"UseHtmlOrTextHighlightInfo" on page 6-261

4

Master List: Idoc Script by Usage

This chapter groups Idoc Script variables and functions by common usage.

- "Searching" on page 4-17
- "Security" on page 4-18
  - "Internal Security" on page 4-18
  - "External Security" on page 4-19
- "Strings" on page 4-19
- "Templates" on page 4-20
- "Users" on page 4-20
- "Web Server" on page 4-21
- "Workflow" on page 4-21

4.1 Archiver

The following Idoc Script variables are related to the Archiver.

"AllowMatchesInDateCheck" on page 8-6
"ARCHIVE_COLLECTION" on page 8-7
"ArchiverDoLocks" on page 8-8
"ArchiverTouchMonitorInterval" on page 8-9
"ArchiverTraceLockMaintenance" on page 8-10
"ArchiverTraceLocks" on page 8-11
"CollectionLocation" on page 8-25
"IsAutoArchiver" on page 8-154
"LocaleArchiverDateFormat" on page 8-207
"MaxArchiveErrorsAllowed" on page 8-212
"OdbcDateFormat" on page 8-221

4.2 Batch Loader

The following Idoc Script variables are related to the Batch Loader.

"BatchLoaderPath" on page 8-13
"BatchLoaderUserName" on page 8-14
"CleanUp" on page 8-23
"EnableErrorFile" on page 8-70
"MaxErrorsAllowed" on page 8-213

4.3 Clients

The following Idoc Script variables are related to client applications.

"BrowserVersionNumber" on page 8-17
"ClientControlled" on page 8-24
"DownloadApplet" on page 8-63
"HasLocalCopy" on page 8-119
"HeavyClient" on page 8-121
"HttpBrowserFullCgiPath" on page 8-129
"IsMac" on page 8-174
"IsSun" on page 8-188
"IsUploadSockets" on page 6-129
"IsWindows" on page 8-194
"MacSupportsSignedApplets" on page 8-211
"MSIE" on page 8-215

"MultiUpload" on page 8-216
"showDocActions" on page 8-292
"UploadApplet" on page 8-334

4.4 Content Items

The following Idoc Script variables and functions are related to content items.

"AllowAlternateMetaFile" on page 6-9
"AllowPrimaryMetaFile" on page 6-11
"AllowSamePrimaryAlternateExtensions" on page 6-13
"AutoNumberPrefix" on page 6-17
"AutoNumberWidth" on page 6-18
"computeDocUrl" on page 6-22
"computeRenditionUrl" on page 6-23
"CONTENT_LENGTH" on page 6-24
"createAlternateMetaFile" on page 6-27
"createPrimaryMetaFile" on page 6-28
"DocTypeSelected" on page 6-37
"DocUrl" on page 6-38
"DownloadSuggestedName" on page 8-64
"EnableExpirationNotifier" on page 6-46
"ExpireRevisionOnly" on page 6-52
"fileUrl" on page 6-53
"FIRSTREV" on page 6-54
"FormInheritMetaDataFieldList" on page 6-58
"ForceDistinctRevLabels" on page 6-55
"HasLocalCopy" on page 8-119
"HasOriginal" on page 6-69
"HasUrl" on page 6-71
"IsAutoNumber" on page 6-105
"IsCriteriaSubscription" on page 6-110
"IsEditRev" on page 6-111
"IsFailedConversion" on page 10-55
"IsFailedIndex" on page 6-112
"IsFilePresent" on page 6-113
"IsFullTextIndexed" on page 6-115
"IsLocalSearchCollectionID" on page 6-119
"isNew" on page 8-178
"IsNotLatestRev" on page 6-121
"IsNotSyncRev" on page 6-122
"MajorRevSeq" on page 6-163
"MinorRevSeq" on page 6-178
"NotificationIntervalInDays" on page 6-182
"NotificationMaximum" on page 6-183
"NotificationQuery" on page 6-184
"NotifyExtras" on page 6-185
"NotifyTime" on page 6-186
"NumAdditionalRenditions" on page 10-70
"SingleGroup" on page 6-242
"SourceID" on page 6-247

4.5 Content Profiles

The following functions are used to manage Content Profiles.

"dpGet" on page 8-65
"dpPromote" on page 8-66
"dpPromoteRs" on page 8-67
"dpSet" on page 8-68
"DisableContentProfiles" on page 8-57
"getFieldViewValue" on page 8-108
"getRequiredMsg" on page 8-112
"IsDpSubmitErrorFatal" on page 8-158
"loadDocumentProfile" on page 8-204
"utLoadDocumentProfiles" on page 8-355

4.6 Content Server

The following Idoc Script variables are related to the Content Server.

"AccessListPrivilegesGrantedWhenEmpty" on page 8-3
"AdditionalSubscriptionCollatedFieldList" on page 6-4
"AppletChunkSize" on page 6-14
"AppletChunkThreshold" on page 6-15
"ChunkedRequestTrace" on page 8-22
"CookieAuthTimeoutInDays" on page 6-25
"CookieProxyPassword" on page 6-26
"DefaultHtmlRefreshTimeoutInSeconds" on page 6-32
"DefaultPasswordEncoding" on page 6-33
"DisableDBLongIntegerField" on page 6-34
"DisableHttpUploadChunking" on page 6-36
"DisableSharedCacheChecking" on page 8-59
"DoDocNameOrder" on page 6-40
"DownloadApplet" on page 8-63
"FileEncoding" on page 8-92
"getFieldConfigValue" on page 6-62
"HasExternalUsers" on page 6-68
"HttpAbsoluteCgiPath" on page 8-125
"HttpAbsoluteCgiRoot" on page 8-126
"HttpAbsoluteWebRoot" on page 8-127
"HttpAdminCgiPath" on page 8-128
"HttpCgiPath" on page 8-130
"HttpCommonRoot" on page 8-131
"HttpEnterpriseCgiPath" on page 8-132
"HttpHelpRoot" on page 8-133
"HttpImagesRoot" on page 8-134
"HttpLayoutRoot" on page 8-135
"HttpRelativeAdminRoot" on page 8-136
"HttpRelativeCgiRoot" on page 8-137
"HttpSystemHelpRoot" on page 8-140
"IDC_Name" on page 6-82
"IdcCommandFile" on page 6-84
"IdcCommandLog" on page 6-85
"IdcCommandServerHost" on page 6-86
"IdcServerSocketQueueDepth" on page 8-146
"InstanceDescription" on page 6-99
"InstanceMenuLabel" on page 6-100
"IntradocServerHostName" on page 6-102
"IntradocServerPort" on page 6-103
"isComponentEnabled" on page 6-108

"IsContributor" on page 6-109
"IsFormsPresent" on page 6-114
"isLayoutEnabled" on page 6-118
"IsPromptingForLogin" on page 6-124
"IsRequestError" on page 8-183
"IsWebServerPagesOnly" on page 6-132
"isValidateFile" on page 6-131
"JvmCommandLine" on page 8-200
"LmDefaultLayout" on page 6-156
"LmDefaultSkin" on page 6-157
"lmGetLayout" on page 6-158
"lmGetSkin" on page 6-159
"LogMessagePrefix" on page 8-209
"LogName" on page 8-210
"MacSupportsSignedApplets" on page 8-211
"MailServer" on page 6-162
"MaxIntegerAllowed" on page 6-167
"MaxQueryRows" on page 6-168
"MaxRequestThreadCount" on page 6-169
"MaxSizeResourceFileCacheMillions" on page 6-173
"MaxStandardDatabaseResults" on page 6-174
"MinIntegerAllowed" on page 6-176
"MultiUpload" on page 8-216
"NoAutomation" on page 6-179
"PrimaryWorkQueueTimeout" on page 6-193
"ProviderClass" on page 6-195
"ProviderConfig" on page 6-196
"ProviderConnection" on page 6-197
"ProviderType" on page 6-198
"ServiceStartupTimeout" on page 6-238
"ShowCreateExternalMenu" on page 6-240
"SkipMozillaMailToWorkaround" on page 6-243
"SmtpPort" on page 6-244
"StatusCode" on page 6-249
"StatusMessage" on page 6-250
"SubscriptionCollatedFieldList" on page 6-252
"SubscriptionMethod" on page 6-253
"SysAdminAddress" on page 8-318
"UploadApplet" on page 8-334
"UserCacheTimeout" on page 8-343
"UseRevisionCreateDate" on page 6-264
"XMLEncodingMode" on page 6-307

4.7 Conversion

This section describes the Idoc Script variables and functions that are related to various conversion products.

- "Inbound Refinery" on page 4-6
- "PDF Converter" on page 4-6
- "Dynamic Converter" on page 4-7

4.7.1 Inbound Refinery

The following Idoc Script variables are related to Inbound Refinery.

"AllowSecondaryConversion" on page 10-4
"DefaultFilterInputFormat" on page 10-27
"DefaultFilterOutputFormat" on page 10-28
"DefaultGraphicTimeout" on page 10-29
"DefaultNativeTimeout" on page 10-31
"FileEncoding" on page 8-92
"IsAutoQueue" on page 10-52
"IsContentRefineryPresent" on page 10-53
"IsFailedConversion" on page 10-55
"JvmCommandLine" on page 8-200
"MaxConversionCacheSizeInMegs" on page 10-66
"MaxNumRecursiveStepDefinitions" on page 10-67
"PostConversionTimeOut" on page 10-75
"PreconversionPath" on page 10-79
"PreconversionWaitTime" on page 10-80
"QueueDir" on page 10-109
"SecondaryConversionStepName" on page 10-118
"VaultDir" on page 8-358
"VerboseMode" on page 10-125
"WeblayoutDir" on page 8-361

4.7.2 PDF Converter

This section lists the Idoc Script variables and functions that are related to PDF Converter.

- "General Variables" on page 4-6
- "Microsoft Office" on page 4-6
- "InDesign" on page 4-7
- "Other Formats" on page 4-7

4.7.2.1 General Variables

The following Idoc Script variables are related to PDF Converter.

"DefaultPostscriptTimeout" on page 10-32
"DistillerPath" on page 10-36
"NumAdditionalRenditions" on page 10-70
"PostprocessPDFPath" on page 10-76
"PostscriptPrinterName" on page 10-77
"PrinterPortPath" on page 10-83
"ShowHyperlinkBox" on page 10-120

4.7.2.2 Microsoft Office

The following Idoc Script variables are related to the use of Microsoft Office applications with PDF Converter.

"AllowSkippedHyperlinktoCauseIncomplete" on page 10-5
"PowerPointPrintOutput" on page 10-78
"ProcessExcelUrlHyperlinks" on page 10-84
"ProcessPowerpointHyperlinks" on page 10-90
"ProcessVisioInternalHyperlinks" on page 10-91

"ProcessVisioUrlHyperlinks" on page 10-92
"ProcessWordBookMarks" on page 10-93
"ProcessWordComments" on page 10-94
"ProcessWordEmailLinks" on page 10-95
"ProcessWordFootEndNote" on page 10-96
"ProcessWordHeadings" on page 10-97
"ProcessWordHyperlinks" on page 10-98
"ProcessWordToc" on page 10-99
"ProcessWordUrlHyperlinks" on page 10-100

4.7.2.3 InDesign

The following Idoc Script variables are related to the use of Adobe InDesign with PDF Converter.

"ProcessInDesignBookMarks" on page 10-85
"ProcessInDesignHyperlinks" on page 10-86
"ProcessInDesignInternalPDFThumbnail" on page 10-87
"ProcessInDesignLinkUpdate" on page 10-88
"ProcessInDesignPDFCompatibility50" on page 10-89

4.7.2.4 Other Formats

The following Idoc Script variables are related to the use of various native applications with PDF Converter.

"FrameMakerCyclesAllowed" on page 10-42
"FrameMakerTimeout" on page 10-43
"MSPubexePath" on page 10-68
"OpenOfficeListenerPort" on page 10-71
"OpenOfficeListenerURL" on page 10-72
"PageMakerExePath" on page 10-74

4.7.3 Dynamic Converter

The following Idoc Script variables are related to Dynamic Converter.

"ConversionCacheExpirationPeriodInDays" on page 10-20
"DatedCacheIntervalDays" on page 10-21
"DCMaxFileSize" on page 10-22
"dcShowExportLink" on page 10-23
"DCTimeOut" on page 10-24
"DCViewFormat" on page 10-25
"DefaultFilterInputFormat" on page 10-27
"DefaultFilterOutputFormat" on page 10-28
"DefaultHtmlConversion" on page 10-30
"DisableForcedConversions" on page 10-33
"DisableKeepingForcedConversions" on page 10-34
"DisableOptimizedLatestReleaseQuerying" on page 10-35
"DynamicConverterHtmlFlavor" on page 10-38
"EnableOptimizedLatestReleaseQuerying" on page 10-39
"EnableRebuildForcesReconversion" on page 10-40
"ForcedConversionRules" on page 10-41
"HighlightBegin" on page 10-46
"HighlightEnd" on page 10-47
"incDynamicConversionByRule" on page 10-50
"IsDynamicConverterEnabled" on page 10-54
"rule:IsFragmentOnly" on page 10-117

"LatestReleasedTimestampTimeout" on page 10-56
"MaxConversionCacheSizeInMegs" on page 10-66
"NeverTimeoutForcedConversionsTemplateChange" on page 10-69
"TimeoutChangedTemplateForcedConversionsInMins" on page 10-121
"TimeoutMismatchedForcedConversionsInMins" on page 10-122
"TraceConversionCacheCleanup" on page 10-123
"UseCascadingStyleSheets" on page 10-124

4.8 Database

The following Idoc Script variables and functions are related to databases.

"ColumnMapFile" on page 6-21
"DatabasePreserveCase" on page 6-29
"DatabaseWildcards" on page 6-30
"formatDateDatabase" on page 6-57
"ForceJdbcSync" on page 6-56
"IsJdbcLockTrace" on page 6-116
"IsJdbcQueryTrace" on page 6-117
"JdbcConnectionString" on page 6-135
"JdbcDriver" on page 6-136
"JdbcPassword" on page 6-137
"JdbcPasswordEncoding" on page 6-138
"JdbcUser" on page 6-139
"MemoFieldSize" on page 6-175
"MinMemoFieldSize" on page 6-177
"NumConnections" on page 6-188
"UseDatabaseShortIndexName" on page 6-260

4.9 Date and Time

The following Idoc Script variables and functions are related to the formatting and manipulation of dates and times.

"CURRENT_DATE" on page 8-28
"dateCurrent" on page 8-31
"DateOutputFormat" on page 8-32
"formatDate" on page 8-97
"formatDateDatabase" on page 6-57
"formatDateOnly" on page 8-99
"formatDateOnlyDisplay" on page 8-100
"formatDateOnlyFull" on page 8-101
"formatDateWithPattern" on page 8-102
"formatTimeOnly" on page 8-103
"lastEntryTs" on page 6-141
"parseDataEntryDate" on page 8-231
"parseDate" on page 8-232
"parseDateWithPattern" on page 8-234
"SystemDateFormat" on page 8-319
"SystemTimeZone" on page 8-321
"UseFourDigitYear" on page 8-338
"WarnAboutTimeZone" on page 8-359

4.10 Debugging

The following Idoc Script variables and functions are related to debugging.

"ArchiverTraceLocks" on page 8-11
"ArchiverTraceLockMaintenance" on page 8-10
"ChunkedRequestTrace" on page 8-22
"DirectoryLockingLogPath" on page 8-56
"EnableEnvironmentPackager" on page 8-69
"EnableErrorFile" on page 8-70
"ForceJdbcSync" on page 6-56
"getDebugTrace" on page 8-106
"getErrorTrace" on page 8-107
"getFreeMemory" on page 8-109
"getTotalMemory" on page 8-114
"IdcAnalyzeLogDir" on page 8-142
"IndexerTraceCycleListChanges" on page 6-96
"IndexerTraceLockFileChanges" on page 6-97
"IndexerTraceStartStop" on page 6-98
"isActiveTrace" on page 8-153
"IsJava" on page 8-172
"IsJdbcLockTrace" on page 6-116
"IsJdbcQueryTrace" on page 6-117
"IsRequestError" on page 8-183
"IsSoap" on page 8-186
"isVerboseTrace" on page 8-193
"IsXml" on page 8-195
"LogMessagePrefix" on page 8-209
"LogName" on page 8-210
"RequestAuditAdditionalVerboseFieldsList" on page 8-237
"RequestAuditIntervalSeconds1" on page 8-238
"RequestAuditIntervalSeconds2" on page 8-239
"RequestAuditListDepth1" on page 8-240
"RequestAuditListDepth2" on page 8-241
"RequestAuditSubClassificationField" on page 8-242
"SafeDir" on page 8-283
"ScriptErrorTrace" on page 8-285
"SearchCacheTrace" on page 6-225
"SearchDebugLevel" on page 6-227
"ScriptDebugTrace" on page 8-284
"SearchQueryTrace" on page 6-231
"TempDir" on page 8-322
"trace" on page 8-328
"TraceResourceConflict" on page 8-330
"TraceResourceLoad" on page 8-331
"TraceResourceOverride" on page 8-332
"UseRedirectedOutput" on page 8-345
"VerboseMode" on page 10-125

4.11 Directories and Paths

The following Idoc Script variables and functions are related to directories and file paths.

"BinDir" on page 8-15
"CollectionLocation" on page 8-25

"ConfigDir" on page 8-26
"DataDir" on page 8-30
"docRootFilename" on page 8-62
"DownloadSuggestedName" on page 8-64
"HTMLEditorPath" on page 8-122
"HttpAbsoluteCgiPath" on page 8-125
"HttpAbsoluteCgiRoot" on page 8-126
"HttpAbsoluteWebRoot" on page 8-127
"HttpAdminCgiPath" on page 8-128
"HttpBrowserFullCgiPath" on page 8-129
"HttpCgiPath" on page 8-130
"HttpCommonRoot" on page 8-131
"HttpEnterpriseCgiPath" on page 8-132
"HttpHelpRoot" on page 8-133
"HttpImagesRoot" on page 8-134
"HttpLayoutRoot" on page 8-135
"HttpRelativeAdminRoot" on page 8-136
"HttpRelativeCgiRoot" on page 8-137
"HttpRelativeWebRoot" on page 8-138
"HttpServerAddress" on page 8-139
"HttpSystemHelpRoot" on page 8-140
"HttpWebRoot" on page 8-141
"IdcAnalyzeLogDir" on page 8-142
"IdcHomeDir" on page 8-143
"IdcNativeDir" on page 10-48
"IdcResourcesDir" on page 10-49
"IntradocDir" on page 8-152
"isPhysicallySplitDir" on page 8-180
"ProviderDir" on page 8-236
"SafeDir" on page 8-283
"SearchDir" on page 6-228
"SubjectsDir" on page 8-317
"TempDir" on page 8-322
"UserCache" on page 8-342
"VaultDir" on page 8-358
"WebBrowserPath" on page 8-360
"WeblayoutDir" on page 8-361
"WorkflowDir" on page 6-305

4.12 Dynamicdata

The following Idoc Script functions are related to dynamicdata tables.

"ddAppendIndexedColumnResultSet" on page 8-33
"ddAppendResultSet" on page 8-35
"ddApplyTableSortToResultSet" on page 8-36
"ddGetFieldList" on page 8-37
"ddIncludePreserveValues" on page 8-38
"ddLoadIndexedColumnResultSet" on page 8-40
"ddLoadResultSet" on page 8-41
"ddMergeIndexedColumnResultSet" on page 8-42
"ddMergeResultSet" on page 8-44
"ddMergeUsingIndexedKey" on page 8-46
"ddSetLocal" on page 8-48
"ddSetLocalByColumnsFromFirstRow" on page 8-49

"ddSetLocalByColumnsFromFirstRowIndexed" on page 8-50
"ddSetLocalEmpty" on page 8-51
"ddSetLocalEmptyByColumns" on page 8-52

4.13 Field Display

The following Idoc Script variables and functions are related to the display of metadata fields on Content Server web pages.

"addEmptyOption" on page 8-4
"captionEntryWidth" on page 8-20
"captionFieldWidth" on page 8-21
"defaultFieldInclude" on page 8-53
"defaultOptionListScript" on page 8-54
"fieldCaption" on page 8-76
"fieldCaptionInclude" on page 8-77
"fieldCaptionStyle" on page 8-78
"fieldDefault" on page 8-79
"fieldEditWidth" on page 8-80
"fieldEntryInclude" on page 8-81
"fieldExtraScriptInclude" on page 8-82
"fieldInclude" on page 8-83
"fieldIsOptionList" on page 8-84
"fieldMaxLength" on page 8-85
"fieldName" on page 8-86
"fieldOptionListType" on page 8-87
"fieldType" on page 8-88
"fieldValue" on page 8-89
"fieldValueStyle" on page 8-90
"fieldWidth" on page 8-91
"ForceDocTypeChoice" on page 8-93
"ForceSecurityGroupChoice" on page 8-96
"hasOptionList" on page 8-120
"isExcluded" on page 8-161
"isFieldExcluded" on page 8-164
"isFieldHidden" on page 8-165
"isFieldInfoOnly" on page 8-166
"isFieldMemo" on page 8-167
"isHidden" on page 8-169
"isInfoOnly" on page 8-171
"isMultiOption" on page 8-176
"IsOverrideFormat" on page 8-179
"isRelocated" on page 8-182
"isRequired" on page 8-184
"isStrictList" on page 8-187
"maxLength" on page 8-214
"MemoFieldSize" on page 6-175
"MinMemoFieldSize" on page 6-177
"NotationForRequiredFields" on page 8-220
"optList" on page 8-228
"optionListKey" on page 8-222
"optionListName" on page 8-223
"optionListResultSet" on page 8-224
"optionListScript" on page 8-225
"optionListValueInclude" on page 8-226

"optionsAllowPreselect" on page 8-227
"requiredMsg" on page 8-243
"ShowOnlyKnownAccounts" on page 6-241
"ShowRevisionsColumn" on page 8-293
"StyleForRequiredFields" on page 8-316
"valueStyle" on page 8-357

4.14 Folders and WebDAV

The following Idoc Script variables are related to Folders and WebDAV installation and use.

"CollectionDeleteEnabled" on page 10-6
"CollectionHiddenMeta" on page 10-7
"CollectionInhibitUpdateMeta" on page 10-8
"CollectionMeta" on page 10-9
"CollectionMoveEnabled" on page 10-10
"CollectionReadOnlyMarkedFolders" on page 10-11
"CollectionReadOnlyMeta" on page 10-12
"CollectionReleasedOnly" on page 10-13
"CollectionSecurityReadOnly" on page 10-14
"CollectionTrashDeleteDate" on page 10-15
"CollectionTrashDeleteLocation" on page 10-16
"CollectionTrashDeleteOldName" on page 10-17
"CollectionTrashDeleter" on page 10-18
"CollectionWebDAVServer" on page 10-19
"InitialColID" on page 10-51

4.15 HTML Preview

The following Idoc Script variables are related to HTML Preview.

"DtmPort" on page 10-37
"PreviewOutputExtension" on page 10-81
"PreviewPath" on page 10-82

4.16 Idoc Script

The following Idoc Script functions and keywords are not related to any specific functionality.

"break" on page 8-16
"cacheInclude" on page 8-19
"docLoadResourceIncludes" on page 8-60
"eval" on page 8-71
"executeService" on page 8-73
"forceExpire" on page 8-94
"getTextFile" on page 8-113
"getUserValue" on page 8-115
"getValue" on page 8-116
"inc" on page 8-148
"incGlobal" on page 8-149
"incTemplate" on page 8-151
"isFalse" on page 8-163
"isTrue" on page 8-189
"setExpires" on page 8-287

"setHttpHeader" on page 8-288
"setMaxAge" on page 8-289
"setResourceInclude" on page 8-290

4.16.1 Keywords

"c" on page 8-18
"exec" on page 8-72
"include" on page 8-150

4.17 Indexing

The following Idoc Script variables are related to indexing.

"AdditionalIndexBuildParams" on page 6-2
"AllowConcurrentUpdate" on page 6-10
"DateOutputFormat" on page 8-32
"IndexableRenditions" on page 6-89
"IndexerAutoWorkInterval" on page 6-90
"IndexerLargeFileSize" on page 6-91
"IndexerRebuildStagingManualCopy" on page 6-92
"IndexerRebuildStagingPath" on page 6-93
"indexerSetCollectionValue" on page 6-94
"IndexerTouchMonitorInterval" on page 6-95
"IndexerTraceCycleListChanges" on page 6-96
"IndexerTraceLockFileChanges" on page 6-97
"IndexerTraceStartStop" on page 6-98
"IsAllowIndexRenditions" on page 6-104
"IsFailedIndex" on page 6-112
"IsFullTextIndexed" on page 6-115
"LkDisableOnRebuild" on page 6-147
"LkDisableReferenceInfo" on page 6-148
"LkDisallowConversionFormats" on page 6-149
"LkIsSecureSearch" on page 6-150
"LkReExtractOnRebuild" on page 6-151
"LkRefreshBatchSize" on page 6-152
"LkRefreshErrorsAllowed" on page 6-154
"LkRefreshErrorPercent" on page 6-153
"LkRefreshErrorThreshold" on page 6-155
"MaxCollectionSize" on page 6-164
"MaxDocIndexErrors" on page 6-165
"TimeoutPerOneMegInSec" on page 6-255

4.18 JSP

The following Idoc Script variables are related to JSP support.

"IsJspServerEnabled" on page 8-173
"JspAdminQuery" on page 8-197
"JspDefaultIndexPage" on page 8-198
"JspEnabledGroups" on page 8-199

4.19 Localization

The following Idoc Script variables and functions are related to localization.

"DefaultFilterInputFormat" on page 10-27
"DefaultFilterOutputFormat" on page 10-28
"FileEncoding" on page 8-92
"lc" on page 8-201
"IdcLocale" on page 8-144
"lcCaption" on page 8-202
"LocaleArchiverDateFormat" on page 8-207
"rptDisplayMapValue" on page 8-244
"SystemDateFormat" on page 8-319
"SystemLocale" on page 8-320
"SystemTimeZone" on page 8-321
"UserLanguageID" on page 8-347
"UserLocaleId" on page 8-348
"WarnAboutTimeZone" on page 8-359

4.20 Page Display

The following Idoc Script variables and functions are related to the display of Content Server web pages.

"abortToErrorPage" on page 8-2
"AfterLogin" on page 8-5
"coreContentOnly" on page 8-27
"docLoadResourceIncludes" on page 8-60
"generateUniqueId" on page 8-105
"getHelpPage" on page 8-110
"getOptionListSize" on page 8-111
"htmlRefreshTimeout" on page 8-123
"htmlRefreshUrl" on page 8-124
"isCheckin" on page 8-155
"IsCurrentNav" on page 8-156
"isDocPage" on page 8-157
"IsDynamic" on page 8-159
"isEditMode" on page 8-160
"isFormSubmit" on page 8-168
"isInfo" on page 8-170
"IsMaxRows" on page 8-175
"IsMultiPage" on page 8-177
"isNew" on page 8-178
"isQuery" on page 8-181
"IsSavedQuery" on page 8-185
"isUpdate" on page 8-190
"isUploadFieldScript" on page 8-191
"localPageType" on page 8-208
"noInfoColumn" on page 8-217
"noMCPrefill" on page 8-218
"noRevisionsColumn" on page 8-219
"PageParent" on page 8-230
"pneNavigation" on page 8-235
"ResultsTitle" on page 6-207
"showDocActions" on page 8-292
"showSearchResultsCheckoutColumn" on page 8-294
"StdPageWidth" on page 8-295
"TemplateClass" on page 8-323
"TemplateFilePath" on page 8-324

"TemplateName" on page 8-325
"TemplateType" on page 8-326
"UseBellevueLook" on page 8-337
"UseXpedioLook" on page 8-352

4.21 Personalization

The following Idoc Script functions are related to user topic (personalization) files.

"utGetValue" on page 8-353
"utLoad" on page 8-354
"utLoadDocumentProfiles" on page 8-355
"utLoadResultSet" on page 8-356

4.22 Pop-Up Calendar

The following configuration variables are used with Pop-Up Calendar.

"pucDefaultTime" on page 10-101
"pucEnableOnSearch" on page 10-102
"pucHideTimeInDateFields" on page 10-103
"pucUseMonthLocale" on page 10-104
"pucUseWeekdayLocale" on page 10-105
"pucXOffset" on page 10-106
"pucYOffset" on page 10-107
"pucZIndex" on page 10-108

4.23 Record Manager and Retention Manager

The following configuration variables are used with Record Manager and Retention Manager.

"AllowRetentionPeriodWithoutCutoff" on page 10-3
"HideVitalReview" on page 10-45
"RecordsManagementNumberOverwriteOnDelete" on page 10-111
"RecordsManagementDenyAuthorFreePassOnRMSecurity" on page 10-110
"RmaAddDocWhereClauseForScreening" on page 10-112
"RMAHideExternalFieldsFromCheckInUpdate" on page 10-113
"RMAHideExternalFieldsFromSearchInfo" on page 10-114
"RmaNotifyDispReviewerAndCatAuthor" on page 10-115
"RmaNotifyReviewerAndAlternateReviewer" on page 10-116
"ShowContentForStorageBrowse" on page 10-119

4.24 ResultSets

The following Idoc Script variables and functions are related to ResultSets.

"CURRENT_ROW" on page 8-29
"getValue" on page 8-116
"loadDocMetaDefinition" on page 8-203
"loadSearchOperatorTables" on page 8-205
"loadUserMetaDefinition" on page 8-206
"rs" on page 8-245
"rsAddFields" on page 8-246
"rsAddFieldsWithDefaults" on page 8-247
"rsAddRowCountColumn" on page 8-249

"rsAppend" on page 8-250
"rsAddFields" on page 8-246
"rsAddRowCountColumn" on page 8-249
"rsAppend" on page 8-250
"rsAppendNewRow" on page 8-251
"rsAppendRowvalues" on page 8-252
"rsCopyFiltered" on page 8-253
"rsCreateReference" on page 8-254
"rsCreateResultSet" on page 8-255
"rsDeleteRow" on page 8-256
"rsDocInfoRowAllowDisclosure" on page 8-257
"rsExists" on page 8-258
"rsFieldByIndex" on page 8-259
"rsFieldExists" on page 8-260
"rsFindRowPrimary" on page 8-261
"rsFirst" on page 8-262
"rsInsertNewRow" on page 8-263
"rsIsRowPresent" on page 8-264
"rsLoopInclude" on page 8-265\
"rsLoopSingleRowInclude" on page 8-266
"rsMakeFromList" on page 8-267
"rsMakeFromString" on page 8-269
"rsMerge" on page 8-271
"rsMergeDelete" on page 8-272
"rsMergeReplaceOnly" on page 8-273
"rsNext" on page 8-274
"rsNumFields" on page 8-275
"rsNumRows" on page 8-276
"rsRemove" on page 8-277
"rsRename" on page 8-278
"rsRenameField" on page 8-279
"rsSetRow" on page 8-280
"rsSort" on page 8-281
"rsSortTree" on page 8-282
"setContentType" on page 8-286
"setValue" on page 8-291
"utLoadResultSet" on page 8-356

4.25 Schema

The following Idoc Script functions are related to schemas.

4.25.1 Schema publish settings

Schema are 'republished' whenever a change occurs that may affect the relationship between the parts of the schema. These settings relate to publishing factors.

"clearSchemaData" on page 6-20
"EnableSchemaPublish" on page 6-48
"getFieldViewDisplayValue" on page 6-63
"getParentValue" on page 6-64
"getViewValue" on page 6-65
"getViewValueResultSet" on page 6-66
"jsFilename" on page 6-140
"loadSchemaData" on page 6-161

"SchemaAllowRecursiveFileMove" on page 6-208
"SchemaBuildDigestFile" on page 6-209
"SchemaPublishCheckInterval" on page 6-210
"SchemaPublishInterval" on page 6-211
"SchemaPublishMaximumInterval" on page 6-212
"SchemaPublishMinimumInterval" on page 6-213
"SchemaPublishRenameRetryInitialInterval" on page 6-214
"SchemaPublishRenameRetryIterations" on page 6-215
"SchemaPublishRetryInterval" on page 6-216
"SchemaPublishRetryIntervalMax" on page 6-217
"SchemaVerifyPublishing" on page 6-218
"SchemaWin32RenameDelayInterval" on page 6-219

4.26 Searching

The following Idoc Script variables and functions are related to searching.

"AllowForceDelete" on page 10-2
"CachedResultRowCount" on page 6-19
"EnableDocumentHighlight" on page 6-45
"HasSiteStudio" on page 10-44
"indexerSetCollectionValue" on page 6-94
"IsAutoSearch" on page 6-106
"IsCurrentNav" on page 8-156
"IsFullTextIndexed" on page 6-115
"IsLocalSearchCollectionID" on page 6-119
"IsMultiPage" on page 8-177
"IsSavedQuery" on page 8-185
"isZoneSearchField" on page 6-134
"loadCollectionInfo" on page 6-160
"MaxHitCount" on page 6-166
"MaxResults" on page 6-170
"MaxSavedSearchResults" on page 6-171
"MaxSearchConnections" on page 6-172
"noInfoColumn" on page 8-217
"NoMatches" on page 6-180
"noRevisionsColumn" on page 8-219
"NoSearchConnectionCloseOnDocChange" on page 6-181
"OneMatch" on page 6-189
"QUERY_STRING" on page 6-199
"regexMatches" on page 6-201
"regexReplaceAll" on page 6-202
"regexReplaceFirst" on page 6-203
"ResultsTitle" on page 6-207
"SearchCacheCleanerInterval" on page 6-221
"SearchCacheHardLimit" on page 6-222
"SearchCacheMaxItemAgeInMinutes" on page 6-223
"SearchCacheSoftLimit" on page 6-224
"SearchCacheTrace" on page 6-225
"SearchConnectionWaitTimeout" on page 6-226
"SearchDebugLevel" on page 6-227
"SearchDir" on page 6-228
"SearchIndexerEngineName" on page 6-229
"SearchQueryFormat" on page 6-230
"SearchQueryTrace" on page 6-231

"showDocActions" on page 8-292
"showSearchResultsCheckoutColumn" on page 8-294
"UniversalSearchAllowFullText" on page 6-257
"UseHtmlOrTextHighlightInfo" on page 6-261
"UseSearchCache" on page 6-270
"UseXmlUrl" on page 6-273

4.27 Security

This section lists the Idoc Script variables and functions that are related to security.

- "Internal Security" on page 4-18
- "External Security" on page 4-19

4.27.1 Internal Security

The following Idoc Script variables and functions are related to internal security.

"AllowAllRedirectUrls" on page 6-8
"AllowCheckin" on page 6-6
"AllowCheckout" on page 6-7
"AdminAtLeastOneGroup" on page 6-5
"AuthorDelete" on page 6-16
"Default Accounts" on page 6-31
"DelimitedUserRoles" on page 8-55
"docUrlAllowDisclosure" on page 6-39
"EmptyAccountCheckinAllowed" on page 6-44
"encodeHtml" on page 6-49
"ExclusiveCheckout" on page 6-51
"GetCopyAccess" on page 6-61
"hasAppRights" on page 6-67
"HasPredefinedAccounts" on page 6-70
"HtmlDataInputFilterLevel" on page 6-72
"rsDocInfoRowAllowDisclosure" on page 8-257
"IsCheckinPreAuthed" on page 6-107
"IsLoggedIn" on page 6-120
"IsSubAdmin" on page 6-127
"IsSysManager" on page 6-128
"isUserOverrideSet" on page 6-130
"SelfRegisteredAccounts" on page 6-232
"SelfRegisteredRoles" on page 6-233
"ShowOnlyKnownAccounts" on page 6-241
"stdSecurityCheck" on page 6-251
"UseAccounts" on page 6-258
"UserAccounts" on page 8-339
"UserAppRights" on page 8-341
"UserDefaultAccount" on page 8-344
"UserFilteredSubjects" on page 6-265
"userHasAccessToAccount" on page 6-266
"userHasGroupPrivilege" on page 6-267
"userHasRole" on page 6-268
"UserIsAdmin" on page 6-269
"UserRoles" on page 8-351

4.27.2 External Security

The following Idoc Script variables are related to external security (Web server and NTLM security).

"DomainControllerName" on page 6-41
"ExternalUserAccounts" on page 8-74
"ExternalUserRoles" on page 8-75
"HasExternalUsers" on page 6-68
"IntradocRealm" on page 6-101
"IsPrimaryUserBase" on page 6-123
"IsPromptingForLogin" on page 6-124
"IsUploadSockets" on page 6-129
"LdapAdminDN" on page 6-142
"LdapAdminPassword" on page 6-143
"LdapPort" on page 6-144
"LdapServer" on page 6-145
"LdapSuffix" on page 6-146
"NtlmSecurityEnabled" on page 6-187
"Priority" on page 6-194
"ProviderClass" on page 6-195
"ProviderConfig" on page 6-196
"ProviderConnection" on page 6-197
"ProviderType" on page 6-198
"RedirectHostsFilter" on page 6-200
"SocketHostAddressSecurityFilter" on page 6-245
"SocketHostNameSecurityFilter" on page 6-246
"SourcePath" on page 6-248
"UseAdsi" on page 6-259
"UseNetscape" on page 6-262
"UseNtlm" on page 6-263
"UseSecureLdap" on page 6-271
"UseSSL" on page 6-272
"WebServerAuthOnly" on page 6-274

4.28 Strings

The following Idoc Script variables and functions are related to strings.

"js" on page 8-196
"strCenterPad" on page 8-296
"strCommaAppendNoDuplicates" on page 8-297
"strConfine" on page 8-298
"strEquals" on page 8-300
"strEqualsIgnoreCase" on page 8-301
"strGenerateRandom" on page 8-302
"strIndexOf" on page 8-303
"strLeftFill" on page 8-304
"strLeftPad" on page 8-305
"strLength" on page 8-306
"strLower" on page 8-307
"StrConfineOverflowChars" on page 8-299
"strRemoveWs" on page 8-308
"strReplace" on page 8-309
"strReplaceIgnoreCase" on page 8-310
"strRightFill" on page 8-311

"strRightPad" on page 8-312
"strSubstring" on page 8-313
"strTrimWs" on page 8-314
"strUpper" on page 8-315
"toInteger" on page 8-327
"url" on page 8-335
"urlEscape7Bit" on page 8-336
"xml" on page 8-362

4.29 Templates

The following Idoc Script variables are related to templates.

"TemplateClass" on page 8-323
"TemplateFilePath" on page 8-324
"TemplateName" on page 8-325
"TemplateType" on page 8-326

4.30 Users

The following Idoc Script variables and functions are related to users.

"AdminAtLeastOneGroup" on page 6-5
"AllowCheckin" on page 6-6
"AllowCheckout" on page 6-7
"AuthorAddress" on page 8-12
"Default Accounts" on page 6-31
"DelimitedUserRoles" on page 8-55
"ExclusiveCheckout" on page 6-51
"ExternalUserAccounts" on page 8-74
"ExternalUserRoles" on page 8-75
"getUserValue" on page 8-115
"getValueForSpecifiedUser" on page 8-118
"HasPredefinedAccounts" on page 6-70
"IsCheckinPreAuthed" on page 6-107
"IsExternalUser" on page 8-162
"IsLoggedIn" on page 6-120
"IsSubAdmin" on page 6-127
"IsSysManager" on page 6-128
"IsUserEmailPresent" on page 8-192
"isUserOverrideSet" on page 6-130
"SysAdminAddress" on page 8-318
"UserAccounts" on page 8-339
"UserAddress" on page 8-340
"UserAppRights" on page 8-341
"UserCacheTimeout" on page 8-343
"UserDefaultAccount" on page 8-344
"UserFullName" on page 8-346
"userHasAccessToAccount" on page 6-266
"userHasGroupPrivilege" on page 6-267
"userHasRole" on page 6-268
"UserIsAdmin" on page 6-269
"UserName" on page 8-349
"UserRoles" on page 8-351
"wfAddUser" on page 6-278

4.31 Web Server

The following Idoc Script variables are related to web servers.

"DomainControllerName" on page 6-41
"GATEWAY_INTERFACE" on page 6-59
"GATEWAY_INTERFACE" on page 6-59
"getCookie" on page 6-60
"HttpAbsoluteWebRoot" on page 8-127
"HTTP_ACCEPT" on page 6-73
"HTTP_ACCEPT_ENCODING" on page 6-74
"HTTP_ACCEPT_LANGUAGE" on page 6-75
"HTTP_COOKIE" on page 6-76
"HTTP_HOST" on page 6-77
"HttpIgnoreWebServerInternalPortNumber" on page 6-81
"HTTP_INTERNETUSER" on page 6-78
"HTTP_REFERER" on page 6-79
"HttpRelativeWebRoot" on page 8-138
"HttpServerAddress" on page 8-139
"HTTP_USER_AGENT" on page 6-80
"IdcAuthExtraRequestParams" on page 6-83
"IdcHttpHeaderVariables" on page 6-88
"IntradocRealm" on page 6-101
"PATH_INFO" on page 6-190
"QUERY_STRING" on page 6-199
"RedirectHostsFilter" on page 6-200
"REMOTE_ADDR" on page 6-204
"REMOTE_HOST" on page 6-205
"REQUEST_METHOD" on page 6-206
"SCRIPT_NAME" on page 6-220
"SERVER_NAME" on page 6-234
"SERVER_PORT" on page 6-235
"SERVER_PROTOCOL" on page 6-236
"SERVER_SOFTWARE" on page 6-237
"setCookie" on page 6-239
"SocketHostAddressSecurityFilter" on page 6-245
"SocketHostNameSecurityFilter" on page 6-246
"UseSSL" on page 6-272
"WebServerAuthOnly" on page 6-274

4.32 Workflow

The following Idoc Script variables and functions are related to workflows.

4.32.1 Configuration Variables

"IsSavedWfCompanionFile" on page 6-126
"PrimaryWorkQueueTimeout" on page 6-193
"WorkflowDir" on page 6-305
"WorkflowIntervalHours" on page 6-306

4.32.2 Global Functions

"getValueForSpecifiedUser" on page 8-118

4.32.3 Workflow Functions

"wfAddActionHistoryEvent" on page 6-276
"wfAddUser" on page 6-278
"wfComputeStepUserList" on page 6-279
"wfCurrentGet" on page 6-280
"wfCurrentSet" on page 6-281
"wfCurrentStep" on page 6-282
"wfDisplayCondition" on page 6-283
"wfExit" on page 6-284
"wfGet" on page 6-285
"wfGetStepTypeLabel" on page 6-286
"wfIsFinishedDocConversion" on page 6-287
"wfIsNotifyingUsers" on page 6-288
"wfIsReleasable" on page 6-289
"wfLoadDesign" on page 6-295
"wfNotify" on page 6-298
"wfReleaseDocument" on page 6-300
"wfSet" on page 6-301
"wfSetIsNotifyingUsers" on page 6-302
"wfUpdateMetaData" on page 6-304

4.32.4 Other Variables

"AllowReview" on page 6-12
"dWfName" on page 6-42
"dWfStepName" on page 6-43
"entryCount" on page 6-50
"IsEditRev" on page 6-111
"isRepromptLogin" on page 6-125
"IsWorkflow" on page 6-133
"lastEntryTs" on page 6-141
"SingleGroup" on page 6-242
"wfAction" on page 6-275
"wfAdditionalExitCondition" on page 6-277
"wfJumpEntryNotifyOff" on page 6-290
"wfJumpMessage" on page 6-291
"wfJumpName" on page 6-292
"wfJumpReturnStep" on page 6-293
"wfJumpTargetStep" on page 6-294
"wfMailSubject" on page 6-296
"wfMessage" on page 6-297
"wfParentList" on page 6-299
"WfStart" on page 6-303

5

Core Functions by Type and Usage

This chapter groups Idoc Script variables and functions that are used for core processing functions such as searching, indexing, and workflow implementation.

This list describes the different core processing functions by *type*:

- "Conditional Dynamic Variables" on page 5-1
- "Configuration Variables" on page 5-2
- "Environment Variables" on page 5-5
- "Global Functions" on page 5-5
- "Setable Variables" on page 5-6
- "Value Variables" on page 5-7

This list divides the different variables by *usage*:

- "Content Items" on page 5-8
- "Content Server" on page 5-8
- "Database" on page 5-9
- "Indexing" on page 5-10
- "Schema" on page 5-10
- "Searching" on page 5-11
- "Security" on page 5-12
- "Web Server" on page 5-13
- "Workflow" on page 5-14

5.1 Conditional Dynamic Variables

Some dynamic variables are conditional and can only be used within a conditional statement such as *if*, *while*, *elseif*, or *loop*.

The following is a list of Idoc Script conditional dynamic variables that are used in core Content Server functionality. For a list of conditional dynamic variables and details about their use, see "Conditional Dynamic Variables" on page 3-1.

"AllowReview" on page 6-12
"EmptyAccountCheckinAllowed" on page 6-44
"HasUrl" on page 6-71
"isComponentEnabled" on page 6-108
"IsContributor" on page 6-109

"IsCriteriaSubscription" on page 6-110
"IsFilePresent" on page 6-113
"IsFullTextIndexed" on page 6-115
"IsLocalSearchCollectionID" on page 6-119
"IsNotSyncRev" on page 6-122
"IsPromptingForLogin" on page 6-124
"IsWorkflow" on page 6-133
"SingleGroup" on page 6-242
"UseXmlUrl" on page 6-273

5.2 Configuration Variables

Configuration variables are predefined Idoc Script variables that are generally used as settings in configuration files. For a list of configuration variables and details about their use, see "Configuration Variables" on page 3-2.

This section lists the configuration variables used in core Content Server functions by the configuration file (typically *.cfg* or *.hda*) in which they are most commonly located. Your actual application will vary depending on the criteria and selections made during installation.

5.2.1 *DomainHome*/ucm/*product_id*/bin/intradoc.cfg

The following configuration variables are located in the *DomainHome*/ucm/*product_id*/bin/intradoc.cfg file. These variables define directory paths and other global settings for the Content Server.

"AllowConcurrentUpdate" on page 6-10
"IdcCommandFile" on page 6-84
"IdcCommandLog" on page 6-85
"IdcCommandUser" on page 6-87

5.2.2 *IntradocDir*/config/config.cfg

The following configuration variables are located in the *IntradocDir*/config/config.cfg file. These variables define most of the configuration settings for the Content Server.

"AdditionalIndexBuildParams" on page 6-2
"AdditionalSubscriptionCollatedFieldList" on page 6-4
"AllowAllRedirectUrls" on page 6-8
"AllowAlternateMetaFile" on page 6-9
"AllowPrimaryMetaFile" on page 6-11
"AllowSamePrimaryAlternateExtensions" on page 6-13
"AppletChunkSize" on page 6-14
"AppletChunkThreshold" on page 6-15
"AuthorDelete" on page 6-16
"AutoNumberPrefix" on page 6-17
"AutoNumberWidth" on page 6-18
"CachedResultRowCount" on page 6-19
"ColumnMapFile" on page 6-21
"CookieAuthTimeoutInDays" on page 6-25
"CookieProxyPassword" on page 6-26
"DatabasePreserveCase" on page 6-29
"DatabaseWildcards" on page 6-30
"DefaultHtmlRefreshTimeoutInSeconds" on page 6-32
"DefaultPasswordEncoding" on page 6-33

"DisableDBLongIntegerField" on page 6-34
"DisableHttpUploadChunking" on page 6-36
"DoDocNameOrder" on page 6-40
"DomainControllerName" on page 6-41
"EnableDocumentHighlight" on page 6-45
"EnableExpirationNotifier" on page 6-46
"EnableSchemaPublish" on page 6-48
"ExpireRevisionOnly" on page 6-52
"ForceDistinctRevLabels" on page 6-55
"FormInheritMetaDataFieldList" on page 6-58
"GetCopyAccess" on page 6-61
"HasExternalUsers" on page 6-68
"HtmlDataInputFilterLevel" on page 6-72
"IdcAuthExtraRequestParams" on page 6-83
"IdcCommandServerHost" on page 6-86
"IdcHttpHeaderVariables" on page 6-88
"IDC_Name" on page 6-82
"IndexableRenditions" on page 6-89
"IndexerAutoWorkInterval" on page 6-90
"IndexerLargeFileSize" on page 6-91
"IndexerRebuildStagingManualCopy" on page 6-92
"IndexerRebuildStagingPath" on page 6-93
"IndexerTouchMonitorInterval" on page 6-95
"InstanceDescription" on page 6-99
"InstanceMenuLabel" on page 6-100
"IntradocRealm" on page 6-101
"IntradocServerHostName" on page 6-102
"IntradocServerPort" on page 6-103
"IsAllowIndexRenditions" on page 6-104
"IsAutoNumber" on page 6-105
"IsAutoSearch" on page 6-106
"IsWebServerPagesOnly" on page 6-132
"IsFormsPresent" on page 6-114
"IsSavedWfCompanionFile" on page 6-126
"JdbcConnectionString" on page 6-135
"JdbcDriver" on page 6-136
"JdbcPassword" on page 6-137
"JdbcPasswordEncoding" on page 6-138
"JdbcUser" on page 6-139
"LmDefaultLayout" on page 6-156
"LmDefaultSkin" on page 6-157
"MailServer" on page 6-162
"MajorRevSeq" on page 6-163
"MaxCollectionSize" on page 6-164
"MaxDocIndexErrors" on page 6-165
"MaxHitCount" on page 6-166
"MaxIntegerAllowed" on page 6-167
"MaxQueryRows" on page 6-168
"MaxRequestThreadCount" on page 6-169
"MaxResults" on page 6-170
"MaxSavedSearchResults" on page 6-171
"MaxSearchConnections" on page 6-172
"MaxSizeResourceFileCacheMillions" on page 6-173
"MinIntegerAllowed" on page 6-176
"MinorRevSeq" on page 6-178

"NoAutomation" on page 6-179
"NoSearchConnectionCloseOnDocChange" on page 6-181
"NotificationIntervalInDays" on page 6-182
"NotificationMaximum" on page 6-183
"NotificationQuery" on page 6-184
"NotifyExtras" on page 6-185
"NotifyTime" on page 6-186
"NtlmSecurityEnabled" on page 6-187
"NumConnections" on page 188
"PrimaryWorkQueueTimeout" on page 6-193
"RedirectHostsFilter" on page 6-200
"SchemaAllowRecursiveFileMove" on page 6-208
"SchemaBuildDigestFile" on page 6-209
"SchemaPublishCheckInterval" on page 6-210
"SchemaPublishInterval" on page 6-211
"SchemaPublishMaximumInterval" on page 6-212
"SchemaPublishMinimumInterval" on page 6-213
"SchemaPublishRenameRetryInitialInterval" on page 6-214
"SchemaPublishRetryInterval" on page 6-216
"SchemaPublishRetryIntervalMax" on page 6-217
"SchemaPublishRenameRetryIterations" on page 6-215
"SchemaVerifyPublishing" on page 6-218
"SchemaWin32RenameDelayInterval" on page 6-219
"SearchCacheCleanerInterval" on page 6-221
"SearchCacheHardLimit" on page 6-222
"SearchCacheMaxItemAgeInMinutes" on page 6-223
"SearchCacheSoftLimit" on page 6-224
"SearchConnectionWaitTimeout" on page 6-226
"SearchIndexerEngineName" on page 6-229
"SearchQueryFormat" on page 6-230
"SelfRegisteredAccounts" on page 6-232
"SelfRegisteredRoles" on page 6-233
"ServiceStartupTimeout" on page 6-238
"ShowCreateExternalMenu" on page 6-240
"SkipMozillaMailToWorkaround" on page 6-243
"SmtpPort" on page 6-244
"SocketHostAddressSecurityFilter" on page 6-245
"SocketHostNameSecurityFilter" on page 6-246
"SubscriptionMethod" on page 6-253
"SubscriptionCollatedFieldList" on page 6-252
"TimeoutPerOneMegInSec" on page 6-255
"UniversalSearchAllowFullText" on page 6-257
"UseAccounts" on page 6-258
"UseAdsi" on page 6-259
"UseNtlm" on page 6-263
"UseRevisionCreateDate" on page 6-264
"UserFilteredSubjects" on page 6-265
"UseSearchCache" on page 6-270
"UseSSL" on page 6-272
"WebServerAuthOnly" on page 6-274
"WorkflowIntervalHours" on page 6-306
"XMLEncodingMode" on page 6-307

5.2.3 *DomainDir*/ucm/*short_product_id*/data/providers/*provider_name*/provider.hda

The following configuration variables are located in the provider.hda file in the *DomainDir*/ucm/*short_product_id*/data/providers/*provider_name*/ directory. These variables define settings for Content Server providers.

"IsPrimaryUserBase" on page 6-123
"LdapAdminDN" on page 6-142
"LdapAdminPassword" on page 6-143
"LdapPort" on page 6-144
"LdapServer" on page 6-145
"LdapSuffix" on page 6-146
"NumConnections" on page 6-188
"Priority" on page 6-194
"ProviderClass" on page 6-195
"ProviderConfig" on page 6-196
"ProviderConnection" on page 6-197
"ProviderType" on page 6-198
"SourcePath" on page 6-248
"UseNetscape" on page 6-262
"UseSecureLdap" on page 6-271

5.3 Environment Variables

Web server variables are the CGI environment variables that are set when the server executes the gateway program. For a list of conditional dynamic variables and details about their use, see "Environment Variables" on page 3-11.

The following is a list of web server variables used in core Content Server functionality:

"CONTENT_LENGTH" on page 6-24
"GATEWAY_INTERFACE" on page 6-59
"HTTP_ACCEPT" on page 6-73
"HTTP_ACCEPT_ENCODING" on page 6-74
"HTTP_ACCEPT_LANGUAGE" on page 6-75
"HTTP_COOKIE" on page 6-76
"HTTP_HOST" on page 6-77
"HTTP_INTERNETUSER" on page 6-78
"HTTP_REFERER" on page 6-79
"HTTP_USER_AGENT" on page 6-80
"PATH_INFO" on page 6-190
"QUERY_STRING" on page 6-199
"REMOTE_ADDR" on page 6-204
"REMOTE_HOST" on page 6-205
"REQUEST_METHOD" on page 6-206
"SCRIPT_NAME" on page 6-220
"SERVER_NAME" on page 6-234
"SERVER_PORT" on page 6-235
"SERVER_PROTOCOL" on page 6-236
"SERVER_SOFTWARE" on page 6-237

5.4 Global Functions

Idoc Script has many built-in global functions. Functions perform actions, including string comparison and manipulation routines, date formatting, and ResultSet manipulation. Some functions also return results, such as the results of calculations or comparisons. For a list of global functions and details about their use, see "Global Functions" on page 3-11.

The following is a list of Idoc Script global functions:

"clearSchemaData" on page 6-20
"computeRenditionUrl" on page 6-23
"docUrlAllowDisclosure" on page 6-39
"encodeHtml" on page 6-49
"formatDateDisplay" on page 8-98
"formatTimeOnlyDisplay" on page 8-104
"getFieldConfigValue" on page 6-62
"getFieldViewDisplayValue" on page 6-63
"getParentValue" on page 6-64
"getViewValue" on page 6-65
"getViewValueResultSet" on page 6-66
"hasAppRights" on page 6-67
"indexerSetCollectionValue" on page 6-94
"isLayoutEnabled" on page 6-118
"jsFilename" on page 6-140
"lmGetLayout" on page 6-158
"lmGetSkin" on page 6-159
"loadCollectionInfo" on page 6-160
"loadSchemaData" on page 6-161
"regexMatches" on page 6-201
"regexReplaceAll" on page 6-202
"regexReplaceFirst" on page 6-203
"stdSecurityCheck" on page 6-251

5.5 Setable Variables

Setable variables can be set within script or used within a CGI string. For a list of setable variables and details about their use, see "Setable Variables" on page 3-19.

"getCookie" on page 6-60
"setCookie" on page 6-239

5.5.1 Workflows

Idoc Script includes predefined settable functions and variables that are used specifically for workflows.

For a detailed description of how workflows are implemented in Content Server, see the *Workflow Implementation Guide*. The following is a list of workflow functions:

"wfAddActionHistoryEvent" on page 6-276
"wfAddUser" on page 6-278
"wfComputeStepUserList" on page 6-279
"wfCurrentGet" on page 6-280
"wfCurrentSet" on page 6-281
"wfCurrentStep" on page 6-282
"wfDisplayCondition" on page 6-283
"wfExit" on page 6-284
"wfGet" on page 6-285
"wfIsFinishedDocConversion" on page 6-287
"wfIsNotifyingUsers" on page 6-288

"wfIsReleasable" on page 6-289
"wfLoadDesign" on page 6-295
"wfNotify" on page 6-298
"wfSet" on page 6-301
"wfSetIsNotifyingUsers" on page 6-302
"wfUpdateMetaData" on page 6-304

Workflow variables are used to set values related to workflows. The following is a list of workflow variables:

"dWfName" on page 6-42
"dWfStepName" on page 6-43
"entryCount" on page 6-50
"lastEntryTs" on page 6-141
"wfAction" on page 6-275
"wfAdditionalExitCondition" on page 6-277
"wfGetStepTypeLabel" on page 6-286
"wfJumpEntryNotifyOff" on page 6-290
"wfJumpMessage" on page 6-291
"wfJumpName" on page 6-292
"wfJumpReturnStep" on page 6-293
"wfJumpTargetStep" on page 6-294
"wfMailSubject" on page 6-296
"wfMessage" on page 6-297
"wfParentList" on page 6-299
"wfReleaseDocument" on page 6-300
"WfStart" on page 6-303

5.6 Value Variables

A *value* variable is evaluated once at the beginning of a service call and that value is used throughout the service call. The variable is then reevaluated on each new service call. Certain value variables can be set at the end of a service call or during a service call. For a list of value variables and details about their use, see "Value Variables" on page 3-21.

The following is a list of Idoc Script value variables used in core Content Server functionality:

"DocTypeSelected" on page 6-37
"DocUrl" on page 6-38
"fileUrl" on page 6-53
"FIRSTREV" on page 6-54
"HasOriginal" on page 6-69
"IsEditRev" on page 6-111
"IsFailedIndex" on page 6-112
"IsNotLatestRev" on page 6-121
"isZoneSearchField" on page 6-134
"NoMatches" on page 6-180
"OneMatch" on page 6-189
"StatusCode" on page 6-249
"StatusMessage" on page 6-250
"UseHtmlOrTextHighlightInfo" on page 6-261

5.7 Content Items

The following Idoc Script variables and functions are related to content items.

"AllowAlternateMetaFile" on page 6-9
"AllowPrimaryMetaFile" on page 6-11
"AllowSamePrimaryAlternateExtensions" on page 6-13
"AutoNumberPrefix" on page 6-17
"AutoNumberWidth" on page 6-18
"computeRenditionUrl" on page 6-23
"CONTENT_LENGTH" on page 6-24
"createAlternateMetaFile" on page 6-27
"createPrimaryMetaFile" on page 6-28
"DocTypeSelected" on page 6-37
"DocUrl" on page 6-38
"EnableExpirationNotifier" on page 6-46
"ExpireRevisionOnly" on page 6-52
"fileUrl" on page 6-53
"FIRSTREV" on page 6-54
"FormInheritMetaDataFieldList" on page 6-58
"ForceDistinctRevLabels" on page 6-55
"HasOriginal" on page 6-69
"HasUrl" on page 6-71
"IsAutoNumber" on page 6-105
"IsCriteriaSubscription" on page 6-110
"IsEditRev" on page 6-111
"IsFailedIndex" on page 6-112
"IsFilePresent" on page 6-113
"IsFullTextIndexed" on page 6-115
"IsLocalSearchCollectionID" on page 6-119
"IsNotLatestRev" on page 6-121
"IsNotSyncRev" on page 6-122
"MajorRevSeq" on page 6-163
"MinorRevSeq" on page 6-178
"NotificationIntervalInDays" on page 6-182
"NotificationMaximum" on page 6-183
"NotificationQuery" on page 6-184
"NotifyExtras" on page 6-185
"NotifyTime" on page 6-186
"SingleGroup" on page 6-242
"SourceID" on page 6-247

5.8 Content Server

The following Idoc Script variables are related to the Content Server.

"AdditionalSubscriptionCollatedFieldList" on page 6-4
"AppletChunkSize" on page 6-14
"AppletChunkThreshold" on page 6-15
"CookieAuthTimeoutInDays" on page 6-25
"CookieProxyPassword" on page 6-26
"DefaultHtmlRefreshTimeoutInSeconds" on page 6-32
"DefaultPasswordEncoding" on page 6-33
"DisableDBLongIntegerField" on page 6-34
"DisableHttpUploadChunking" on page 6-36
"DoDocNameOrder" on page 6-40

"getFieldConfigValue" on page 6-62
"HasExternalUsers" on page 6-68
"IDC_Name" on page 6-82
"IdcCommandFile" on page 6-84
"IdcCommandLog" on page 6-85
"IdcCommandServerHost" on page 6-86
"InstanceDescription" on page 6-99
"InstanceMenuLabel" on page 6-100
"IntradocServerHostName" on page 6-102
"IntradocServerPort" on page 6-103
"isComponentEnabled" on page 6-108
"IsContributor" on page 6-109
"IsFormsPresent" on page 6-114
"isLayoutEnabled" on page 6-118
"IsPromptingForLogin" on page 6-124
"IsWebServerPagesOnly" on page 6-132
"isValidateFile" on page 6-131
"LmDefaultLayout" on page 6-156
"LmDefaultSkin" on page 6-157
"lmGetLayout" on page 6-158
"lmGetSkin" on page 6-159
"MailServer" on page 6-162
"MaxIntegerAllowed" on page 6-167
"MaxQueryRows" on page 6-168
"MaxRequestThreadCount" on page 6-169
"MaxSizeResourceFileCacheMillions" on page 6-173
"MaxStandardDatabaseResults" on page 6-174
"MinIntegerAllowed" on page 6-176
"NoAutomation" on page 6-179
"PrimaryWorkQueueTimeout" on page 6-193
"ProviderClass" on page 6-195
"ProviderConfig" on page 6-196
"ProviderConnection" on page 6-197
"ProviderType" on page 6-198
"ServiceStartupTimeout" on page 6-238
"ShowCreateExternalMenu" on page 6-240
"SkipMozillaMailToWorkaround" on page 6-243
"SmtpPort" on page 6-244
"StatusCode" on page 6-249
"StatusMessage" on page 6-250
"SubscriptionCollatedFieldList" on page 6-252
"SubscriptionMethod" on page 6-253
"UseRevisionCreateDate" on page 6-264
"XMLEncodingMode" on page 6-307

5.9 Database

The following Idoc Script variables and functions are related to databases.

"ColumnMapFile" on page 6-21
"DatabasePreserveCase" on page 6-29
"DatabaseWildcards" on page 6-30
"ForceJdbcSync" on page 6-56
"IsJdbcLockTrace" on page 6-116
"IsJdbcQueryTrace" on page 6-117

"JdbcConnectionString" on page 6-135
"JdbcDriver" on page 6-136
"JdbcPassword" on page 6-137
"JdbcPasswordEncoding" on page 6-138
"JdbcUser" on page 6-139
"MemoFieldSize" on page 6-175
"MinMemoFieldSize" on page 6-177
"NumConnections" on page 6-188
"UseDatabaseShortIndexName" on page 6-260

5.10 Indexing

The following Idoc Script variables are related to indexing.

"AdditionalIndexBuildParams" on page 6-2
"AllowConcurrentUpdate" on page 6-10
"IndexableRenditions" on page 6-89
"IndexerAutoWorkInterval" on page 6-90
"IndexerLargeFileSize" on page 6-91
"IndexerRebuildStagingManualCopy" on page 6-92
"IndexerRebuildStagingPath" on page 6-93
"indexerSetCollectionValue" on page 6-94
"IndexerTouchMonitorInterval" on page 6-95
"IndexerTraceCycleListChanges" on page 6-96
"IndexerTraceLockFileChanges" on page 6-97
"IndexerTraceStartStop" on page 6-98
"IsAllowIndexRenditions" on page 6-104
"IsFailedIndex" on page 6-112
"IsFullTextIndexed" on page 6-115
"LkDisableOnRebuild" on page 6-147
"LkDisableReferenceInfo" on page 6-148
"LkDisallowConversionFormats" on page 6-149
"LkIsSecureSearch" on page 6-150
"LkReExtractOnRebuild" on page 6-151
"LkRefreshBatchSize" on page 6-152
"LkRefreshErrorsAllowed" on page 6-154
"LkRefreshErrorPercent" on page 6-153
"LkRefreshErrorThreshold" on page 6-155
"MaxCollectionSize" on page 6-164
"MaxDocIndexErrors" on page 6-165
"TimeoutPerOneMegInSec" on page 6-255

5.11 Schema

The following Idoc Script functions are related to schemas.

5.11.1 Schema publish settings

Schema are 'republished' whenever a change occurs that may affect the relationship between the parts of the schema. These settings relate to publishing factors.

"clearSchemaData" on page 6-20
"EnableSchemaPublish" on page 6-48
"getFieldViewDisplayValue" on page 6-63
"getParentValue" on page 6-64
"getViewValue" on page 6-65

"getViewValueResultSet" on page 6-66
"jsFilename" on page 6-140
"loadSchemaData" on page 6-161
"SchemaAllowRecursiveFileMove" on page 6-208
"SchemaBuildDigestFile" on page 6-209
"SchemaPublishCheckInterval" on page 6-210
"SchemaPublishInterval" on page 6-211
"SchemaPublishMaximumInterval" on page 6-212
"SchemaPublishMinimumInterval" on page 6-213
"SchemaPublishRenameRetryInitialInterval" on page 6-214
"SchemaPublishRenameRetryIterations" on page 6-215
"SchemaPublishRetryInterval" on page 6-216
"SchemaPublishRetryIntervalMax" on page 6-217
"SchemaVerifyPublishing" on page 6-218
"SchemaWin32RenameDelayInterval" on page 6-219

5.12 Searching

The following Idoc Script variables and functions are related to searching.

"CachedResultRowCount" on page 6-19
"EnableDocumentHighlight" on page 6-45
"HasSiteStudio" on page 10-44
"indexerSetCollectionValue" on page 6-94
"IsAutoSearch" on page 6-106
"IsFullTextIndexed" on page 6-115
"IsLocalSearchCollectionID" on page 6-119
"isZoneSearchField" on page 6-134
"loadCollectionInfo" on page 6-160
"MaxHitCount" on page 6-166
"MaxResults" on page 6-170
"MaxSavedSearchResults" on page 6-171
"MaxSearchConnections" on page 6-172
"NoMatches" on page 6-180
"NoSearchConnectionCloseOnDocChange" on page 6-181
"OneMatch" on page 6-189
"QUERY_STRING" on page 6-199
"regexMatches" on page 6-201
"regexReplaceAll" on page 6-202
"regexReplaceFirst" on page 6-203
"SearchCacheCleanerInterval" on page 6-221
"SearchCacheHardLimit" on page 6-222
"SearchCacheMaxItemAgeInMinutes" on page 6-223
"SearchCacheSoftLimit" on page 6-224
"SearchCacheTrace" on page 6-225
"SearchConnectionWaitTimeout" on page 6-226
"SearchDebugLevel" on page 6-227
"SearchDir" on page 6-228
"SearchIndexerEngineName" on page 6-229
"SearchQueryFormat" on page 6-230
"SearchQueryTrace" on page 6-231
"UniversalSearchAllowFullText" on page 6-257
"UseHtmlOrTextHighlightInfo" on page 6-261
"UseSearchCache" on page 6-270
"UseXmlUrl" on page 6-273

5.13 Security

This section lists the Idoc Script variables and functions that are related to security.

- "Internal Security" on page 5-12
- "External Security" on page 5-12

5.13.1 Internal Security

The following Idoc Script variables and functions are related to internal security.

"AdminAtLeastOneGroup" on page 6-5
"AllowAllRedirectUrls" on page 6-8
"AllowCheckin" on page 6-6
"AllowCheckout" on page 6-7
"AuthorDelete" on page 6-16
"Default Accounts" on page 6-31
"docUrlAllowDisclosure" on page 6-39
"EmptyAccountCheckinAllowed" on page 6-44
"encodeHtml" on page 6-49
"ExclusiveCheckout" on page 6-51
"GetCopyAccess" on page 6-61
"hasAppRights" on page 6-67
"HasPredefinedAccounts" on page 6-70
"HtmlDataInputFilterLevel" on page 6-72
"IsCheckinPreAuthed" on page 6-107
"IsLoggedIn" on page 6-120
"IsSubAdmin" on page 6-127
"IsSysManager" on page 6-128
"isUserOverrideSet" on page 6-130
"SelfRegisteredAccounts" on page 6-232
"SelfRegisteredRoles" on page 6-233
"ShowOnlyKnownAccounts" on page 6-241
"stdSecurityCheck" on page 6-251
"UseAccounts" on page 6-258
"UserFilteredSubjects" on page 6-265
"userHasAccessToAccount" on page 6-266
"userHasGroupPrivilege" on page 6-267
"userHasRole" on page 6-268
"UserIsAdmin" on page 6-269

5.13.2 External Security

The following Idoc Script variables are related to external security (web server and NTLM security).

"DomainControllerName" on page 6-41
"HasExternalUsers" on page 6-68
"IntradocRealm" on page 6-101
"IsPrimaryUserBase" on page 6-123
"IsPromptingForLogin" on page 6-124
"IsUploadSockets" on page 6-129
"LdapAdminDN" on page 6-142
"LdapAdminPassword" on page 6-143
"LdapPort" on page 6-144
"LdapServer" on page 6-145
"LdapSuffix" on page 6-146

"NtlmSecurityEnabled" on page 6-187
"Priority" on page 6-194
"ProviderClass" on page 6-195
"ProviderConfig" on page 6-196
"ProviderConnection" on page 6-197
"ProviderType" on page 6-198
"RedirectHostsFilter" on page 6-200
"SocketHostAddressSecurityFilter" on page 6-245
"SocketHostNameSecurityFilter" on page 6-246
"SourcePath" on page 6-248
"UseAdsi" on page 6-259
"UseNetscape" on page 6-262
"UseNtlm" on page 6-263
"UseSecureLdap" on page 6-271
"UseSSL" on page 6-272
"WebServerAuthOnly" on page 6-274

5.14 Web Server

The following Idoc Script variables are related to Web servers.

"DomainControllerName" on page 6-41
"GATEWAY_INTERFACE" on page 6-59
"GATEWAY_INTERFACE" on page 6-59
"getCookie" on page 6-60
"HTTP_ACCEPT" on page 6-73
"HTTP_ACCEPT_ENCODING" on page 6-74
"HTTP_ACCEPT_LANGUAGE" on page 6-75
"HTTP_COOKIE" on page 6-76
"HTTP_HOST" on page 6-77
"HTTP_INTERNETUSER" on page 6-78
"HTTP_REFERER" on page 6-79
"HTTP_USER_AGENT" on page 6-80
"HttpIgnoreWebServerInternalPortNumber" on page 6-81
"IdcAuthExtraRequestParams" on page 6-83
"IdcHttpHeaderVariables" on page 6-88
"IntradocRealm" on page 6-101
"PATH_INFO" on page 6-190
"PLUGIN_DEBUG" on page 6-192
"QUERY_STRING" on page 6-199
"RedirectHostsFilter" on page 6-200
"REMOTE_ADDR" on page 6-204
"REMOTE_HOST" on page 6-205
"REQUEST_METHOD" on page 6-206
"SCRIPT_NAME" on page 6-220
"SERVER_NAME" on page 6-234
"SERVER_PORT" on page 6-235
"SERVER_PROTOCOL" on page 6-236
"SERVER_SOFTWARE" on page 6-237
"setCookie" on page 6-239
"SocketHostAddressSecurityFilter" on page 6-245
"SocketHostNameSecurityFilter" on page 6-246
"UseSSL" on page 6-272
"WebServerAuthOnly" on page 6-274

5.15 Workflow

The following Idoc Script variables and functions are related to workflows.

5.15.1 Configuration Variables

"isRepromptLogin" on page 6-125
"IsSavedWfCompanionFile" on page 6-126
"PrimaryWorkQueueTimeout" on page 6-193
"WorkflowIntervalHours" on page 6-306

5.15.2 Workflow Functions

"wfAddActionHistoryEvent" on page 6-276
"wfAddUser" on page 6-278
"wfComputeStepUserList" on page 6-279
"wfCurrentGet" on page 6-280
"wfCurrentSet" on page 6-281
"wfCurrentStep" on page 6-282
"wfDisplayCondition" on page 6-283
"wfExit" on page 6-284
"wfGet" on page 6-285
"wfGetStepTypeLabel" on page 6-286
"wfIsFinishedDocConversion" on page 6-287
"wfIsNotifyingUsers" on page 6-288
"wfIsReleasable" on page 6-289
"wfLoadDesign" on page 6-295
"wfNotify" on page 6-298
"wfReleaseDocument" on page 6-300
"wfSet" on page 6-301
"wfSetIsNotifyingUsers" on page 6-302
"wfUpdateMetaData" on page 6-304

5.15.3 Other Variables

"AllowReview" on page 6-12
"dWfName" on page 6-42
"dWfStepName" on page 6-43
"entryCount" on page 6-50
"IsEditRev" on page 6-111
"IsWorkflow" on page 6-133
"lastEntryTs" on page 6-141
"SingleGroup" on page 6-242
"wfAction" on page 6-275
"wfAdditionalExitCondition" on page 6-277
"wfJumpEntryNotifyOff" on page 6-290
"wfJumpMessage" on page 6-291
"wfJumpName" on page 6-292
"wfJumpReturnStep" on page 6-293
"wfJumpTargetStep" on page 6-294
"wfMailSubject" on page 6-296
"wfMessage" on page 6-297
"wfParentList" on page 6-299
"WfStart" on page 6-303

6

Core Content Server Variables and Functions

This chapter lists Idoc Script variables and functions in alphabetical order, and provides details about each item.

AdditionalIndexBuildParams

Adds build parameters to every indexer execution.

- This can be used to force optimization to occur after every indexing bulkload instead of every few bulkloads. For example, when a collection is not fully merged, the collection can be spread over many files, making searching inefficient and consuming many file handles.
- Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Parameters

- Each build parameter is prefixed with a hyphen. For example, `-repair`.
- If a build parameter has its own sub-parameters, the first sub-parameter does not have a hyphen. For example, `-optimize maxmerge`.
- If a build parameter has multiple sub-parameters, each sub-parameter after the first one is prefixed with a hyphen. For example, `-optimize maxmerge -squeeze`.

Parameter	Description
`-repair`	Repairs the collection.
`-servlev`	Overrides the service level to allow collection functions to be enabled, using one or more of the following parameters: **search**: Enables search and retrieval. **optimize**: Enables collection optimization. **assist**: Enables building of the word list. **housekeep**: Enables housekeeping of unneeded files. **insert**: Enables adding and updating documents. **delete**: Enables document deletion. **backup**: Enables backup. **purge**: Enables background purging. **repair**: Enables collection repair. **index**: Same as *insert -delete*. **dataprep**: Same as *search -optimize -assist -housekeep -index*.

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
AdditionalIndexBuildParams=-repair
```

```
AdditionalIndexBuildParams=-optimize maxclean
```

```
AdditionalIndexBuildParams=-optimize maxmerge -squeeze
```

AdditionalSubscriptionCollatedFieldList

Defines a list of additional fields displayed for subscriptions.

The default fields are defined in `SubscriptionCollatedFieldList`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

```
AdditionalSubscriptionCollatedFieldList=dOriginalName
```

See Also

- "SubscriptionCollatedFieldList" on page 6-252

AdminAtLeastOneGroup

Checks if the current user has the *admin* role for at least one security group.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user is an administrator for at least one security group.
- Returns FALSE if the user is not assigned the *admin* role.

Example

Can be used to do an optional presentation for an administrator:

```
<$if (AdminAtLeastOneGroup)$>
    <a href="<$redirect$>">
<$endif$>
```

See Also

- "UserAppRights" on page 8-341
- "UserIsAdmin" on page 6-269

AllowCheckin

Checks if the current user has checkin permission for the content item's security group.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user has checkin permission.
- Returns FALSE if the user does not have checkin permission.

Example

Can be used to do an optional presentation for a user with checkin permission:

```
<$if (AllowCheckin)$>
    <a href="<$redirect$>">
<$endif$>
```

AllowCheckout

Checks whether current user has checkout permission for the content item's security group.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user has checkout permission.
- Returns FALSE if the user does not have checkout permission.

Example

Can be used to do an optional presentation for a user with checkout permission:

```
<$if (AllowCheckout)$>
    <a href="<$redirect$>">
<$endif$>
```

AllowAllRedirectUrls

Resets functionality defined by `RedirectHostsFilter` and causes the Content Server to act as it did before that variable was set. Setting this variable to TRUE allows the user to be redirected to any URL instead of those specified by `RedirectHostsFilter`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
AllowAllRedirectUrls=true
```

See Also

- "RedirectHostsFilter" on page 6-200

AllowAlternateMetaFile

Allows users to submit metadata-only content as the alternate file.

- Enables the metadata-only file creation capability. In this scenario, the contributor does not specify an alternate file. Instead, the system will create the alternate file using a template and the content item's metadata.
- If set to TRUE in the *config.cfg* file, an extra check box is added to the checkin page. When this check box is selected, the createAlternateMetaFile setting is enabled, which directs the system to actually create the alternate file.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
AllowAlternateMetaFile=true
```

See Also

- "createAlternateMetaFile" on page 6-27
- "AllowPrimaryMetaFile" on page 6-11

AllowConcurrentUpdate

Used to allow new documents to be added to the existing collection during a rebuild of the index.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
AllowConcurrentUpdate=TRUE
```

AllowPrimaryMetaFile

Allows users to submit metadata-only content as the primary file.

- Enables the metadata-only file creation capability. In this scenario, the contributor does not specify a primary file. Instead, the system will create the primary file using a template and the content item's metadata.
- If set to TRUE in the *config.cfg* file, an extra check box is added to the checkin page. When this check box is selected, the createPrimaryMetaFile setting is enabled, which directs the system to actually create the primary file.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
AllowPrimaryMetaFile=true
```

See Also

- "AllowAlternateMetaFile" on page 6-9
- "createPrimaryMetaFile" on page 6-28

AllowReview

Checks if the current user is allowed to approve or reject the current workflow item.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Workflow" on page 4-21

Output

- Returns TRUE if the user is a reviewer for the current workflow step.
- Returns FALSE if the user is not a reviewer for the current workflow step.

Example

Displays Approve and Reject buttons if the user is a reviewer:

```
<$if AllowReview$>
    <$include workflow_doc_action_buttons$>
<$endif$>
```

AllowSamePrimaryAlternateExtensions

Allows the primary file and alternate file to have the same file extension.

- When set to TRUE, the primary file and alternate file can have the same file extension.
- When set to FALSE, the primary file and alternate file cannot have the same file extension.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
AllowSamePrimaryAlternateExtensions=true
```

AppletChunkSize

Sets the size of the transfer chunks for the chunking function, in bytes.

- Files that exceed the AppletChunkThreshold size will be transferred in chunks of this size.
- Default is 1 MB.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
AppletChunkSize=10000
```

See Also

- "AppletChunkThreshold" on page 6-15
- "ChunkedRequestTrace" on page 8-22
- "DisableHttpUploadChunking" on page 6-36

AppletChunkThreshold

Sets the minimum file size for the chunking function, in bytes.

- The chunking function will not be used for files smaller than this size.
- Default is 1 MB.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
AppletChunkThreshold=500000
```

See Also

- "AppletChunkThreshold" on page 6-15
- "ChunkedRequestTrace" on page 8-22
- "DisableHttpUploadChunking" on page 6-36

AuthorDelete

Enables authors to delete their own revisions without Delete permission to the security group.

- When set to TRUE, authors can delete their own revisions without Delete permission to the security group.
- Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18

Location

- System Properties, Content Security tab-Allow author to delete revision
- Admin Server, Content Security-Allow author to delete revision
- *IntradocDir*/config/config.cfg

Example

As a configuration entry:

```
AuthorDelete=true
```

As script:

```
<$if AuthorDelete$>
    <$AuthorDelete$>
<$else$>
    false
<$endif$>
```

AutoNumberPrefix

Defines the prefix that will be added to all automatically numbered Content IDs.

- Returns the automatic numbering prefix (returns value in configuration settings).
- Returns a string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

- System Properties, Options tab, Auto Name Prefix
- Admin Server, General Configuration, Auto Number Prefix
- *IntradocDir*/config/config.cfg

Example

As a configuration setting, defines the automatic numbering prefix:

```
AutoNumberPrefix=HR
```

As script, returns the value of the configuration setting:

```
<$AutoNumberPrefix$>
```

See Also

- "IsAutoNumber" on page 6-105

AutoNumberWidth

Allows you to use more than six digits for a Content ID number, providing support for up to one hundred million content items (eight digits).

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

- *IntradocDir*/config/config.cfg

Example

As a configuration setting:

```
AutoNumberWidth=8
```

CachedResultRowCount

Sets the size of the search cache.

- The search cache can improve search performance by caching recent search engine queries. Increasing the size reduces search time, but increases memory usage.
- Default is 20,000 rows.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
CachedResultRowCount=30000
```

clearSchemaData

Clears the data from a schema ResultSet.

Type and Usage

- "Global Functions" on page 3-11
- "Schema" on page 4-16

Parameters

This function can take zero, one, or two parameters.

- If passed zero arguments, it clears the data binder. Returns no value.
- If passed one argument, the argument is the name of the resultset to clear the values from the current row. Returns no value.
- If passed two arguments, the first argument is the resultset name and the second argument is the key identifying the data object to clear. Returns 0 if the data does not exist or 1 if it does exist. The use is True() or False() to conditionally execute scripts based on the return value.

Example

```
<$clearSchemaData()$>
```

See Also

- "loadSchemaData" on page 6-161

ColumnMapFile

Specifies the column mapping HTM resource file.

- The system uses the column mapping file to map database column names to all uppercase, which is required for Oracle.
- If no path is referenced, the default path is *instance_dir*/shared/config/resources/.
- The standard column mapping file provided with Content Server is *upper_clmns_map.htm*.
- There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ColumnMapFile=upper_clmns_map.htm
```

computeDocUrl

This function computes the URL to a content item based on the data on the page.

For example, this function can be called to generate a URL to an item when looping over a ResultSet of items.

The following information must be present on the page:

- dDocAccount (optional)
- dDocName
- dDocType
- dProcessingState
- dRevLabel
- dSecurityGroup
- dWebExtension

Type and Usage

- "Global Functions" on page 3-11
- "Content Items" on page 4-3

Parameters

The only parameter indicates if the URL is relative, set to TRUE (1) or FALSE (0).

computeRenditionUrl

Returns the URL of a given rendition.

Type and Usage

- "Global Functions" on page 3-11
- "Content Items" on page 4-3

Parameters

Takes three parameters:

- The first parameter is the URL of the content item.
- The second parameter is the *dRevLabel* value.
- The third parameter is the *dRendition1* value. Possible values of *dRendition1* include:
  - **T** = Thumbnail rendition
  - **X** = XML rendition

Output

Returns the complete URL of the rendition as a string.

Example

Returns the URL of the rendition as a string.

```
<$computeRenditionUrl(url, dRevLabel, dRendition1)$>
```

CONTENT_LENGTH

Retrieves the length in bytes of the requested content item as supplied by the client.

This variable is specific to the current gateway program request.

Important: This setting is obsolete for version 7.0 and later. The Web server filter no longer sends this information.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21
- "Content Items" on page 4-3

Output

Returns the content length in bytes.

Example

As information output on a page or to a log:

```
CONTENT_LENGTH=0
```

As part of an Idoc Script statement or evaluation:

```
<$if CONTENT_LENGTH$>
<!--statement-->
```

CookieAuthTimeoutInDays

Used with the ExtranetLook component. This variable is used with the CookieLoginPlugin, which monitors Web requests and determines if a request is authenticated. It can support authentication based on browser cookies.

- When `CookieAuthTimeoutInDays` is set to a positive numeric value, the cookie will not time out for that number of days. The user needs a logout option that clears the cookie.
- If set to zero (0) or less, the cookie lasts as long as the browser session.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

```
CookieAuthTimeoutInDays=2
```

See Also

- "CookieProxyPassword" on page 6-26
- "IsWebServerPagesOnly" on page 6-132

CookieProxyPassword

Used with the ExtranetLook component. This variable is used to encrypt the passwords when users log in to the Web site using cookie authentication. It is used with the CookieLoginPlugin, which monitors Web requests and determines if a request is authenticated. It can support authentication based on browser cookies. This can be set to any desired value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

```
CookieProxyPassword=x2x
```

See Also

- "CookieAuthTimeoutInDays" on page 6-25
- "IsWebServerPagesOnly" on page 6-132

createAlternateMetaFile

Allows users to submit metadata-only content for the alternate file.

- Directs the system to create a metadata-only alternate file. In this scenario, the contributor does not specify an alternate file. Instead, the system will create the alternate file using a template and the content item's metadata.
- When AllowAlternateMetaFile is set to TRUE, an extra check box is added to the checkin page. When this check box is selected, the *createAlternateMetaFile* variable is enabled.
- For use in the Batch Loader, *createAlternateMetaFile* must be set to TRUE in the batch load file.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
createAlternateMetaFile=true
```

See Also

- "AllowAlternateMetaFile" on page 6-9
- "createPrimaryMetaFile" on page 6-28

createPrimaryMetaFile

Allows users to submit metadata-only content for the primary file.

- Directs the system to create a metadata-only primary file. In this scenario, the contributor does not specify a primary file. Instead, the system will create the primary file using a template and the content item's metadata.
- When AllowPrimaryMetaFile is set to TRUE, an extra check box is added to the checkin page. When this check box is selected, the *createPrimaryMetaFile* variable is enabled.
- For use in the Batch Loader, *createPrimaryMetaFile* must be set to TRUE in the batch load file.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
createPrimaryMetaFile=true
```

See Also

- "AllowPrimaryMetaFile" on page 6-11
- "createAlternateMetaFile" on page 6-27

DatabasePreserveCase

Defines whether the character case from the database is preserved.

- When set to TRUE, character case is preserved.
- This setting must be TRUE for Oracle and any other database that is case sensitive.
- Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

- System Properties, Database tab, Enable database preserve case
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
DatabasePreserveCase=false
```

DatabaseWildcards

Defines the wildcard characters that are used to represent multiple characters in database queries.

- For JDBC databases, the default is `%`.
- For Microsoft Access, the default is `*`.

Type and Usage

- "Configuration Variables" on page 3-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DatabaseWildcards=%
```

Default Accounts

Defines the default accounts for anonymous users.

- This must be a comma-delimited list of accounts.
- Permissions for each account must be specified in parentheses after the account name.
- The `#none` entry grants privileges to content items that have no account assigned. The `#all` entry grants privileges to all accounts.
- Default is #*none(RWDA)*.
- Returns the list of accounts as a string.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Location

IntradocDir/config/config.cfg

Example

As a configuration setting, defines default account information:

```
DefaultAccounts=BOS(R),SEA(RW),MSP/Gen(RWD)
```

As script, returns the account information as a string:

```
<$DefaultAccounts$>
```

See Also

- "ExternalUserAccounts" on page 8-74
- "SelfRegisteredAccounts" on page 6-232

DefaultHtmlRefreshTimeoutInSeconds

Defines the time, in seconds, that a Work In Progress page, My Checked-Out Content page and My Workflow Assignments page refreshes.

In addition to this variable, `htmlRefreshTimeout` can be set in the URL or the service's databinder. This can be used to set a different refresh time for different pages. If that is needed, do not set `DefaultHtmlRefreshTimeoutInSeconds` in the config.cfg file, but instead set `htmlRefreshTimeout` in the URL or databinder.

`htmlRefreshUrl` can be set to the URL of the page to load after the page is refreshed. The default is the current page. This variable can be set in the URL of the page or the databinder.

The default value for `DefaultHtmlRefreshTimeoutInSeconds` is 60.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DefaultHtmlRefreshTimeoutInSeconds=120
```

DefaultPasswordEncoding

Defines the type of encoding to use when storing user passwords.

- Default is to use the Secure Hash Algorithm update 1 (SHA1).
- If you want to have open text passwords, the suggested value is `OpenText`.
- If any other nonempty value is assigned, no encoding is performed.
- Default value is `SHA1-CB`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DefaultPasswordEncoding=OpenText
```

DisableDBLongIntegerField

In older versions of MSDE software, 4-byte integers were allowed. In newer versions, 8-byte integers and larger are used. This variable allows the Content Server to continue to use 4-byte integers.

If you upgrade Content Server 6.2 with MSDE to a newer release, the upgraded instance does not start because of the addition of the dRevRank column. Version 6.2 uses an older MSDE version than later versions. The system will start with the addition of this variable set to TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DisableDBLongIntegerField=true
```

DisableGzipCompression

Disables gzip compression of HTML pages and log files. Pages are compressed only if the client browser supports compressed HTML. Compression makes pages an average of 80% smaller and is supported by any HTTP 1.1 compatible browser.

Note that output from FILTER_DEBUG is in gzipped format unless this entry is used.

- When set to TRUE, pages will not be compressed before being sent to a client Web browser.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Conversion" on page 4-5

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DisableGzipCompression=true
```

See Also

- "HTTP_ACCEPT_ENCODING" on page 6-74

DisableHttpUploadChunking

Disables the upload chunking function, which is used with the Upload Applet or an HTTP provider.

- When set to TRUE, the chunking function is disabled.
- When set to FALSE, the chunking function is enabled.
- Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
DisableHttpUploadChunking=false
```

See Also

- "AppletChunkSize" on page 6-14
- "AppletChunkThreshold" on page 6-15
- "ChunkedRequestTrace" on page 8-22

DocTypeSelected

Evaluates whether the Type of the current content item matches the Type in the active ResultSet.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Items" on page 4-3

Output

- Returns TRUE if the content item Types match.
- Returns FALSE if the content item Types do not match.

Example

Returns value based on whether the content item type matches the type for the ResultSet.

```
<$DocTypeSelected$>
```

DocUrl

Retrieves the URL of the file in the *weblayout* directory. This variable is evaluated once per content item, not once per service call.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Items" on page 4-3

Output

Returns the URL of the file as a string.

Example

Used to build URL links to content items.

```
<$if HasUrl$>
    <a href="<$DocUrl$>"><$dDocName$></a>
<$else$>
    <$dDocName$>
<$endif$>
```

docUrlAllowDisclosure

Evaluates whether a URL can be disclosed to the current user.

Type and Usage

- "Global Functions" on page 3-11
- "Internal Security" on page 4-18

Parameters

The only parameter is an absolute path, such as the following:

```
http://mycomputer/domain/groups/.../documents/mydoc.pdf
```

A full relative path can be used, such as the following:

```
/oracle/domain/.../documents/mydoc.pdf).
```

Output

Returns a Boolean value.

- Returns TRUE if the URL can be disclosed.
- Returns FALSE if the URL is restricted.

Example

Determines if the user can view the URL of the *mydoc.pdf* document.

```
<$docUrlAllowDisclosure("/domain/groups/documents/mydoc.pdf")$>
```

DoDocNameOrder

Sorts content items in the Repository Manager by *dDocName*. When Repository Manager is started, a default query is run against the database. By default the query results are sorted by the dDocName of the items. Ordering by dDocName makes the order predictable but can result in longer query results.

- When set to TRUE, content items are sorted by *dDocName*.
- When set to FALSE, content items are not sorted.
- Default is TRUE.

You can enable the JDBC Query Trace to log information to the console log and verify that the 'order by' clause is no longer part of the query. To enable the JDBC Query Trace, select `systemdatabase` as an option in the Edit Active Console Output Tracing section of the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DoDocNameOrder=false
```

DomainControllerName

Defines the domain controller to use for NTLM calls.

- If a value is not set, an API call is made to find a domain controller to talk to.
- Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21
- "External Security" on page 4-19

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DomainControllerName=PDC servername
```

dWfName

Retrieves the name of the workflow.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the name of the workflow as a string.

Example

The following code in the *IdcHomeDir*/resources/core/templates/workflow_info.htm template page is used to display the workflow name on the Workflow Step Information page:

```
<td align=right><span class=infoLabel><$lc("wwLabelWorkflow")$></span></td>
<td><span class=tableEntry><$dWfName$></span></td>
```

dWfStepName

Retrieves the name of the current step in the workflow.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the name of the current step as a string.

Example

The following code in the *IdcHomeDir*/resources/core/templates/workflow_info.htm template page is used to display the current step name on the Workflow Step Information page:

```
<td align=right><span class=infoLabel><$lc("wwCurrentStep")$>
</span></td>
<td><span class=tableEntry>
    <$loop WorkflowStep$>
    <$dWfStepName$>
    <$exec RequiredUsers = dWfStepWeight$>
    <$if isTrue(dWfStepIsAll)$><$exec RequiredUsers = 0$><$endif$>
    <$endloop$></span>
</td>
```

EmptyAccountCheckinAllowed

Checks if an account must be specified on the checkin page.

Used on the Standard Page Resources page to display an error message if an account is not specified.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18

Output

- Returns TRUE if an Account value is required.
- Returns FALSE if an Account value is not required.

Example

Evaluates whether an account number is required and displays an error message.

```
<$if not EmptyAccountCheckinAllowed$>
    <$isRequired = 1, requiredMsg = "Please specify an account."$>
<$endif$>
```

EnableDocumentHighlight

Enables highlighting of full-text search terms in PDF, text, and HTML files.

- When set to TRUE, search term highlighting is enabled.
- When set to FALSE, search term highlighting is disabled.
- Default is FALSE after installation of Content Server.
- Default is TRUE after installation of PDF Converter.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

- System Properties, Options tab, Enable search keyword highlighting
- Admin Server, General Configuration, Enable search keyword highlighting
- *IntradocDir*/config/config.cfg

Example

As a configuration setting:

```
EnableDocumentHighlight=false
```

As script, returns the value of the configuration setting:

```
<$EnableDocumentHighlight$>
```

See Also

- "UseHtmlOrTextHighlightInfo" on page 6-261

EnableExpirationNotifier

Enables notification of expiration (NOE) of content items. This provides email notification to authors and administrators for documents that will be expired in a short time.

- After enabling the NOE and restarting, an entry is included in the ScheduledEvents *DomainHome*/ucm/cs/config/activestate.hda file.
- The include that controls when the NOE runs is *set_doevent_for_notification_of_expiration*. The NotifyTime parameter described below is used in this include.
- Note that in the following parameters, NotificationQuery can be set to 30 (not 7, as is the default) and NotificationIntervalInDays can be set to 7; this provides weekly emails about what will expire within the month.
- The table that defines scheduled events is in the *IdcHomeDir*/resources/core/tables/std_resources.htm file. The table is named *IdcScheduledSystemEvents* and the row is named *NotificationOfExpiration*.
- To obtain debugging information for the NOE, set the following variables in the config.cfg file and restart the Content Server. Logging goes to the server output:

```
TraceSectionsList=system,scheduledevents
TraceIsVerbose=true
```

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Parameters

Takes the following optional parameters:

Parameter	Description
NotificationQuery	Specifies the query to be run to find the expired documents for the notification. The default is the following: `dOutDate > <$dateCurrent()$> <AND> dOutDate < <$dateCurrent(7)$>`
NotificationMaximum	Specifies the maximum number of documents returned from the query. Default: 1000.
NotifyExtras	Denotes the administration users who receive the notification emails. If this entry is not used, the default is `sysadmin`. If left blank, no extra notification is sent.
NotifyTime	The time of day to send the notification. Default: 00:01. Specified in 24-hour clock notation.

Parameter	Description
NotificationFieldNames	Specifies the field names to include in the notification. Default: dDocAuthor, dOutDate, dDocTitle, dDocName, dDocType, dID, dInDate, dRevLabel, dSecurityGroup
NOEUsePlainTextEmail	Allows the use of plain text email for the notification. Default: false. The email template is named QUERY_NOTIFICATION. If `NOEUsePlainTextEmail=true`, `emailFormat=text`.
NotificationIntervalInDays	Specifies the interval in days. Default: 1.

Example

Used as a configuration entry:

```
EnableExpirationNotifier=true
```

EnableSchemaPublish

Enables or disables schema publishing.

- When set to TRUE, publishing is enabled. When set to FALSE, publishing is disabled.
- Default is TRUE.

> **Caution:** If set to FALSE, publishing will be disabled even if the Republish Scheme option is used on the Configuration Manager Admin Applet. Setting this value to FALSE is normally used in clusters where Schema Publisher should be enabled only on selected nodes.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
EnableSchemaPublish=false
```

encodeHtml

Idoc function used to filter data input for illegal or corruptive HTML constructs. This function is used by default in Threaded Discussions, a component available during installation.

All input data received by the Content Server when using the `unsafe` value for the `rule` parameter applies only to well-known unsafe script tags. This functionality can be altered by using the `HtmlDataInputFilterLevel` configuration variable to change the filtering that is done.

Type and Usage

- "Global Functions" on page 3-11
- "Security" on page 4-18

Parameters

Takes two required and one optional parameter:

- The first parameter is the string to encode.
- The second parameter is the rule to apply when encoding HTML constructs. The following values are allowed:
  - `none`: no conversion is done to HTML constructs.
  - `unsafe`: only well-known unsafe script tags are encoded.
  - `exceptsafe`: well-known safe script tags are not encoded.
  - `lfexceptsafe`: recommended for use when extended comments are entered by users and line breaks in the original text must be preserved.

  See the security and user access chapter in *Oracle Fusion Middleware System Administrator's Guide for Content Server* for details about comment handling, attribute values, parenthesis handling, and allowed values for unsafe and safe script tags.
- An optional parameter is also available that specifies if long strings without space characters are to be broken up and what maximum word size to apply. Specify either wordbreak or nowordbreak.

Output

Returns the encoded string.

See Also

- "HtmlDataInputFilterLevel" on page 6-72

entryCount

Retrieves the number of times the current workflow step has been entered by the current revision.

- This variable can be used to create conditional statements, but it should not be hard-coded or altered.
- This variable is localized in the companion file and maintained in the key:

  ```
  <step_name>@<workflow_name>.entryCount
  ```

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the entry count as an integer.

Example

The following code defines a jump called *MaxEntry*, which exits to the parent workflow and notifies the reviewers if the last time the step was entered was more than one week ago:

```
<$if wfCurrentGet("entryCount")==2$>
    <$wfSet("WfJumpName","MaxEntry")$>
    <$wfSet("WfJumpTargetStep",wfExit(0,0))$>
    <$wfSet("WfJumpEntryNotifyOff","0")$>
<$endif$>
```

ExclusiveCheckout

Determines whether users can check out content that was authored by another user.

- When set to TRUE, only the author or a user with Admin permission to the security group can check out a content item.
- When set to FALSE, users with Write permission to the security group can check out content that was authored by another user.
- Default is FALSE.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Location

- System Properties, Content Security tab, Allow only original contributor to check out
- Admin Server, Content Security, Allow only original contributor to check out
- *IntradocDir*/config/config.cfg

Example

As a configuration setting, only original contributor can check out a content item.

```
ExclusiveCheckout=true
```

As script, returns the value of the configuration setting:

```
<$ExclusiveCheckout$>
```

See Also

- "isUserOverrideSet" on page 6-130

ExpireRevisionOnly

Determines if all revisions or only a specific revision of a content item is expired on the specified Expiration Date.

> **Note:** This variable applies to Content Server 6.2 and later. Before version 6.2, only the specific revision expired on the specified Expiration Date. Starting with version 6.2, the default behavior is that if the current revision expires, all revisions of that content item also expire. Setting this variable to TRUE returns the system to the pre-6.2 behavior.

- When set to TRUE, only the current revision expires, and the most recent unexpired revision becomes the current revision.
- When set to FALSE, all revisions expire when the current revision expires.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ExpireRevisionOnly=true
```

fileUrl

Retrieves the relative URL of the current dynamic server page (HCSP or HCST).

This variable is typically used in self-referencing pages, such as a form that posts back to itself.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Items" on page 4-3

Output

Returns the URL as a string.

Example

Returns the relative URL of the current dynamic server page:

```
<$fileUrl$>
```

FIRSTREV

Returns the first revision label for the current revision label sequence.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Items" on page 4-3

Output

Returns the first revision label as a string.

Example

Returns the first revision label (default is 1):

```
<$FIRSTREV$>
```

ForceDistinctRevLabels

Defines whether additional revisions for the same content item can have the same revision label.

- Changing this setting will apply to new content only.
- When set to TRUE, two revisions of the same content cannot have the same revision label.
- Default setting is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ForceDistinctRevLabels=true
```

ForceJdbcSync

Restricts access to the database to one query at a time.

This setting is typically used for debugging or for problematic JDBC drivers.

If the system is being used mostly for read-only type activity, setting this to FALSE will improve the throughput to the database.

When set to TRUE, one query at a time can access the database.

When set to FALSE, multiple queries can access the database at a time.

Default setting is FALSE. (Default for version 4.5 and earlier was TRUE.)

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Database" on page 4-8
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ForceJdbcSync=true
```

formatDateDatabase

Formats the date and time in preparation for an SQL query. Long-formatted dates cannot be evaluated (for example, May 22, 2000).

Type and Usage

- "Date and Time" on page 4-8

Parameters

The only parameter is a string or variable that specifies a date/time.

Output

- Returns an ODBC-formatted date/time:

  ```
  [ts 'yyyy-mm-dd hh:mm:ss']
  ```

- Returns null if the parameter cannot be evaluated.
- If a time is not provided, returns a default time of 00:00:00.

Example

Formats the current date and time for an SQL query:

```
<$formatDateDatabase(dateCurrent())$>
```

Formats the date and time and displays as 2001-03-19 15:32:00:

```
<$formatDateDatabase("03/19/2001 3:32 PM")$>
```

Formats the date and time and displays as 1999-04-03 00:00:00:

```
<$formatDateDatabase("4/3/99")$>
```

FormInheritMetaDataFieldList

Defines the custom metadata fields that are inherited by subsequent revisions of a PDF form.

This is a comma-delimited list.

Standard metadata fields (such as Type, Security Group, and Account) are inherited by default.

Custom metadata fields begin with a lowercase x (for example, *xComments*).

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
FormInheritMetaDataFieldList=xDepartment,xRegion
```

GATEWAY_INTERFACE

Retrieves the revision level of the CGI specification to which this server complies. This variable is not request-specific; it is set for all requests.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the revision level as a string in the format `CGI/revision`.

Example

As information output on a page or to a log:

```
GATEWAY_INTERFACE=CGI/1.1
```

As part of an Idoc Script statement or evaluation:

```
<$if GATEWAY_INTERFACE$>
<!--statement-->
```

getCookie

Obtains a cookie from a browser.

This is useful for tracking user sessions when the information does not need to be stored in the personalization.

Can be used to track the last pages navigated to, or the last searches which were run.

Type and Usage

- "Setable Variables" on page 3-19
- "Web Server" on page 4-21

Parameters

This function takes one parameter, the name of the cookie.

Example

```
<$myCounter=getCookie("myCounter")$>
<$if not myCounter$>
    <$myCounter=1$>
<$endif$>
Num times loaded: <$myCounter$>
<$setCookie("myCounter", myCounter+1, dateCurrent(1))$>
```

See Also

- "setCookie" on page 6-239

GetCopyAccess

Determines what permission a user must have to get a copy of a content item.

When set to TRUE, users can get a copy of a content item for which they have only Read permission.

When set to FALSE, users must have Write permission to a content item to be able to get a copy.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18

Location

- System Properties, Content Security tab, Allow get copy for user with read privilege
- Admin Server, Content Security, Allow get copy for user with read privilege
- *IntradocDir*/config/config.cfg

Example

As a configuration setting:

```
GetCopyAccess=true
```

As script, returns the value of the configuration setting:

```
<$GetCopyAccess$>
```

getFieldConfigValue

This function returns a configuration flag for a specific field. If the field does not exist, the default is returned instead.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4

Parameters

Takes three parameters:

- The name of a metadata field.
- The flag to return for the field when it is found.
- A default value to return if the field does not exist.

Output

Returns the specified configuration flag.

Example

```
<$caption = getFieldConfigValue("dDocTitle", "fieldCaption", lc("wwTitle"))$>
```

getFieldViewDisplayValue

Returns the display value for an item in a Schema option list.

For example, assume an option list exists for a custom metadata field named `Customer`. In the database there is a schema table with the `CustomerName` and a unique `CustomerID`. On checkin the `CustomerName` is visible to the user but the `CustomerID` is what is stored in the database. The `getFieldViewDisplayValue` function extracts the human-readable `CustomerName` based on the `CustomerID`.

Type and Usage

- "Global Functions" on page 3-11
- "Schema" on page 4-16

Parameters

Takes three parameters:

- The first parameter is the name of the field to be used for the option list.
- The second parameter is the name of the schema view used for the current field.
- The third parameter is the value assigned to the field.

Output

Returns the display value.

Example

```
<$customerName = getFieldViewDisplayValue("xCustomer", "Customer_View", "1234")$>
```

getParentValue

This function returns a parent value from a Schema view. This is needed before generating the dependent option list for the current field.

In most cases this function returns the same value as the internal parent value parameter because most option lists trigger their dependency on the unique key for the parent field. However, this is not a requirement for complex dependent choice lists.

Type and Usage

- "Global Functions" on page 3-11
- "Schema" on page 4-16

Parameters

Takes four parameters:

- The name of the schema view used for the current field.
- The name of the schema relation between this field and the parent field.
- The name of the parent field.
- The internal value for the parent field. This may be different than the value displayed on the page. This is often a unique number stored internally.

Example

```
<$parentValue=getParentValue("xState", "Country_State", "xCountry", "123")$>
```

getViewValue

Returns the display value for an item in a Schema option list.

For example, assume an option list exists for a custom metadata field named `Customer`. In the database there is a schema table with the `CustomerName` and a unique `CustomerID`. On checkin the `CustomerName` is visible to the user but the `CustomerID` is what is stored in the database. The `getFieldViewDisplayValue` function extracts the human-readable `CustomerName` based on the `CustomerID`.

Type and Usage

- "Global Functions" on page 3-11
- "Schema" on page 4-16

Parameters

Takes three parameters:

- The name of the schema view used for the current field.
- The value stored in the database for the ID.
- The column name associated with the view.

Output

Returns the display value.

Example

```
<$custName = getViewValue("Customer_View", "1234", "CustomerName")$>
<$custRegion = getViewValue("Customer_View", "1234", "CustomerRegion")$>
```

See Also

- "getFieldViewDisplayValue" on page 6-63

getViewValueResultSet

This function loads a schema table and places it on the page as a ResultSet named SchemaData. The column names in the ResultSet are the same as the names in the database table.

This function is most useful to obtain a list of dependent choices based on a parent value.

Type and Usage

- "Global Functions" on page 3-11
- "Schema" on page 4-16

Parameters

Takes three parameters:

- The name of the schema view used.
- The relation for the schema and the view.
- The value for the schema parent.

Example

Assume you have a schema table and view for a list of countries. The view is named `Country_view`..

To output the contents of that table to the page, use the following code.

```
<$getViewValuesResultSet("Country_View", "", "")$>
<$loop SchemaData$>
<$count = 0, num = rsNumFields("SchemaData")$>
    <$loopwhile count < num$>
        <$fieldName=rsFieldByIndex("SchemaData", count)$>
        <$fieldName$> = <$getValue("SchemaData", fieldName)$>
        <$count = count + 1$>
    <$endloop$>
<$endloop$>
```

This will output the table even if you do not know the column name.

In the following example, assume you have a DCL for the fields `Country` and `State`. The list of `States` depends on which `Country` is selected. To obtain the list of `States` when the `Country` is USA, use this code:

```
<$getViewValuesResultSet("State_View", "Country_State", "USA")$>
```

hasAppRights

Checks if the current user has rights to an administrative application.

Type and Usage

- "Global Functions" on page 3-11
- "Internal Security" on page 4-18

Parameters

The only parameter is one of the following administrative applications:

- UserAdmin
- WebLayout
- RepoMan
- Workflow
- ConfigMan
- Archiver

Output

- Returns TRUE if the user has rights to the specified application.
- Returns FALSE if the user does not have rights to the specified application.
- Returns a StatusCode of `-1` if the value is unspecified or unknown.

Example

Evaluates whether the current user has rights to the specified application.

```
<$hasAppRights('RepoMan')$>
```

HasExternalUsers

Indicates to the Content Server that an external user database is present.

When set to TRUE, the system recognizes external users. If a custom component has been written to support an external user integration (such as LDAP), this should be set to TRUE.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "External Security" on page 4-19

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
HasExternalUsers=true
```

As script, returns the value of the configuration setting:

```
<$HasExternalUsers$>
```

See Also

- "NtlmSecurityEnabled" on page 6-187

HasOriginal

Checks if an original file exists for a revision.

This variable checks for a *dOriginalName* value.

It is possible for a content item to be checked in but to not have an original file (for example, when a Basic workflow has been started but files have not been checked in at the initial contribution step).

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Items" on page 4-3

Output

- Returns TRUE if an original file exists.
- Returns FALSE if no original file exists.

Example

Checks for an original file:

```
<$if HasOriginal and not isNew$>
```

HasPredefinedAccounts

Checks if the current user has permission to any predefined accounts. Predefined accounts are those that are created in the User Admin utility.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user is assigned to any predefined accounts.
- Returns FALSE if the user is not assigned to any predefined accounts.

Example

Displays the Accounts option list if the user is assigned to any predefined accounts:

```
<$if HasPredefinedAccounts$>
    <$fieldIsOptionList = 1, optionListName = "docAccounts",
    fieldOptionListType = "combo"$>
<$endif$>
```

HasUrl

Checks if a file exists in the *weblayout* directory for the current content item.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Items" on page 4-3

Output

- Returns TRUE if a weblayout file exists.
- Returns FALSE if no weblayout file exists.

Example

Checks for a web layout file:

```
<$if HasUrl$>
    <$include doc_url_field$>
<$endif$>
```

HtmlDataInputFilterLevel

A configuration entry used in conjunction with encodeHtml. This entry applies a level of encoding to filter all input to the system for bad HTML constructions.

All input data received by the system when using the `unsafe` value for the `encodeHtml` `rule` parameter applies only to well-known unsafe script tags. This functionality can be altered by using the `HtmlDataInputFilterLevel` configuration variable to change the filtering that is done.

Type and Usage

- "Configuration Variables" on page 3-2
- "Security" on page 4-18

Parameters

This entry takes one parameter, the filter level. Accepted values are:

- none: no filtering is performed.
- unsafe: protects against bad HTML constructions.
- exceptsafe: allows only well-known safe constructions through the filter.

Location

IntradocDir/config/config.cfg

See Also

- "encodeHtml" on page 6-49
- *Managing Security and User Access Guide*

HTTP_ACCEPT

Retrieves a list of content types that the browser can accept.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns a list of content types as a comma-delimited string.

Example

As information output on a page or to a log:

```
HTTP_ACCEPT=text/xml,application/xml,application/xhtml+xml,text/html;
q=0.9,text/plain;q=0.8,video/x-mng,image/png,image/jpeg,image/gif;
q=0.2,text/css,*/*;q=0.1
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_ACCEPT$>
<!--statement-->
```

HTTP_ACCEPT_ENCODING

Retrieves a list of compression encodings that the browser supports.

As of Content Server version 7.0, HTML files are compressed for delivery. GZIP compression is the default.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns a list of compression encodings as a comma-delimited string.

Example

As information output on a page or to a log:

```
HTTP_ACCEPT_ENCODING=gzip, deflate, compress;q=0.9
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_ACCEPT_ENCODING$>
<!--statement-->
```

HTTP_ACCEPT_LANGUAGE

Retrieves a list of ISO languages that are set for the browser.

This variable might be useful for estimating which user locale to set for new or guest users.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns a list of languages as a comma-delimited string.

Example

As information output on a page or to a log:

```
HTTP_ACCEPT_LANGUAGE=en-us,en;q=0.50
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_ACCEPT_LANGUAGE$>
<!--statement-->
```

HTTP_COOKIE

Retrieves the name/value pair of the cookie in the HTTP request header.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns a string in the format `name1=string1; name2=string2`.

Example

As information output on a page or to a log:

```
HTTP_COOKIE=IntradocAuth=Basic; IntrdocLoginState=1
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_COOKIE$>
<!--statement-->
```

HTTP_HOST

Retrieves the name of the Web server.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the Web server name as a string.

Example

As information output on a page or to a log:

```
HTTP_HOST=centralserver
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_HOST$>
<!--statement-->
```

HTTP_INTERNETUSER

Retrieves the CGI parameter that the Web server security filter passes to the system so that the system can set the user.

If this variable is not set, the value of the REMOTE_USER variable is used.

If HTTP_INTERNETUSER and REMOTE_USER are not set, the user is anonymous.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the user name as a string.

Example

As information output on a page or to a log:

```
HTTP_INTERNETUSER=admin
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_INTERNETUSER$>
<!--statement-->
```

HTTP_REFERER

Retrieves the complete URL of the referenced directory on the local server.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the complete URL as a string.

Example

As information output on a page or to a log:

```
HTTP_REFERER=http://centralserver/domain/
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_REFERER$>
<!--statement-->
```

HTTP_USER_AGENT

Retrieves the client browser type, version number, library, and platform the browser is configured for.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns a string in the format `software/version (library) (platform).`

Example

As information output on a page or to a log:

```
HTTP_USER_AGENT=Mozilla/4.7 [en] (WinNT; U)
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_USER_AGENT$>
<!--statement-->
```

HttpIgnoreWebServerInternalPortNumber

When set to TRUE, causes the HTTP address computed for the Admin Server and applets to **not** use the server port value provided by the Web server.

This prevents a load balancing configuration problem in which the load balancing server is using a different port number than the internal Web server actually delivering the response to the request.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Web Server" on page 4-21

Location

IntradocDir/config/config.cfg

Example

```
HttpIgnoreWebServerInternalPortNumber=true
```

See Also

- "HttpBrowserFullCgiPath" on page 8-129

IDC_Name

Defines the unique name of the Content Server instance.

> **Caution:** Using duplicate *IDC_Names* will cause data corruption. The Archiver cannot be used to move or copy data between two instances that share the same *IDC_Name*. To do so will corrupt the data on the target system. Give each of your instances a unique *IDC_Name*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IDC_Name=OracleMSP
```

See Also

- "InstanceDescription" on page 6-99
- "InstanceMenuLabel" on page 6-100

IdcAuthExtraRequestParams

Specifies values for the Web server plug-in to get from its local data and send in the header of its requests.

For example, if you have a plug-in to the Web server filter getting some attributes for a user and placing them into a local value, you could specify the name of the attributes in *IdcAuthExtraRequestParams*, and the data would be passed to the Content Server.

This setting can also be used to access extra HTTP header values using Idoc Script. These variables will become part of the Data Binder environment variable set.

- The names in the *IdcAuthExtraRequestParams* list must match the header names without regard for case (case insensitive match).
- When used as Idoc Script variables, the names in the *IdcAuthExtraRequestParams* must match exactly (case sensitive match).

There is no default value.

See the *IdcCommand Reference Manual* for more information.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21

Location

IntradocDir/config/config.cfg

Example

The following setting tells the IIS Web server plug-in to send the local value of HTTP_REFERER in the header (IIS automatically adds the *HTTP_* prefix):

```
IdcAuthExtraRequestParams=referer
```

IdcCommandFile

Specifies a file containing commands to execute using IdcCommand.

There is no default setting.

See *Oracle Fusion Middleware System Administrator's Guide for Content Server* for more information.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
IdcCommandFile=c:/ucm/idc/add_user.txt
```

IdcCommandLog

Specifies the path to the log file for IdcCommand execution.

The log file contains the time the command was executed and the success or failure status of the command.

There is no default setting.

See the *Oracle Fusion Middleware System Administrator's Guide for Content Server* for more information.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
IdcCommandLog=c:/idc/idccommand.log
```

IdcCommandServerHost

Specifies the Web server address of the Content Server that IdcCommand is connecting to.

Default is the HttpServerAddress of the Content Server that IdcCommand is connecting to.

See the *IdcCommand Reference Manual* for more information.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IdcCommandServerHost=instance_name.company.com
```

See Also

- "HttpServerAddress" on page 8-139

IdcCommandUser

Specifies the user that is permitted to execute commands using IdcCommand.

See the *Oracle Fusion Middleware System Administrator's Guide for Content Server* for more information.

There is no default setting.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
IdcCommandUser=sysadmin
```

IdcHttpHeaderVariables

Defines data parameters to include in HTTP header responses.

This is a comma-delimited list.

The format for encoding the name value pairs is the typical hda format after applying the Content Server HTTP header encoding.

The values are UTF-8 encoded and special characters are *%xx* encoded.

This setting is specifically designed for Web server plug-ins that want to audit the requests made by the client. The plug-ins can examine the HTTP headers in the responses, but not the body of the content. By pushing some of the parameters of the request into an HTTP header response, a plug-in can audit which documents were accessed and what actions were performed on them.

The name of the HTTP header generated by the Content Server when sending a response to an HTTP-based client is *IdcVariables*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21

Location

IntradocDir/config/config.cfg

Example

If `IdcHttpHeaderVariables=IdcService,dDocName`, then prior to the UTF-8 encoding being applied, the header value might be:

```
@Properties LocalData
IdcService=GET_DOC_PAGE
dDocName=TestDoc
blDateFormat=M/d{/yy}{ h:mm[:ss]{ a}}!mAM,PM!tAmerica/New_York
blFieldTypes=
@end
```

The UTF-8 encoding would leave all the above characters alone, except that it would change every line feed character into the `%0A` characters. The `blDateFormat` is the system date format for the Content Server.

IndexableRenditions

A comma-delimited list of single character codes for renditions to be indexed other than the primary, alternate, or webviewable renditons. Typically only one letter will be used because only one rendition will be indexable (such as the XML rendition).

This is used in conjunction with `IsAllowIndexRenditions`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

See Also

- "IsAllowIndexRenditions" on page 6-104

IndexerAutoWorkInterval

Defines the interval in seconds at which the Indexer automatic update cycle checks for changes.

Default is 300 (5 minutes).

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IndexerAutoWorkInterval=600
```

IndexerLargeFileSize

Determines the file size in megabytes at which the Indexer will place the file in its own bulkload (batch) file.

Files larger than this setting are indexed in a separate bulkload. Files smaller than this setting will be indexed in a batch with other files.

Default is 1 megabyte.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IndexerLargeFileSize=2
```

IndexerRebuildStagingManualCopy

Enables an index rebuild to create the collection in another location.

This variable is typically used when rebuilding extremely large collections. When the rebuild is finished, the results can be merged back to the collection manually.

When set to TRUE, the rebuilt index is created in the location set by the IndexerRebuildStagingPath value.

When set to FALSE, the rebuilt index is created in the next active index directory.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IndexerRebuildStagingManualCopy=true
```

See Also

- "IndexerRebuildStagingPath" on page 6-93

IndexerRebuildStagingPath

Defines the path where an Indexer rebuild will be created.

This variable is typically used when rebuilding extremely large collections. When the rebuild is finished, the results can be merged back to the collection manually.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IndexerRebuildStagingPath=c:/buildX/
```

See Also

- "IndexerRebuildStagingManualCopy" on page 6-92

indexerSetCollectionValue

This function modifies the logic of the search indexer. This function is designed to allow calculation of cumulative statistics about the index collection.

Type and Usage

- "Global Functions" on page 3-11
- "Content Items" on page 4-3

Parameters

Takes two parameters:

- The first parameter is the name of the flag used.
- The second parameter is the value attributed to the flag.

Any name and value can be used. These values become available the next time the resource include is executed for the index collection.

IndexerTouchMonitorInterval

Defines the time that the Content Server will wait before it will treat an Indexer lock file whose timestamp has not changed as having expired. The system will also touch any locks it is maintaining twice during this interval.

An Indexer lock is created by placing a lock file into an appropriate directory. In order for the lock to persist, the timestamp on the file must change periodically. Otherwise the Content Server will treat the lock as an expired lock.

The smaller the value of this setting, the quicker that locks can alternate between Indexing cycles. The larger the value, the more likely that high activity (which could keep a touch request delayed for many seconds) will cause the Content Server to falsely report a lock as having expired. If a lock is falsely expired, it can cause duplicated effort (although it should not have any long term impact on the state of the Content Server).

Default is 12000 milliseconds (12 seconds).

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
IndexerTouchMonitorInterval=15000
```

See Also

- "IndexerTraceCycleListChanges" on page 6-96
- "IndexerTraceLockFileChanges" on page 6-97

IndexerTraceCycleListChanges

Dumps trace information to the console for changes to the list of active indexing cycles maintained by the Content Server.

At certain points in an indexing cycle, an exclusive lock is required, so indexing cycles may have to wait for other cycles to become inactive. Enabling this setting can help you audit the performance of the Indexer.

When set to TRUE, changes to the indexing cycle list are output to the console.

When set to FALSE, changes to the indexing cycle list are not output.

Default is FALSE.

This variable is obsolete as of the 7.0 release. Functionality is now available on the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IndexerTraceCycleListChanges=true
```

See Also

- "IndexerTouchMonitorInterval" on page 6-95
- "IndexerTraceLockFileChanges" on page 6-97

IndexerTraceLockFileChanges

Dumps trace information to the console for changes the Content Server makes to the Indexer lock and suspension files.

When set to TRUE, changes to the files are output to the Content Server console.

When set to FALSE, changes to the files are not output.

Default is FALSE.

This variable is obsolete as of the 7.0 release. Functionality is now available on the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IndexerTraceLockFileChanges=true
```

See Also

- "IndexerTouchMonitorInterval" on page 6-95
- "IndexerTraceCycleListChanges" on page 6-96

IndexerTraceStartStop

When set to TRUE, traces the starting and stopping of index cycles.

Default is FALSE.

This variable is obsolete as of the 7.0 release. Functionality is now available on the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IndexerTraceStartStop=true
```

See Also

- "IndexerTouchMonitorInterval" on page 6-95
- "IndexerTraceCycleListChanges" on page 6-96

InstanceDescription

Defines a description for the instance.

The instance description is used in the Content Server interface.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

- System Properties, Server tab, Instance Description
- *IntradocDir*/config/config.cfg

Example

As a configuration entry:

```
InstanceDescription=Master_on_Server1
```

As script, returns the server instance description as a string:

```
<$InstanceDescription$>
```

See Also

- "IDC_Name" on page 6-82
- "InstanceMenuLabel" on page 6-100

InstanceMenuLabel

Defines the label for the instance that is used in the Windows Start menu.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

System Properties, Server tab, Instance Menu Label

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
InstanceMenuLabel=Master_on_Server1
```

See Also

- "IDC_Name" on page 6-82
- "InstanceDescription" on page 6-99

IntradocRealm

Defines the realm used when the Content Server or the Web server plug-in challenge for Basic authentication.

In browsers, the realm is usually called a *resource* when it is displayed in the login dialog.

NTLM does not use a realm; authentication is always against the entire Microsoft network.

Default is *IDC Security.*

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21
- "External Security" on page 4-19

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IntradocRealm=main
```

See Also

- "NtlmSecurityEnabled" on page 6-187

IntradocServerHostName

Defines the host name to use when opening a socket connection to the Content Server.

This setting is used by the CGI code that translates calls from the Web server to requests to the Server.

Default value is *localhost*.

To call services remotely, IntradocServerHostName must be set to IP or DNS.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IntradocServerHostName=IP
```

See Also

- "HttpServerAddress" on page 8-139

IntradocServerPort

Defines the port that the Web server filter or any other application should use to talk to the Content Server.

This setting tells the CGI code that translates from a call to the Web server to a request to the Content Server to use this port when talking to the Content Server. The CGI code will pick up this value automatically if the Web server is stopped and started after the Content Server has been stopped and started.

In IIS, the CGI code is implemented by the iis_idc_cgi.dll ISAPI extension.

Default is 4444.

To call services remotely, the server port must be defined.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IntradocServerPort=5550
```

IsAllowIndexRenditions

If set to TRUE, allows extra renditions of a document. This is only used if there are no indexable vault renditions or Web-viewable renditions.

The accepted rendition types are listed in `IndexableRenditions`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

See Also

- "IndexableRenditions" on page 6-89

IsAutoNumber

Enables automatic numbering of Content IDs.

When set to TRUE, a Content ID (*dDocName*) is automatically assigned to each new content item upon checkin.

When set to FALSE, the user must specify a Content ID for each new content item.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

- System Properties, Options tab, Automatically assign a Content ID on check in
- Admin Server, General Configuration, Automatically assign a content ID on check in
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
IsAutoNumber=true
```

See Also

- "AutoNumberPrefix" on page 6-17

IsAutoSearch

Enables automatic indexing and archive export.

When set to TRUE, the Content Server will automatically index content items or automatically export documents to an archive.

When set to FALSE, the system will not automatically index content items or automatically export documents to an archive.

The NoAutomation setting overrides this setting.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IsAutoSearch=true
```

See Also

- "NoAutomation" on page 6-179

IsCheckinPreAuthed

Checks if a checkin application pre-authorized the current checkin by getting a security token.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the checkin is pre-authorized.
- Returns FALSE if the checkin is not pre-authorized.

Example

Checks for checkin pre-authorization:

```
<$IsCheckinPreAuthed$>
```

isComponentEnabled

Checks if the defined component in enabled.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Server" on page 4-4

Parameters

The only parameter is the component name.

Output

- Returns TRUE if the defined component is enabled.
- Returns FALSE if not enabled.

Example

```
<$isComponentEnabled("Folders")$>
```

IsContributor

Used to evaluate whether a user is a contributor. Generally used to determine whether to show special links on a page (for example, std_page uses it to decide whether to display the Content Manager link).

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Server" on page 4-4

Output

- Returns TRUE if the user is a contributor.
- Returns FALSE if the user is not a contributor.

Example

Used to evaluate whether a user is a contributor:

```
<$if IsContributor$>
```

IsCriteriaSubscription

Evaluates whether a subscription to the content item is criteria-based rather than based on the Content ID (*dDocName*).

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Items" on page 4-3

Output

- Returns TRUE if the subscription is criteria-based.
- Returns FALSE if the subscription is to a particular content item.

Example

Evaluates whether subscription is criteria based:

```
<$IsCriteriaSubscription$>
```

IsEditRev

Checks whether the current revision is in an Editor step in a workflow.

When set to TRUE, reviewers can check out, edit, and check in the revision.

This variable is set by defining a workflow step as an Editor step in the Workflow Admin tool. This variable is evaluated on the checkin page.

Type and Usage

- "Value Variables" on page 3-21
- "Content Items" on page 4-3
- "Workflow" on page 4-21

Output

- Returns TRUE if the current revision is in an Editor step in a workflow.
- Returns FALSE if the current revision is not in an Editor step in a workflow.

Example

Provides workflow details:

```
<$if IsEditRev$>
    addCheckinValue("IdcService", "WORKFLOW_CHECKIN");
if (form.isFinished.checked)
    addCheckinValue("isFinished", form.isFinished.value);
<$else$>
    addCheckinValue("IdcService", "CHECKIN_SEL");
<$endif$>
```

IsFailedIndex

Checks if the Indexer has failed to index the content item.

Type and Usage

- "Value Variables" on page 3-21
- "Content Items" on page 4-3
- "Indexing" on page 4-13

Output

- Returns TRUE if the Indexer was unable to index the content item.
- Returns FALSE if the content item was indexed successfully.

Example

Displays text if the content item was not indexed:

```
<$if IsFailedIndex $>
    <p><font face="arial" size="2">
    Unable to index content item.</p>
<$endif$>
```

IsFilePresent

Checks if the page currently being displayed is for the revision being looped over while building the Revision History table on a content information page.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Items" on page 4-3

Output

- Returns TRUE if the loop is at the same revision as the current revision.
- Returns FALSE if the loop is not at the same revision as the current revision.

Example

If the loop is at the current revision, the revision number is displayed as plain text. If the loop is not at the current revision, the revision number is displayed as an active button.

```
<@dynamichtml doc_rev_info@>
<$if IsFilePresent$>
  <td width=10% align=center><span class=strongHighlight><$dRevLabel$></span></td>
  <td nowrap width=30%><span class=strongHighlight><$dInDate$></span></td>
  <td nowrap width=30%><span class=strongHighlight>
    <$if dOutDate$><$dOutDate$><$else$><$lc("wwNone")$><$endif$></span></td>
      <td width=20%><span class=strongHighlight>
      <$rptDisplayMapValue("StatusList", dStatus)$></span></td>
    <$else$>
  <td width=10%><form action="<$HttpCgiPath$>" method=GET>
  <input type=hidden name=IdcService value="DOC_INFO">
  <input type=hidden name=dID value="<$dID$>">
  <input type=hidden name=dDocName value="<$dDocName$>">
  <input type=submit value="       <$dRevLabel$>      ">
  </form></td>
...
<@end@>
```

IsFormsPresent

Specifies if the PDF Forms software is installed.

This value is set to FALSE during initial installation.

This value is set to TRUE during PDF Forms installation.

Note: This variable is deprecated. Use IsContentRefineryPresent instead.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IsFormsPresent=false
```

See Also

- "IsContentRefineryPresent" on page 10-53

IsFullTextIndexed

Checks if the Indexer has full-text indexed the content item.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Items" on page 4-3
- "Indexing" on page 4-13
- "Content Items" on page 4-3

Output

- Returns TRUE if the content item has been full-text indexed.
- Returns FALSE if the content item has not been full-text indexed.

Example

Provides a specified URL if the content item is full-text indexed:

```
<$if IsFullTextIndexed$>
    <a href="<$redirect$>">
<$endif$>
```

IsJdbcLockTrace

Dumps trace information to the console for database lock activity.

Used only with SQL Server and Oracle databases.

On Solaris, this output also appears in a log file in the *instance_dir*/etc/log directory.

Default is an empty string.

This variable is obsolete as of the 7.0 release. The functionality is available on the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IsJdbcLockTrace=true
```

See Also

- "IsJdbcQueryTrace" on page 6-117

IsJdbcQueryTrace

Dumps trace information to the console for database query activity.

Used only with SQL Server and Oracle databases.

On Solaris, this output also appears in a log file in the *instance_dir*/etc/log directory.

Default is an empty string.

This variable is obsolete as of the 7.0 release. The functionality is available on the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IsJdbcQueryTrace=true
```

See Also

- "IsJdbcLockTrace" on page 6-116

isLayoutEnabled

Used to determine if a particular layout is installed and enabled.

Type and Usage

- "Global Functions" on page 3-11
- "Content Server" on page 4-4

Parameters

Takes one parameter, the ID of the layout.

Example

In the std_resources.htm file:

```
<$if isLayoutEnabled("Trays")$><$do Publish=1$><$endif$>
```

IsLocalSearchCollectionID

Checks if the content item is in the local search collection. Searches for the content item's Content ID in the local search collection.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Items" on page 4-3
- "Content Items" on page 4-3

Output

- Returns TRUE if the content item is in the local collection.
- Returns FALSE if the content item is not found in the local collection.

Example

Evaluates whether a content item is from a local collection:

```
<@dynamichtml searchapi_define_result_doc_parameters@>
<$exec IsLocalSearchCollection="1"$>
<$if not IsLocalSearchCollectionID$>
    <!--Collection has external ID-->
<$exec IsLocalSearchCollection=""$>
```

IsLoggedIn

Checks if the current user is logged in.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user has logged in.
- Returns FALSE if the user has not logged in.

Example

Checks whether the user is logged in and has an e-mail address before performing a function.

```
<@dynamichtml subscription_action_script@>
function allowSubscription(form)
    {<$if IsLoggedIn$>
    <$if IsUserEmailPresent$>
    <$else$>
…}
<@end@>
```

IsNotLatestRev

Checks if the revision is the last revision to be checked in. This is not necessarily the last released revision.

Type and Usage

- "Value Variables" on page 3-21
- "Content Items" on page 4-3

Output

- Returns TRUE if the content item is other than the latest revision to be checked in.
- Returns FALSE if the content item is the latest revision to be checked in.

Example

Checks for the latest revision:

```
<$IsNotLatestRev$>
```

See Also

- "ClientControlled" on page 8-24
- "HasLocalCopy" on page 8-119

IsNotSyncRev

Checks whether the file on the client computer matches the most current revision by performing a revision ID (dID) comparison.

This variable is generally used to display an error message when the local copy of a content item has not been updated to the latest revision.

This variable is used to interface with client-side products, such as Desktop.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Items" on page 4-3

Output

- Returns TRUE if revisions do not match.
- Returns FALSE if revisions match.

Example

Checks for a match with the latest revision and displays an error message:

```
<$if IsNotSyncRev$>
The local copy of this content item has not been updated to the latest revision.
Use Get Native File or Check Out to update your local copy of <$dDocName$>.
<$endif$>
```

IsPrimaryUserBase

Specifies whether the LDAP provider is the primary LDAP server where external users are defined.

When set to TRUE, the provider will be used to authenticate new LDAP users who have not previously logged in to the Content Server.

When set to FALSE, the provider will be used to authenticate only LDAP users who have previously logged in using credentials from that provider.

Default is TRUE.

This setting is obsolete for version 7.0 and later. See Priority for information on setting the order in which LDAP providers will be checked.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, Primary User Base
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
IsPrimaryUserBase=false
```

IsPromptingForLogin

Checks if the Content Server is set to prompt for login or if login is being handled programmatically.

This variable is set to TRUE in situations such as cookie login pages, where the last request failed because the user is not logged in yet.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "External Security" on page 4-19
- "Content Server" on page 4-4

Output

- Returns TRUE if the Content Server is set to prompt for login.
- Returns FALSE if login is being handled programmatically.

Example

Evaluates if server is set to prompt for login:

```
<$IsPromptingForLogin$>
```

isRepromptLogin

This workflow configuration entry is a suffix appended to a workflow step, forcing a new authentication challenge.

When set to TRUE, this configuration variable forces a login at a specific step of a workflow. This acts as a digital signature for that step.

This variable can be added at multiple steps of a workflow.

Type and Usage

- "Workflow" on page 4-21

Location

IntradocDir/config/config.cfg

Example

In the following example, a review step named *VPApproval* requires a digital signature before continuing in the workflow. Set the `isRepromoptLogin` configuration entry as follows before initiating the workflow:

```
VPApproval:isRepromptLogin=true
```

See Also

- *The Workflow Implementation Guide*

IsSavedWfCompanionFile

Saves a copy of the companion file that is created when a piece of content enters a workflow. Normally the companion file is deleted when the content is approved and exits the workflow.

To use this variable, insert the variable as shown into the config.cfg file and restart the Content Server. The file will be saved in the *instance_dir*/data/workflow/saved directory.

Type and Usage

- "Configuration Variables" on page 3-2
- "Workflow" on page 4-21

Location

IntradocDir/config/config.cfg

Example

As a configuration variable:

```
IsSaveWfCompanionFiles=true
```

IsSubAdmin

Checks if the current user has subadministrator rights to at least one administrative application.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user has subadministrator rights.
- Returns FALSE if the user does not have subadministrator rights.

Example

Checks whether the user is logged in and has subadministrator rights before performing a function.

```
<@dynamichtml subscription_action_script@>
    function allowSubscription(form)
    {
    <$if IsLoggedIn$>
    <$if IsSubAdmin$>
    <$else$>
    …
    }
<@end@>
```

IsSysManager

Checks if the current user has the `sysmanager` role, meaning the user has access to the Admin Server. This variable is usually used to conditionally display specific navigation links.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user has the `sysmanager` role.
- Returns FALSE if the user does not have the role

IsUploadSockets

Used by the Upload applet to determine whether the upload socket should be used.

This is an internal flag and is not intended for user configuration.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "External Security" on page 4-19
- "Clients" on page 4-2

Output

- Returns TRUE if the upload socket is defined for use with the Upload applet.
- Returns FALSE if the upload socket should not be used.

Example

N/A

isUserOverrideSet

Enables users to check in content for other users.

This affects the Author option list on checkin pages. By default, only administrators are allowed to specify another user as the Author during checkin.

Type and Usage

- "Global Functions" on page 3-11
- "Internal Security" on page 4-18
- "Users" on page 4-20

Parameters

The only parameter is the value TRUE or FALSE.

Output

Evaluates to TRUE if the user override is enabled.

Example

Enables users to check in content items with another user specified as the Author:

```
<$isUserOverrideSet(true)$>
```

See Also

- "ExclusiveCheckout" on page 6-51

isValidateFile

Used as a parameter to the `Upload` applet. This variable verifies that the file to be uploaded exists. In order to be used, a component must be created that overwrites the applet definition as defined in the `std_multiupload_applet_definition` include in the std_page.htm resource file.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Example

Used as a configuration entry:

```
isValidateFile=true
```

IsWebServerPagesOnly

Used with the ExtranetLook component. This variable determines if cookie-based authentication is used or not. See the *Oracle Fusion Middleware System Administrator's Guide for Content Server* for details about using the ExtranetLook functionality.

When `IsWebServerPagesOnly` is set to TRUE, basic authentication is used and cookie-based logon/logoff functionality is disabled. Customized versions of the Web server filter are used.

If set to FALSE, cookie-based authentication is used.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

- *IntradocDir*/config/config.cfg

Example

```
IsWebServerPagesOnly=FALSE
```

See Also

- "CookieAuthTimeoutInDays" on page 6-25
- "CookieProxyPassword" on page 6-26

IsWorkflow

Checks if the content item on a checkin page is in a workflow.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Workflow" on page 4-21

Output

- Returns TRUE if the content item is in a workflow.
- Returns FALSE if the content item is not in the workflow.

isZoneSearchField

This function is used to specify a field for full text searching.

Type and Usage

- "Setable Variables" on page 3-19
- "Searching" on page 4-17

Parameters

The name of the field to be checked.

JdbcConnectionString

Defines the Java Database Connectivity (JDBC) connection (SQL Server Data Source Name), including the hostname, port number, and instance name.

This setting is automatically configured during installation if there are no network errors that make it impossible for the system to connect to the database. This setting must be changed manually if the database is changed or if the system could not find the database during installation.

If the run time version of Microsoft Access or MSDE is used, there are no database configuration options to set.

Default is *JdbcConnectionString=JDBC:ODBC:SQLForInstance_name.*

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

- System Properties, Database tab, JDBC Connection String
- *IntradocDir*/config/config.cfg

Example

Defines the JDBC connection path:

```
JdbcConnectionString=jdbc:oracle:thin:@hostname:port_number:instance_name
```

See Also

- "JdbcDriver" on page 6-136

JdbcDriver

Defines the Java Database Connectivity (JDBC) device driver name.

This setting is automatically configured during installation if there are no network errors that make it impossible for the system to connect to the database. This setting must be manually changed if the database is changed or if the system could not find the database during the installation.

If the run time version of Microsoft Access or MSDE is used, there are no database configuration options to set.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

- System Properties, Database tab, JDBC Driver options/JDBC Driver Name
- *IntradocDir*/config/config.cfg

Example

Defines the JDBC device driver:

```
JdbcDriver=oracle.jdbc.driver.OracleDriver
```

See Also

- "JdbcConnectionString" on page 6-135

JdbcPassword

Defines the SQL Server database password.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

- System Properties, Database tab, JDBC User Password
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
JdbcPassword=password
```

See Also

- "JdbcUser" on page 6-139
- "JdbcPasswordEncoding" on page 6-138

JdbcPasswordEncoding

Defines the type of encoding to use when storing the SQL Server database password.

To disable password encryption, set this value to `ClearText`.

Default is *Intradoc*.

If any other nonempty value is assigned, no encoding is performed.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
JdbcPasswordEncoding=ClearText
```

JdbcUser

Defines the SQL Server database user name.

Default is *sa*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

- System Properties, Database tab, JDBC User Name
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
JdbcUser=sa
```

See Also

- "JdbcPassword" on page 6-137

jsFilename

Used by schema. Encodes a string that may contain non-ASCII characters into the valid filename strings required for the operating system and Java Script (performs an encoding function).

Type and Usage

- "Global Functions" on page 3-11
- "Schema" on page 4-16

Parameters

The only parameter is the string to be encoded.

Output

Returns an encoded string.

Example

```
<$jsFilename(fileName)$>
```

lastEntryTs

Retrieves the timestamp from the last time the workflow step was entered.

This variable can be used to create conditional statements, but it should not be hard-coded or altered.

The last entry time is localized in the companion file and maintained in the key:

```
<step_name>@<workflow_name>.lastEntryTs
```

Type and Usage

- "Workflow Variables" on page 3-21
- "Date and Time" on page 4-8

Output

Returns the timestamp in the format defined by dateCurrent.

Example

The following code defines a jump called *LastEntry*, which exits to the parent workflow and notifies the reviewers if the last time the step was entered was more than one week ago:

```
<$if parseDate(wfCurrentGet("lastEntryTs")) < parseDate(dateCurrent(-7))$>
    <$wfSet("WfJumpName","LastEntry")$>
    <$wfSet("WfJumpTargetStep",wfExit(0,0))$>
    <$wfSet("WfJumpEntryNotifyOff","0")$>
<$endif$>
```

LdapAdminDN

Specifies a user name that has Read permission for the LDAP server.

If this setting is not defined, the provider will connect to the LDAP server anonymously.

If the LDAP provider is communicating with Active Directory, a user name and password is required, and must be a valid domain user in the format *DOMAIN*\username.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, LDAP Admin DN
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
LdapAdminDN=ldap_admin
```

See Also

- "LdapAdminPassword" on page 6-143

LdapAdminPassword

Specifies the password for the LdapAdminDN setting, which specifies a user name that has Read permission for the LDAP server.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, LDAP Admin Password
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
LdapAdminPassword=idc2
```

See Also

- "LdapAdminDN" on page 6-142

LdapPort

Defines the port number the LDAP server listens on.

Default is 389, unless you are using SSL.

If you are using SSL (see UseSecureLdap), set this value to 636.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, LDAP Port
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
LdapPort=636
```

See Also

- "UseSecureLdap" on page 6-271

LdapServer

Host name of the LDAP server.

If the provider is communicating with Active Directory, this should be the host name of a primary domain controller.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, LDAP Server
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
LDAPServer=iplanet_dir
```

LdapSuffix

Specifies the root suffix (naming context) to use for all LDAP operations (such as *o=company.com* or *dc=company,dc=com*).

All mapping of LDAP groups to Content Server roles and accounts will begin at this root.

Do not include spaces before or after commas.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, LDAP Suffix
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
LdapSuffix=dc=company,dc=com
```

LkDisableOnRebuild

Used with the Link Manager functionality.

This variable controls link extraction during the indexing rebuild cycle.

If set to TRUE, this variable prevents Link Manager from extracting links during the index rebuild cycle. If set to FALSE, links are extracted.

The default is FALSE.

Because Link Manager does all of its work during the indexing cycle, it will increase the amount of time required to index documents and to rebuild collections.

However, the time taken may not be noticeable because most of the time is spent indexing the document into the collection. The amount of time required depends on the type and size of the documents involved. That is, if the file must be converted, this requires more time than text-based (HTML) files.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkDisableOnRebuild=TRUE
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.

LkDisableReferenceInfo

Used with the Link Manager functionality.

This variable enables or disables the Resources section on the Content Information page. If set to TRUE, it configures the Content Server to suppress the References section. Link Manager then does not compute the link references for presentation.

If set to FALSE, Content Server displays the Show and Hide toggle switches on the Content Information page. By clicking on the toggle switch, link references are displayed or concealed.

The default is FALSE.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkDisableReferenceInfo=TRUE
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.

LkDisallowConversionFormats

Used with the Link Manager functionality.

This variable provides a list of file formats that Link Manager disregards during the link extraction process. By default, this is empty.

There are some file formats (such as Word) that need to be converted by HtmlExport before links can be extracted. However, links in text-based file formats (HTML) can be extracted by Link Manager without requiring conversion by HtmlExport.

For example, you may have PHP files (or files in some other custom format) that are actually text-based. For such files, conversion by HtmlExport may not be necessary before Link Manager processes them for link extraction. Such formats can be listed in this configuration variable.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Parameters

The only parameter(s) are the formats.

Location

IntradocDir/config/config.cfg

Examples

In the following example, specifying the full MIME type format limits the variations of the listed types that are excluded. For example, listing `application/msword` will not exclude `application/vnd.msword` or other variations. In this case, you must include each specific MIME type variation in the list that you want to exclude.

```
LkDisallowConversionFormats=application/msword,audio/wav,video/avi
```

In the following example, all variations of each MIME type in the list are excluded. Using the shortened format of the MIME type is more advantageous in that it allows greater flexibility.

```
LkDisallowConversionFormats=msword,wav,avi
```

Important: Link Manager does not use HtmlExport to convert files that contain any of the following strings in the file format: hcs, htm, image, text, xml, jsp, and asp. These files are handled by Link Manager without need for conversion. To check the current file formats and extension mappings of your system, use the File Formats window in the Configuration Manager

See Also

- *Managing Repository Content Guide* for details about using Link Manager.

LkIsSecureSearch

Used with the Link Manager functionality.

This variable is used to manage security restriction checks for non-admin users during a link search.

If set to TRUE, this configures Link Manager to perform a security check on non-admin users while implementing a managed link search. This is the default setting.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkIsSecureSearch=TRUE
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.

LkReExtractOnRebuild

Used with the Link Manager functionality.

This variable controls link extraction from previously indexed documents during a rebuild. If set to TRUE, it configures Link Manager to extract links from documents as they are indexed into the system during a rebuild. This is the default setting.

If set to FALSE, this prevents Link Manager from extracting links from documents as they are indexed into the system during a rebuild.

Because Link Manager does all of its work during the indexing cycle, it will increase the amount of time required to index documents and to rebuild collections.

However, the time taken may not be noticeable since most of the time is spent indexing the document into the collection. Although, the amount of time required does depend on the type and size of the documents involved. That is, if the file must be converted, this requires more time than text-based (HTML) files.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkReExtractOnRebuild=FALSE
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.

LkRefreshBatchSize

Used to control the number of links submitted to Site Studio during a refresh process. By default, the batch size value is set to 100.

Depending on the value, this configuration variable can make the refresh processes more (or less) receptive to an abort request. The calculation for `LkRefreshErrorsAllowed` is completed after, not during, each batch process. Therefore, the lower the value for `LkRefreshBatchSize`, the more likely it is that the refresh activity will promptly terminate because the number of allowed errors is more quickly exceeded.

If you are using Site Studio, LkRefreshBatchSize enables you to have better control over the refresh abort option than using the combined LkRefreshErrorPercent and LkRefreshErrorThreshold settings. For example, if you set the percent value to 5 and the threshold value to 20, you would expect the refresh activity to abort after the first error. However, Link Manager might actually process multiple errors before aborting.

The reason for this is that during refresh activities, all the links recognized as Site Studio links (or those requiring Site Studio processing) are grouped and sent to Site Studio as a batch. Consequently, refreshes are more efficient, but abort requests are unresponsive during this time because Site Studio is unaware of aborts and total error counts.

However, Site Studio is aware of the number of errors it has encountered in the current batch. For this reason, the Link Manager's abort calculation can not take place in all situations and the error configuration values (percent and threshold) are simply suggestions to Link Manager for when an abort should occur. Using `LkRefreshBatchSize`, however, enables you to more accurately control the abort receptivity during refresh activities that involve Site Studio link batches.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshBatchSize=200
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.

LkRefreshErrorPercent

Used with Link Manager functionality.

This variable is used with `LkRefreshErrorThreshold` to compute if the refresh activity should be aborted. When the refresh activity processes more links than the threshold value, Link Manager calculates the proportion of errors. If the number of errors exceeds the percent value, the refresh activity is aborted.

The percent value is set to 10 by default.

> **Note:** If you choose to set a value for `LkRefreshErrorsAllowed`, it could override the combination of threshold and percent values. For example, if the value is less than the calculated threshold/percent value for links/errors, then the refresh activity could abort even if the number of links processed does not exceed the threshold limit. Therefore, it is recommended that you use either the `LkRefreshErrorsAllowed` configuration setting or the combination of the `LkRefreshErrorThreshold` and `LkRefreshErrorPercent` configuration settings.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshErrorPercent=20
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.
- "LkRefreshErrorsAllowed" on page 6-154
- "LkRefreshErrorThreshold" on page 6-155

LkRefreshErrorsAllowed

Used with Link Manager functionality.

This variable sets an absolute error count for refresh processes. When the set number of errors is encountered, the refresh activity aborts. This configuration setting is not used by default.

> **Note:** If you choose to set a value for `LkRefreshErrorsAllowed`, it could override the combination of threshold and percent values. For example, if the value is less than the calculated threshold/percent value for links/errors, then the refresh activity could abort even if the number of links processed does not exceed the threshold limit. Therefore, it is recommended that you use either the `LkRefreshErrorsAllowed` configuration setting or the combination of the `LkRefreshErrorThreshold` and `LkRefreshErrorPercent` configuration settings.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshErrorsAllowed=10
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.
- "LkRefreshErrorPercent" on page 6-153
- "LkRefreshErrorThreshold" on page 6-155

LkRefreshErrorThreshold

Used with Link Manager functionality.

This variable is used with `LkRefreshErrorPercent` to compute if the refresh activity should be aborted. When the refresh activity processes more links than the threshold value, Link Manager calculates the proportion of errors. If the number of errors exceeds the percent value, the refresh activity is aborted.

The threshold value is set to 100 by default.

> **Note:** If you choose to set a value for `LkRefreshErrorsAllowed`, it could override the combination of threshold and percent values. For example, if the value is less than the calculated threshold/percent value for links/errors, then the refresh activity could abort even if the number of links processed does not exceed the threshold limit. Therefore, it is recommended that you use either the `LkRefreshErrorsAllowed` configuration setting or the combination of the `LkRefreshErrorThreshold` and `LkRefreshErrorPercent` configuration settings.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshErrorThreshold=200
```

See Also

- *Managing Repository Content Guide* for details about using Link Manager.
- "LkRefreshErrorsAllowed" on page 6-154
- "LkRefreshErrorPercent" on page 6-153

LmDefaultLayout

Defines the default layout to use (for example, Top Menus or Trays).

The default installation value is `Trays`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

```
LmDefaultLayout=Top Menus
```

LmDefaultSkin

Defines the default skin to use (for example, Oracle, Oracle2).

Type and Usage

- "Global Functions" on page 3-11
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

```
LmDefaultSkin=Oracle
```

lmGetLayout

Retrieves the layout selected by the user (for example, Top Menus or Trays).

Type and Usage

- "Global Functions" on page 3-11
- "Content Server" on page 4-4

Parameters

None.

Example

```
<$lmGetLayout()$>
```

See Also

- "lmGetSkin" on page 6-159

lmGetSkin

Retrieves the skin selected by the user.

Type and Usage

- "Global Functions" on page 3-11
- "Content Server" on page 4-4

Parameters

None.

Example

```
<$lmGetSkin()$>
```

See Also

- "lmGetLayout" on page 6-158

loadCollectionInfo

Loads metadata for a search collection.

Used by the search service to load metadata about a search collection.

This function is not intended for user configuration.

Type and Usage

- "Global Functions" on page 3-11
- "Searching" on page 4-17

Parameters

Takes one parameter, the name of the search collection to be loaded.

Output

None.

Example

Loads search collection information:

```
<$loadCollectionInfo(collection_name)$>
```

loadSchemaData

Loads the data from a schema resultset into the local data of the current data binder.

Type and Usage

- "Global Functions" on page 3-11
- "Schema" on page 4-16

Parameters

This function can take zero, one, or two parameters.

- If passed zero arguments, it fills the data binder with the data in the active resultset's row. Returns no value.
- If passed one argument, the argument is the name of the resultset to get the values from the current row. Returns no value.
- If passed two arguments, the first argument is the resultset name and the second argument is the key identifying the data object to load. Returns 0 if the data does not exist or 1 if it does exist. The use is True() or False() to conditionally execute scripts based on the return value.

Example

```
<$loadSchemaData()$>
```

See Also

- "clearSchemaData" on page 6-20

MailServer

Defines the e-mail server that is used to process e-mails.

E-mail support is required for workflow notifications, subscription notifications, and project notifications in Oracle Collaboration Server.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

- System Properties, Internet tab, Mail Server
- Admin Server, Internet Configuration, Mail Server
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
MailServer=mail.company.com
```

See Also

- "SysAdminAddress" on page 8-318
- "SmtpPort" on page 6-244

MajorRevSeq

Defines the major sequence for revision numbers.

Returns the major revision label sequence (returns the value of the configuration setting).

Returns a string.

Default is *1-99*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

- System Properties, Options tab, Major Revision Label Sequence
- Admin Server, General Configuration, Major Revision Label Sequence
- *IntradocDir*/config/config.cfg

Example

As a configuration setting:

```
MajorRevSeq=A-Z
```

As script, returns the value of the configuration setting:

```
<$MajorRevSeq$>
```

See Also

- "MinorRevSeq" on page 6-178

MaxCollectionSize

Defines the number of files to be passed to the Indexer in one batch.

Valid range is 1 to 65535.

A value of 2000 is recommended for large index collections. Lower values will result in inefficient indexing performance.

Returns the number of files per batch.

Default is 25.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

- Repository Manager, Indexer tab, Configure, Content Items Per Indexer Batch
- *IntradocDir*/config/config.cfg

Example

As a configuration setting, defines the batch size:

```
MaxCollectionSize=25
```

As script, returns the value from configuration files:

```
<$MaxCollectionSize$>
```

MaxDocIndexErrors

Sets the number of errors after which the Indexer stops indexing or rebuilding. Default is 50.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxDocIndexErrors=50
```

MaxHitCount

Sets the maximum number of results that can be found during a search.

Default is 100000.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxHitCount=500000
```

MaxIntegerAllowed

Sets the maximum value allowed in custom Integer metadata fields.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
MaxIntegerAllowed=1000000
```

See Also

- "MinIntegerAllowed" on page 6-176

MaxQueryRows

Sets the maximum number of rows that are displayed in the Repository Manager, Active Report pages, and the Work In Progress page.

Increasing this value will slow response time.

Default is 200.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxQueryRows=50
```

See Also

- "IsMaxRows" on page 8-175

MaxRequestThreadCount

Sets the maximum number of simultaneous request threads.

Default is 100.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxRequestThreadCount=50
```

MaxResults

Sets the maximum number of content items returned from a search.

Specifying a value overrides the programmed default.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxResults=200
```

MaxSavedSearchResults

Sets the maximum number of saved searches a user can make. Specifying a value overrides the default (10).

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxSavedSearchResults=25
```

MaxSearchConnections

Sets the maximum number of search connections that are open simultaneously.

For example, if `MaxSearchConnections=5` and six search requests are made at the same time, the sixth request goes into a queue until one of the first five requests is completed.

Default is 10.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxSearchConnections=5
```

MaxSizeResourceFileCacheMillions

Sets the maximum size of resource files, in millions of double-byte characters.

This value limits the size of files before they are read in and parsed.

Multiply file sizes by 2.5 to get a better estimate of true size in memory.

Default is 10.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxSizeResourceFileCacheMillions=20
```

MaxStandardDatabaseResults

Sets the maximum number of users displayed.

This is valid for any applet that allows you to select users from a list.

Default is 500.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxStandardDatabaseResults=1000
```

MemoFieldSize

Defines the field size created in the database for memo fields.

Ensure that the database supports whatever size is chosen.

Default is 255 characters.

Type and Usage

- "Configuration Variables" on page 3-2
- "Field Display" on page 4-11
- "Database" on page 4-8

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MemoFieldSize=300
```

See Also

- "MinMemoFieldSize" on page 6-177

MinIntegerAllowed

Sets the minimum value allowed in custom Integer metadata fields.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
MinIntegerAllowed=0
```

See Also

- "MaxIntegerAllowed" on page 6-167

MinMemoFieldSize

Defines the size at which a custom metadata field is defined as a memo field rather than a long text field.

If the field sizes of the *DocMeta* table have been edited directly in the database, the Content Server classifies the *varchar* field types by their lengths.

Default is 255 characters.

The following field type rules apply when this setting is defined:

Field Type	Evaluation Rules
Text	<50
Long Text	>= 50 and < MinMemoFieldSize
Memo	>= MinMemoFieldSize

Type and Usage

- "Configuration Variables" on page 3-2
- "Field Display" on page 4-11
- "Database" on page 4-8

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MinMemoFieldSize=300
```

See Also

- "MemoFieldSize" on page 6-175

MinorRevSeq

Defines the minor sequence for revision numbers.

Returns the minor revision label sequence (returns the value of the configuration setting).

Returns a string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

- System Properties, Options tab, Minor Revision Label Sequence
- Admin Server, General Configuration, Minor Revision Label Sequence
- *IntradocDir*/config/config.cfg

Example

As a configuration setting:

```
MinorRevSeq=a-c
```

As script, returns the value of the configuration setting:

```
<$MinorRevSeq$>
```

See Also

- "MajorRevSeq" on page 6-163

NoAutomation

Disables all automated Content Server activity.

This setting overrides the IsAutoArchiver, IsAutoQueue, and IsAutoSearch settings.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
NoAutomation=true
```

See Also

- "IsAutoArchiver" on page 8-154
- "IsAutoQueue" on page 10-52
- "IsAutoSearch" on page 6-106

NoMatches

Checks whether matches were found from a search query.

Generally used to display a message on the search results page.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Searching" on page 4-17

Output

- Returns TRUE if no matches were found.
- Returns FALSE if any matches were found.

Example

Displays text if no matches were found from a query:

```
<$if NoMatches$>
    <p><font face="arial" size="2">
    Found no matches out of <$TotalDocsProcessed$> documents searched matching the 
query.</p>
<$endif$>
```

NoSearchConnectionCloseOnDocChange

Disables automatic closure of all search connections when a document has been modified (inserted, updated, or deleted).

> **Caution:** Setting this to TRUE may lead to incorrect search results when the system is under high stress.

When set to TRUE, search connections stay open even when a document is changed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
NoSearchConnectionCloseOnDocChange=true
```

NotificationIntervalInDays

This configuration variable defines how often a notification query is run. Notification queries define the criteria used to search for expired content.

If `NotificationIntervalInDays` is not set, the default value is one day.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

```
NotificationIntervalInDays=3
```

See Also

- "EnableExpirationNotifier" on page 6-46
- "NotificationMaximum" on page 6-183
- "NotificationQuery" on page 6-184
- "NotifyExtras" on page 6-185
- "NotifyTime" on page 6-186

NotificationMaximum

This configuration variable defines the maximum number of content items that will be returned by a notification query. Notification queries define the criteria used to search for expired content.

If `NotificationMaximum` is not set, the default value is 1000.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

```
NotificationMaximum=500
```

See Also

- "EnableExpirationNotifier" on page 6-46
- "NotificationIntervalInDays" on page 6-182
- "NotificationQuery" on page 6-184
- "NotifyExtras" on page 6-185
- "NotifyTime" on page 6-186

NotificationQuery

This setting defines criteria for the automatic query that searches for expired content.

If `NotificationQuery` is not set, the default value is all content that expires in the next seven days.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Parameters

There is one parameter, the query to be used. The query can be one of the following. See the EXAMPLES section for sample queries.

- An Idoc script query, built from Idoc script.
- A URL encoded query. This uses the URL displayed in the Web browser address when a search is performed.
- A plain text query which defines the search variables.

Location

IntradocDir/config/config.cfg

Example

Idoc Script Example

When used in conjunction with database indexing, the following query provides email notification for all documents that expire in seven days:

```
NotificationQuery=dOutDate < '<$dateCurrent(7)$>'>
```

URL Encoded Example

The following query returns all content expiring after August 1, 2007. The URL from the browser address line is copied, beginning with the `QueryText` portion:

```
NotificationQuery=QueryText=dOutDate+%3C+%608%2F1%2F06%60&SearchProviders= [...}
```

Plain Text Query

The following query returns all content expiring after August 1, 2007:

```
NotificationQuery=8/1/07
```

See Also

- "EnableExpirationNotifier" on page 6-46
- "NotificationIntervalInDays" on page 6-182
- "NotificationMaximum" on page 6-183
- "NotifyExtras" on page 6-185
- "NotifyTime" on page 6-186

NotifyExtras

This configuration variable defines the users who will receive a list of expired content in addition to the authors of each content item. Notification queries define the criteria used to search for expired content.

If `NotifyExtras` is not set, the default value is sysadmin. If the `NotifyExtras` setting is in the config.cfg file but has no value defined, no extra notification is sent.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

```
NotificationExtras=jwilson, nadams
```

See Also

- "EnableExpirationNotifier" on page 6-46
- "NotificationIntervalInDays" on page 6-182
- "NotificationMaximum" on page 6-183
- "NotificationQuery" on page 6-184
- "NotifyTime" on page 6-186

NotifyTime

This configuration variable defines the time of day the notification query is run, specified in 24-hour notation. Notification queries define the criteria used to search for expired content.

If `NotifyTime` is not set, the default value is midnight (00:01).

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

The following example sets the time to 11:30 AM:

```
NotifyTime=11:30
```

The following example sets the time to 1:30 PM:

```
NotifyTime=13:30
```

See Also

- "EnableExpirationNotifier" on page 6-46
- "NotificationIntervalInDays" on page 6-182
- "NotificationMaximum" on page 6-183
- "NotificationQuery" on page 6-184
- "NotifyExtras" on page 6-185

NtlmSecurityEnabled

Enables Windows network authentication for the Content Server.

This option is available only when using an IIS Web server.

When set to TRUE, users defined on a Windows network (NTLM) or in Active Directory can log in to the Content Server using their network credentials.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- System Properties, Internet tab, Use Microsoft Security
- Admin Server, Internet Configuration, Use Microsoft Security
- *IntradocDir*/config/config.cfg
- *instance_dir*/data/users/config/filter.hda

Example

Enables NTLM security:

```
NtlmSecurityEnabled=true
```

See Also

- "UseNtlm" on page 6-263
- "UseAdsi" on page 6-259
- "HasExternalUsers" on page 6-68
- "IntradocRealm" on page 6-101

NumConnections

Sets the number of open connections to the database. Default is 15. The default for standalone applets is 4.

This setting applies to the Content Server and any standalone applications and utilities, so each application will use the specified number of connections.

If your database does not allow enough connections to be open at one time, an error will occur when you attempt to start an application or utility.

To avoid this problem, you can isolate the *NumConnections* setting for the Content Server from the setting for applications and utilities as follows:

1. Specify the *NumConnections* setting for the Content Server in the *DomainHome*/ucm/cs/bin/intradoc.cfg file rather than *IntradocDir*/config/config.cfg.
2. Make a copy of the *DomainHome*/ucm/cs/bin directory and give it a unique name (such as *DomainHome*/ucm/cs/applet_bin/).
3. Set the *NumConnections* value for application and utilities in the intradoc.cfg file in the new directory (for example, *NumConnections=2*).
4. Run applications and utilities from the new directory (change any scripts and shortcuts in Windows), but continue to run the Content Server from the *DomainHome*/ucm/cs/bin/ directory.

If necessary, you can create a separate directory and *NumConnections* setting for each application and utility.

Type and Usage

- "Configuration Variables" on page 3-2
- "Database" on page 4-8

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
NumConnections=10
```

OneMatch

Checks if only one match was found from a search query.

Generally used to display a message on the search results page.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Searching" on page 4-17

Output

- Returns TRUE if only one match was found.
- Returns FALSE if no matches or more than one match were found.

Example

Displays text if only one match was found from a query:

```
<$if OneMatch$>
    <p><font face="arial" size="2">
    Found <$TotalRows$> document matching the query.</p>
<$endif$>
```

PATH_INFO

This setting is obsolete. The Web server filter no longer sends this information.

Retrieves additional information about the file system path to the CGI computer.

When the virtual path is returned by the PATH_TRANSLATED variable, any additional information at the end of this path is also returned as PATH_INFO.

This variable is specific to the current gateway program request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the additional virtual path information as a string.

Example

Retrieves additional CGI path information:

```
<$PATH_INFO$>
```

See Also

- "PATH_TRANSLATED" on page 6-191

PATH_TRANSLATED

Retrieves the file system path to the CGI computer, for example:

`c:/*domain*/weblayout/idcplg/idc_cgi_isapi-idcm1.dll.`

This variable is specific to the current gateway program request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the virtual path as a string.

Example

Retrieves the CGI path:

```
<$PATH_TRANSLATED$>
```

PLUGIN_DEBUG

Enables logging of events that occur inside the Web server plug-in filters.

When set to TRUE, events that occur inside any plugins that understand this flag will be recorded in the following log file:

-
- **Apache:** *IntradocDir*/data/users/authfilter.log

When changing this setting on Apache on Content Server, only the Content Server must be restarted.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21
- "Debugging" on page 4-9

Location

IntradocDir/data/users/SystemFilters.hda

Example

Used as a configuration entry:

```
PLUGIN_DEBUG=true
```

PrimaryWorkQueueTimeout

Sets the number of seconds until workflow and subscription notification e-mails are sent.

Set this to a lower number if you do not want e-mails to be delayed or if you want immediate response during debugging.

Default is 600 (10 minutes).

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Workflow" on page 4-21

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
PrimaryWorkQueueTimeout=30
```

Priority

Specifies the order in which LDAP providers will be checked for the user credentials.

This field is used only when a user has not previously logged into the Content Server. If the user has previously requested credentials, the SourcePath will be stored for that user, so the LDAP provider specified by the Source Path will be used.

Each LDAP provider in a Content Server instance must have a unique Priority number.

Default is 1.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, Priority
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
Priority=2
```

See Also

- "SourcePath" on page 6-248
- "IsPrimaryUserBase" on page 6-123

ProviderClass

Specifies the name of a Java class that implements the provider.

The default depends on the type of provider:

Provider Type	Default Value
Outgoing	intradoc.provider.SocketOutgoingProvider
Database	intradoc.jdbc.JdbcWorkspace
Incoming	intradoc.provider.SocketIncomingProvider
Preview	intradoc.provider.SocketOutgoingProvider
Ldapuser	intradoc.provider.LdapUserProvider

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19
- "Content Server" on page 4-4

Location

- Add/Edit Provider pages, Provider Class
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
ProviderConnection=intradoc.provider.SocketOutgoingProvider
```

ProviderConfig

Specifies the name of a Java class that performs some extra configuration for a provider.

This setting is most useful for database providers, where the connection classes are already providers.

There is no default.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19
- "Content Server" on page 4-4

Location

- Add/Edit Provider pages, Configuration Class
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
ProviderConfig=intradoc.provider.CustomClass
```

ProviderConnection

Specifies the name of a Java class that implements the provider connection.

The default depends on the type of provider:

Provider Type	Default Value
Outgoing	intradoc.provider.SocketOutgoingConnection
Database	intradoc.jdbc.JdbcConnection
Incoming	intradoc.provider.SocketIncomingConnection
Preview	intradoc.provider.SocketOutgoingConnection
Ldapuser	intradoc.provider.LdapConnection

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19
- "Content Server" on page 4-4

Location

- Add/Edit Provider pages, Connection Class
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
ProviderConnection=intradoc.provider.SocketOutgoingConnection
```

ProviderType

Specifies the type of provider.

The following provider types are available:

Provider Type	Value
Outgoing	outgoing
Database	database
Incoming	incoming
Preview	preview
Ldapuser	ldapuser

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19
- "Content Server" on page 4-4

Location

instance_dir/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
ProviderType=ldapuser
```

QUERY_STRING

Retrieves the string that follows the ? delimiter in the URL for a query.

This variable is specific to the current CGI request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21
- "Searching" on page 4-17

Output

Returns the query information as a string.

Example

As information output on a page or to a log:

```
QUERY_STRING=IdcService=GET_DOC_PAGE&Action=GetTemplatePage&Page=STD_QUERY_PAGE
```

As part of an Idoc Script statement or evaluation:

```
<$if QUERY_STRING$>
<!--statement-->
```

RedirectHostsFilter

Used to indicate a filter of allowed redirect hosts. The current host is added to this filter by default.

Use a pipe character | to separate multiple addresses. Make sure there are no spaces on either side of the pipe.

Use wildcards to indicate groups of addresses (* for zero or more characters, ? for a single character).

Default is the URL of the local host.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21
- "External Security" on page 4-19

Location

- System Properties, Server tab, IP Address Filter
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
RedirectHostsFilter=*.google.com|www.cnn.com
```

regexMatches

Searches a string for a specific pattern using a regular expression to do matching. Regular expression constructs can contain characters, character classes, and other classes and quantifiers. See `http://java.sun.com` for details about the Java API for Class Pattern.

This feature is only available with JVM 1.4 or later versions; this is the default version for Content Server version 7.0 and later.

Type and Usage

- "Global Functions" on page 3-11
- "Content Items" on page 4-3

Parameters

This function takes two parameters:

- The first parameter is the string to match.
- The second parameter is the expression.

Example

The following example returns FALSE, because the string does not match the expression:

```
<$regexMatches("abcdef","abc")$>
```

The following example returns TRUE because the wild cards are present. If standard wild cards such as the asterisk (*) were used instead of the dot-asterisk (.*) convention, the match would fail.

```
<$regexMatches("abcdef", ".*abc.*")$>
```

See Also

- "Operators" on page 2-14
- "regexReplaceAll" on page 6-202
- "regexReplaceFirst" on page 6-203
- "strEquals" on page 8-300
- "strIndexOf" on page 8-303

regexReplaceAll

Searches a string for a specific pattern using a regular expression to do matching and replacing. Regular expression constructs can contain characters, character classes, and other classes and quantifiers. See `http://java.sun.com` for details about the Java API for Class Pattern.

This feature is only available with JVM 1.4 or later versions; this is the default version for Content Server version 7.0 and later.

It replaces all instances of the regular expression with the replacement string.

Type and Usage

- "Global Functions" on page 3-11
- "Content Items" on page 4-3

Parameters

This function takes three parameters:

- The first parameter is the original string.
- The second parameter is the string to match.
- The third parameter is the replacement string.

Example

The following example returns `xyzdef xyzdef`:

```
<$regexReplaceAll("abcdef abcdef","abc","xyz")$>
```

See Also

- "Operators" on page 2-14
- "regexMatches" on page 6-201
- "regexReplaceFirst" on page 6-203
- "strEquals" on page 8-300
- "strIndexOf" on page 8-303
- "strSubstring" on page 8-313

regexReplaceFirst

Searches a string for a specific pattern using a regular expression to do matching and replaces the first instance with a replacement string. Regular expression constructs can contain characters, character classes, and other classes and quantifiers. See `http://java.sun.com` for details about the Java API for Class Pattern.

This feature is only available with JVM 1.4 or later versions; this is the default version for Content Server version 7.0 and later.

Type and Usage

- "Global Functions" on page 3-11
- "Content Items" on page 4-3

Parameters

This function takes three parameters:

- The first parameter is the original string.
- The second parameter is the string to match.
- The third parameter is the replacement string.

Example

The following example returns `xyzdef abcdef`:

```
<$regexReplaceFirst("abcdef abcdef","abc","xyz")$>
```

See Also

- "Operators" on page 2-14
- "regexMatches" on page 6-201
- "regexReplaceAll" on page 6-202
- "strEquals" on page 8-300
- "strIndexOf" on page 8-303
- "strSubstring" on page 8-313

REMOTE_ADDR

Returns the IP address of the remote host making the request.

This variable is specific to the current gateway program request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the IP address as a string.

Example

As information output on a page or to a log:

```
REMOTE_ADDR=207.0.0.1
```

As part of an Idoc Script statement or evaluation:

```
<$if REMOTE_ADDR$>
<!--statement-->
```

See Also

- "REMOTE_HOST" on page 6-205

REMOTE_HOST

Returns the name of the remote host making the request.

This variable is specific to the current gateway program request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

- Returns the host name as a string.
- If the hostname is unknown to the server, returns the value of REMOTE_ADDR as a string.

Example

As information output on a page or to a log:

```
REMOTE_HOST=207.0.0.1
```

As part of an Idoc Script statement or evaluation:

```
<$if REMOTE_HOST$>
<!--statement-->
```

See Also

- "REMOTE_ADDR" on page 6-204

REQUEST_METHOD

Retrieves the method that was used to make the request.

This variable is specific to the current gateway program request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the request method as a string.

Example

As information output on a page or to a log:

```
REQUEST_METHOD=GET
```

As part of an Idoc Script statement or evaluation:

```
<$if REQUEST_METHOD$>
<!--statement-->
```

ResultsTitle

Defines a title for the search results page.

This variable is used by the Web Layout Editor to name the search results page and display a heading at the top of that page.

Type and Usage

- "Value Variables" on page 3-21
- "Page Display" on page 4-14
- "Searching" on page 4-17

Output

Returns the page title as a string.

Example

As an HDA entry, names the search results page:

```
@Properties LocalData
ResultsTitle=Content Items
@end
```

As a script, returns the defined name:

```
<$if ResultsTitle$>ResultsTitle=<$url(ResultsTitle)$>
```

SchemaAllowRecursiveFileMove

If set to TRUE, enables the ability to move files on UNIX systems on an individual basis from a schema publishing location into the production location. This allows swapping of the schema directory into its production directory.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaAllowRecursiveFileMove=true
```

SchemaBuildDigestFile

If set to TRUE, builds a file containing digests of all the files published. Setting this to TRUE increases the time required to publish schema.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaBuildDigestFile=true
```

See Also

- "SchemaPublishCheckInterval" on page 6-210

SchemaPublishCheckInterval

Establishes the interval between checks for non-subject-driven schema publishing requests.

This is triggered if the data/schema/publishlock/publish.dat file is accessed.

Specify the value in seconds.

Default is 120 seconds (two minutes).

Change this setting if there is an external program that modifies data in schema-managed tables and which accesses the file *instance_dir*/data/schema/publishlock/publish.dat.

On a development server, use a low value to ensure prompt publishing (for example, `SchemaPublishCheckInterval=2`).

Lowering the default value on a production server may cause performance problems.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

To set this interval for 10 minutes, use the following:

```
SchemaPublishCheckInterval=600
```

SchemaPublishInterval

The interval between unprovoked schema operations (that is, operations that are not caused by an event).

Specify this value in seconds.

Default is 14400 (four hours).

SchemaPublisher will usually publish at least once during this interval.

On a development server, use a low value to ensure prompt publishing (for example, `SchemaPublishInterval=30`).

Lowering the default value on a production server may cause performance problems.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

To set this interval to one week, use the following:

```
SchemaPublishInterval=604800
```

SchemaPublishMaximumInterval

The maximum amount of time between automatic schema publishing cycles.

Specify this value in seconds.

Default is 86400 (one day).

On a development server, set a low value to ensure prompt publishing (for example, `SchemaPublishMaximumInterval=30`).

Lowering the default value on a production server may cause performance problems.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

To specify this value as four hours, use the following:

```
SchemaPublishMaximumInterval=14400
```

SchemaPublishMinimumInterval

The minimum amount of time between automatic schema publishing cycles.

Specify this value in seconds.

Default is four seconds.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaPublishMinimumInterval=8
```

SchemaPublishRenameRetryInitialInterval

Specifies the delay before retrying a rename operation.

The rename operation involves renaming of the *schema* directory to *schema.old* or the *schema.work* directory to *schema*.

After a failed retry, the delay value is doubled.

If `SchemaPublishRenameRetryIterations` is reached, the publishing operation fails.

Specify the value in milliseconds.

The default value is 100 milliseconds.

This affects behavior on Windows systems only.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaPublishRenameRetryInitialInterval=50
```

See Also

- "SchemaPublishRenameRetryIterations" on page 6-215

SchemaPublishRenameRetryIterations

Specifies the number of times a rename operation will be tried.

This affects operations on Windows systems only.

The operation involves renaming of the *schema* directory to *schema.old* or the *schema.work* directory to *schema*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaPublishRenameRetryIterations=6
```

See Also

- "SchemaPublishRenameRetryInitialInterval" on page 6-214

SchemaPublishRetryInterval

Specifies the delay introduced after SchemaPublisher fails to publish the schema.

This delay prevents the SchemaPublisher from continually starting and failing.

After a failed retry, this value is doubled but is never larger than the value of `SchemaPublishRetryIntervalMax`.

When publishing succeeds, the value is reset to the original value

Default value is three seconds.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaPublishRetryInterval=4
```

See Also

- "SchemaPublishRetryIntervalMax" on page 6-217

SchemaPublishRetryIntervalMax

Specifies the longest length of time for `SchemaPublishRetryInterval`.

Specify this value in seconds.

Default value is 300 (five minutes).

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaPublishRetryIntervalMax=120
```

See Also

- "SchemaPublishRetryInterval" on page 6-216

SchemaVerifyPublishing

If set to TRUE, digests will be used to verify the files that SchemaPublisher creates before doing the final renaming.

Setting `SchemaVerifyPublishing`=TRUE will increase the time required to publish schema.

This should be set to TRUE on a production server.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaVerifyPublishing=true
```

See Also

- "SchemaBuildDigestFile" on page 6-209
- "SchemaPublishCheckInterval" on page 6-210

SchemaWin32RenameDelayInterval

Specifies the size of the delay prior to closing a file before attempting to rename it.

This affects behavior only on Windows systems.

Tune this number as low as possible.

Value should be specified in milliseconds.

The default value is 10 milliseconds.

Type and Usage

- "Configuration Variables" on page 3-2
- "Schema" on page 4-16

Location

IntradocDir/config/config.cfg

Example

```
SchemaWin32RenameDelayInterval=4
```

SCRIPT_NAME

Retrieves the relative path to the CGI linking file.

The CGI linking file is the executable file for the Web server security filter, which is a different file for each operating system and Web server. For example, the file name is *nph-idc_cgi.exe* file for Solaris.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the relative path as a string.

Example

As information output on a page or to a log:

```
SCRIPT_NAME=/idcplg/idc_cgi_isapi-instance>.dll
```

As part of an Idoc Script statement or evaluation:

```
<$if SCRIPT_NAME$>
<!--statement-->
```

SearchCacheCleanerInterval

Sets the time between search cache cleanup attempts in seconds.

Default is 120 (2 minutes).

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
SearchCacheCleanerInterval=60
```

SearchCacheHardLimit

Defines the maximum number of results that will be included in the search cache when a search request includes the hard limit.

If the end row is larger than `SearchCacheSoftLimit` but less than or equal to `SearchCacheHardLimit`, the search cache contains all rows from 1 to `SearchCacheHardLimit`.

If the end row is larger than `SearchCacheHardLimit`, nothing is cached for the search.

Typically, this setting should match the maximum number of search results displayed per page. See MaxResults.

Default is 200.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
SearchCacheHardLimit=300
```

See Also

- "MaxResults" on page 6-170
- "SearchCacheSoftLimit" on page 6-224

SearchCacheMaxItemAgeInMinutes

Sets the maximum age of an item in the search cache in minutes.

Default is 60 (1 hour).

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
SearchCacheMaxItemAgeInMinutes=30
```

SearchCacheSoftLimit

Defines the maximum number of results that will be included in the search cache when a search request includes the soft limit.

- If the end row of the search is less than or equal to `SearchCacheSoftLimit`, the search cache contains the rows from 1 to `SearchCacheSoftLimit`.
- If the end row is larger than `SearchCacheSoftLimit` but less than or equal to the `SearchCacheHardLimit`, the search cache contains all rows from 1 to `SearchCacheHardLimit`.
- If the end row is larger than `SearchCacheHardLimit`, nothing is cached for the search.

Typically, this setting should match the default number of search results displayed per page.

Default is 25.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
SearchCacheSoftLimit=50
```

See Also

- "SearchCacheHardLimit" on page 6-222

SearchCacheTrace

Obsolete as of the 7.0 release. The functionality is now available in the System Audit Information page.

Dumps trace information to the console for the search cache cleaning algorithm.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
SearchCacheTrace=true
```

SearchConnectionWaitTimeout

Sets the search connection wait timeout in milliseconds.

This is the maximum amount of time that a search request waits in the queue before the search is refused.

Default is 60000 (1 minute).

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
SearchConnectionWaitTimeout=120000
```

SearchDebugLevel

Defines the level of detail for entries in the Indexer log files.

The log files are located at */weblayout/groups/secure/logs/*.

The following list shows the debug levels:

Level	Description
none	No log is generated. This is the default.
verbose	Displays information for each file accessed. Indicates indexed, ignored, or failed, and generates a full report.
debug	Additional information at the functional level is generated.
trace	Information is logged as each activity is performed.
all	A full report including debug, trace, and verbose information is generated.

Tip: If you set the debug level to anything but *none*, a log file will be generated for each index cycle. You may need to perform system maintenance periodically to reduce the number of log files.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17
- "Debugging" on page 4-9

Location

- Repository Manager, Indexer tab, Configure, Indexer Debug Level
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
SearchDebugLevel=all
```

SearchDir

Defines the path to the *search* directory.

The default value is *DomainHome*/ucm/cs/search

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9
- "Searching" on page 4-17

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
SearchDir=f:/search/
```

SearchIndexerEngineName

This variable specifies the name of the search engine to use. Supported values include:

- DATABASE
- DATABASEFULLTEXT

Each search engine uses its own format for query strings. The default uses universal query syntax, but database search requires SQL syntax.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

```
SearchIndexerEngineName=DATABASE
```

SearchQueryFormat

Retrieves search query pages with query syntax that is recognized regardless of supported search engines.

If used dynamically, appends the following to the URL:

```
SearchQueryFormat=Universal
```

When used dynamically, the universal search syntax is maintained through one search session. The syntax is maintained on a query page, result page and its navigation link and during actions such as searching within the page or saving the query. It is maintained until the user clicks the menu selection to re-load a search query page.

This can also be added to the config.cfg file so that the universal search query page is always used.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
SearchQueryFormat=Universal
```

See Also

- "UniversalSearchAllowFullText" on page 6-257

SearchQueryTrace

This variable is obsolete.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Searching" on page 4-17
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
SearchQueryTrace=true
```

SelfRegisteredAccounts

Defines the default accounts and permissions to be given to self-registered users.

This is a comma-delimited list.

Permissions for each account must be specified in parentheses after the account name.

The `#none` entry grants privileges to content items that have no account assigned. The `#all` entry grants privileges to all accounts.

There is no default value.

Returns the list of accounts as a string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
SelfRegisteredAccounts=#none(RWDA),USERS/<$NewUser$>,BOS(R)
```

As script, returns the defined account information as a string:

```
<$SelfRegisteredAccounts$>
```

See Also

- "SelfRegisteredAccounts" on page 6-232

SelfRegisteredRoles

Defines the default roles to be given to self-registered users.

This is a comma-delimited list.

There is no default value.

Returns the roles as a string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
SelfRegisteredRoles=guest,salesRole
```

As script, returns the defined roles as a string:

```
<$SelfRegisteredRoles$>
```

See Also

- "SelfRegisteredRoles" on page 6-233

SERVER_NAME

Retrieves the hostname, DNS alias, or IP address of the Web server as it would appear in a self-referencing URL.

This variable is not request-specific; it is set for all requests.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the server information as a string.

Example

As information output on a page or to a log:

```
SERVER_NAME=centralserver
```

As part of an Idoc Script statement or evaluation:

```
<$if SERVER_NAME$>
<!--statement-->
```

SERVER_PORT

Retrieves the Web server port number to which the request was sent.

This variable is specific to the current gateway program request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns the port number as a string.

Example

As information output on a page or to a log:

```
SERVER_PORT=80
```

As part of an Idoc Script statement or evaluation:

```
<$if SERVER_PORT$>
<!--statement-->
```

SERVER_PROTOCOL

Returns the protocol and revision of the incoming request.

This variable is specific to the current gateway program request.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns a string in the format `protocol/revision`.

Example

As information output to a log:

```
SERVER_PROTOCOL=HTTP/1.0
```

As part of an Idoc Script statement or evaluation:

```
<$if SERVER_PROTOCOL$>
<!--statement-->
```

SERVER_SOFTWARE

This setting is obsolete for the 7.0 version and later. The Web server filter no longer sends this information.

Returns the name and version of the Web server software that is answering the request.

This is also the server running the gateway.

This variable is not request-specific; it is set for all requests.

Type and Usage

- "Environment Variables" on page 3-11
- "Web Server" on page 4-21

Output

Returns a string in the format `name/version`.

Example

As information output on a page or to a log:

```
SERVER_SOFTWARE=Microsoft-IIS/4.0
```

As part of an Idoc Script statement or evaluation:

```
<$if SERVER_SOFTWARE$>
<!--statement-->
```

ServiceStartupTimeout

Sets the Content Server startup timeout value. This configuration setting applies only to Content Servers running on a Microsoft Windows platform.

Used to modify the server startup timeout value when a service requires more time to start up than is provided by the default Content Server startup timeout value, which is 120 seconds (2 minutes).

The value must be specified in seconds.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

The following example sets the startup timeout value to 240 seconds (4 minutes):

```
ServiceStartupTimeout=240
```

setCookie

Sets a cookie value in a browser.

Used to retain session information or information about anonymous users.

An expiration date can also be passed to make the cookie expire at a specified time.

Important: Do not store secure data in a cookie. You should limit the use of cookies due to the risk of malicious users setting cookies on your site.

Type and Usage

- "Setable Variables" on page 3-19
- "Web Server" on page 4-21

Parameters

This function can take three parameters:

- The first parameter is the name of the cookie.
- The second parameter is the value to be set.
- The third optional parameter is an expiration indicator specifying when the cookie will expire.

Example

The following example sets the cookie `myCounter` in the user's browser and increments the value when the user visits the page. The cookie will expire in one day.

```
<$myCounter=getCookie("myCounter")$>
<$if not myCounter$>
    <$myCounter=1$>
<$endif$>
Num times loaded: <$myCounter$>
<$setCookie("myCounter", myCounter+1, dateCurrent(1))$>
```

See Also

- "getCookie" on page 6-60

ShowCreateExternalMenu

This Content Server configuration variable is specific to URM. Set this variable to true to display the following create external source features in your interface:

- **Create** <external_source_name> **Item** option in the External Content, <external_source_name> submenu. Note that the External Content menu is displayed only if an external source has been created.
- **Create** <external_source_name> **Record** option (record categories) or Check in New <external_source_name> Item option (non-record categories) in the Action menu on the Exploring Retention Category page.
- **Create** <external_source_name> **Record** option (record categories only) in the Action menu on the Exploring Records Folder page.
- **Link New** <external_source_name> **Item** option in the Action menu on any link page.

When set to false, these features will not be displayed.

The default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager Configuration Variables" on page 3-10

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ShowCreateExternalMenu=true
```

ShowOnlyKnownAccounts

Defines whether to display only predefined accounts on checkin and search pages.

- When set to TRUE, only predefined accounts appear in the Accounts option list on checkin and search pages.
- When set to FALSE, any user-defined accounts that the user belongs to are also displayed in the option list.

Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18
- "Field Display" on page 4-11

Location

- System Properties, Content Security tab, Show only known accounts
- Admin Server, Content Security, Show only known accounts
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
ShowOnlyKnownAccounts=true
```

SingleGroup

Evaluates if the current revision is in a contributor step of a Basic workflow.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Content Items" on page 4-3
- "Workflow" on page 4-21

Output

- Returns TRUE if the revision is in a contributor step.
- Returns FALSE if the revision is not in a contributor step.

Example

Evaluates if revision is in a contributor step:

```
<$if not SingleGroup$>
```

SkipMozillaMailToWorkaround

Used to disable an email solution that allows Mozilla mail applications to accurately handle email from the Search Result and Content Info page. If your mailer does not send correctly formatted email, set this variable to `TRUE`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

- *IntradocDir*/config/config.cfg

Example

Disables special email handling:

```
SkipMozillaMailToWorkaround=true
```

SmtpPort

Defines the Simple Mail Transfer Protocol (SMTP) port number.

Default is 25.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

- System Properties, Internet tab, SMTP Port
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
SmtpPort=25
```

See Also

- "SysAdminAddress" on page 8-318
- "MailServer" on page 6-162

SocketHostAddressSecurityFilter

Defines the IP addresses that are allowed to make calls to the Content Server.

Either this setting or SocketHostNameSecurityFilter is required to allow remote IdcCommandX calls to the Content Server. (Generally, both settings should not be used at the same time.) See the *IdcCommand-Java Command Utility Reference Guide* for more information.

Use a pipe character | to separate multiple addresses. Make sure that there are no spaces on either side of the pipe.

Use wildcards to indicate groups of addresses (* for zero or more characters, ? for a single character).

Default is the IP address of the local host.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21
- "External Security" on page 4-19

Location

- System Properties, Server tab, IP Address Filter
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
SocketHostAddressSecurityFilter=10.10.3.*|127.0.0.1
```

See Also

- "SocketHostNameSecurityFilter" on page 6-246

SocketHostNameSecurityFilter

Defines the server names that are allowed to make calls to the Content Server.

Either this setting or SocketHostAddressSecurityFilter is required to allow remote IdcCommandX calls to the Content Server. (Generally, both settings should not be used at the same time.) See the *Idc Command Reference Guide* for more information.

Use a pipe character `|` to separate multiple server names. Make sure that there are no spaces on either side of the pipe.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21
- "External Security" on page 4-19

Location

- System Properties, Server tab, Hostname Filter
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
SocketHostNameSecurityFilter=olympus|localhost|zeus
```

See Also

- "SocketHostAddressSecurityFilter" on page 6-245

SourceID

Provides the Content ID for the current dynamic server page.

This variable returns the same value as `ref:dID`. See Referencing Metadata in Dynamic Server Pages.

Type and Usage

- "Content Read-Only Variables" on page 3-19
- "Content Items" on page 4-3

Output

Returns the Content ID as a string.

Example

Sets the *dID* variable to the value of the current HCSF page:

```
<input type=hidden name="dID" value="<$SourceID$>">
```

SourcePath

Specifies a unique string that identifies an LDAP provider. The first time a user requests credentials through the provider, this string is stored with the user information so it can be used to match the user with the provider next time the user asks for credentials.

We suggest using the name of the LDAP provider as the Source Path.

There is no default.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, Source Path
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
SourcePath=LdapProvider1
```

See Also

- "Priority" on page 6-194

StatusCode

Indicates if the last function call was successful or failed. This variable is set at the end of a service call.

Certain functions set this variable to zero (0) upon success.

Certain functions set this variable to a negative numeric value (-1) upon failure. Typically, the abortToErrorPage function is called to display an error page with the StatusMessage from the most recent function call.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4

Output

- Returns 0 if the last function call was successful.
- Returns -1 if the last function call failed.

Example

Returns the current status code:

```
<$StatusCode$>
```

See Also

- "abortToErrorPage" on page 8-2
- "getUserValue" on page 8-115
- "hasAppRights" on page 6-67
- "StatusMessage" on page 6-250

StatusMessage

Defines the error message for the last function call. This variable is set at the end of a service call, but it can be set during a service call.

This variable is typically displayed on an error page when the abortToErrorPage function is called.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4

Output

- Returns OK if the last function call was successful.
- Returns a localized error message if the last function call failed.

Example

Returns the current status message:

```
<$StatusMessage$>
```

See Also

- "abortToErrorPage" on page 8-2
- "StatusCode" on page 6-249

stdSecurityCheck

Checks if the current user has access to the requested content item based on the standard security model.

The active data is checked to determine if the standard (or default) security model allows the user to have access to the content item. This enables a custom implementation of security to still execute the standard security model as a baseline.

Type and Usage

- "Global Functions" on page 3-11
- "Internal Security" on page 4-18

Parameters

None.

Output

- Returns TRUE if the current user has access to the content item.
- Returns FALSE if the current user does not have access to the content item.

Example

Compares the permission level of the user to the requested content item:

```
<$stdSecurityCheck()$>
```

SubscriptionCollatedFieldList

Defines a list of fields displayed for subscriptions.

The default fields that are defined at startup are dID and dDocName.

Additional fields can be defined using `AdditionalSubscriptionCollatedFieldList`.

It is recommended that you use `AdditionalSubscriptionCollatedFieldList` to add subscription fields to the collated list rather than change `SubscriptionCollatedFieldList.`

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

See Also

- "AdditionalSubscriptionCollatedFieldList" on page 6-4

SubscriptionMethod

Specifies how subscriptions are processed.

The possible values are:

- **SubscriptionOriented:** The Content Server compares each subscription with the content items that have been indexed.

  This value should be used when the number of subscriptions is small relative to the number of content items (for example, 100,000 content items and 1000 subscriptions).

- **DocumentOriented:** The Content Server compares each indexed content item with the active subscriptions.

  This value should be used when the number of subscriptions is large relative to the number of content items (for example, 1000 content items and 1000 subscriptions).

Default is *SubscriptionOriented*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
SubscriptionMethod=DocumentOriented
```

TextIndexerFilterValidateFormats

This variable specifies a comma-delimited list of file types that are verified for unreadable characters. When this variable is used, specified file formats are checked for unreadable characters and those characters are corrected before the text file is passed to be indexed.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

instance_dir/custom/TextIndexerFilter/textindexerfilter_einviornment.cfg

Example

```
TextIndexerFilterValidateFormats=application/pdf
```

TimeoutPerOneMegInSec

Sets the timeout for indexing files into the index collection.

- The size of a bulkload file being indexed is rounded up to the nearest megabyte, and then multiplied by this setting to determine the timeout value in seconds.
- If files have a large amount of numeric data, the timeout should be increased to 3000–6000. Indexing cleanup cycles, which occur after every four or so bulkloads and count toward the timeout, can take longer for numeric data.

Default is 600 seconds per megabyte.

Type and Usage

- "Configuration Variables" on page 3-2
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
TimeoutPerOneMegInSec=3000
```

UieHideSearchCheckboxes

Records Manager only.

Controls if the metadata field check boxes on the search page are shown.

If set to TRUE, the metadata field check boxes are not shown on the page. If set to FALSE, the metadata field check boxes are shown.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/custom/RecordsManagement/records_management_environments.cfg

instance_dir/custom/RetentionManager/retentionmanager_environments.cfg

Example

Used as a configuration entry:

```
UieHideSearchCheckboxes=false
```

UniversalSearchAllowFullText

Changes the displayed fields when `SearchQueryFormat=Universal` is used. If `UniversalSearchAllowFullText` is set to FALSE, the Full Text Search field is not displayed on the Search page.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

```
UniversalSearchAllowFullText=false
```

See Also

- "SearchQueryFormat" on page 6-230

UseAccounts

Enables the use of accounts.

When set to TRUE, accounts are enabled.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UseAccounts=true
```

UseAdsi

Enables Active Directory (ADSI) authentication for the Content Server.

This setting is valid only when NtlmSecurityEnabled is TRUE.

This option is available only when using an IIS Web server.

When set to TRUE, users defined in Active Directory can log in to the Content Server using their network credentials.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- System Properties, Internet tab, Active Directory Security (ADSI) option
- Admin Server, Internet Configuration, Active Directory Security option
- *IntradocDir*/config/config.cfg

Example

Enables ADSI security:

```
UseAdsi=true
```

See Also

- "NtlmSecurityEnabled" on page 6-187
- "UseNtlm" on page 6-263

UseDatabaseShortIndexName

Required if the database in use does not support index names longer than eighteen characters.

When set to TRUE, short index names are used.

When set to FALSE, short index names are not used.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UseDatabaseShortIndexName=false
```

UseHtmlOrTextHighlightInfo

Checks whether full-text search keyword highlighting is enabled and the file format supports highlighting (such as PDF, HTM, or TXT).

If the EnableDocumentHighlight setting is TRUE, keyword highlighting is enabled.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Searching" on page 4-17

Output

- Returns TRUE if highlighting is enabled.
- Returns FALSE if highlighting is not enabled.

Example

Returns search keyword highlighting status:

```
<$UseHtmlOrTextHighlightInfo$>
```

See Also

- "UseXmlUrl" on page 6-273
- "EnableDocumentHighlight" on page 6-45

UseNetscape

Specifies the method used to connect to an LDAP server through a provider.

- When set to TRUE, the Netscape SDK will be used. This is the recommended setting for better performance.
- When set to FALSE, the JNDI connection will be used.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- Add/Edit LDAP Provider page, Use Netscape SDK
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
UseNetscape=0
```

UseNtlm

Enables NT LAN Manager (NTLM) authentication for the Content Server.

- This setting is valid only when NtlmSecurityEnabled is TRUE.
- This option is available only when using an IIS Web server.
- When set to TRUE, users defined on a Windows network (NTLM) can log in to the Content Server using their network credentials.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19

Location

- System Properties, Internet tab, NT Domain Security (NTLM) option
- Admin Server, Internet Configuration, NT Domain Security option
- *IntradocDir*/config/config.cfg

Example

Enables NTLM security:

```
UseNtlm=true
```

See Also

- "NtlmSecurityEnabled" on page 6-187
- "UseAdsi" on page 6-259

UseRevisionCreateDate

When this variable is set in the system importing an archive, the Archiver uses the dCreateDate for comparisons.

This allows the Content Server to differentiate revisions by their initial create date and not by the assigned dInDate.

This is not set by default.

Important: Use this configuration setting if no legacy content from older versions (pre 7.5) of Content Server are being imported.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

```
UseRevisionCreateDate=true
```

UserFilteredSubjects

Specifies the subjects that must be refreshed if the current user's security credentials have changed.

- Subjects are subsystems that are notified when a Content Server service makes a change in the system, such as modifying an option list or adding a new user.
- Add any custom security subjects to this setting. Custom subjects are typically created only for unusually complex security integrations.
- This is a comma-delimited list.

Default is *users,accounts*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UserFilteredSubjects=users,accounts,licenses
```

userHasAccessToAccount

This function returns TRUE if the user has access to a named account at a specific privilege level. The privilege is a one-character representation of the access level, as follows:

- R: Read. Specified as 1.
- W: Write. Specified as 2.
- D: Delete. Specified as 4.
- A: Administration. Specified as 8.

Type and Usage

- "Global Functions" on page 3-11
- "Internal Security" on page 4-18
- "Users" on page 4-20

Parameters

Takes two parameters:

- The first is the account name.
- The second is the access level to be checked.

Output

- Returns TRUE if the current user has the specified access.
- Returns FALSE if the user does not have the specified access.

Example

Evaluates whether the user has Read access to the specified account:

```
<$userHasAccessToAccount("profile_account", 1)$>
```

Evaluates whether the user has Read and Write access to the specified account:

```
<$userHasAccessToAccount("profile_account", 3)$>
```

userHasGroupPrivilege

This function returns TRUE if the user has the specified privilege to the specified group. The privilege is a one-character representation of the access level, as follows:

- R: Read
- W: Write
- D: Delete
- A: Administration

Type and Usage

- "Global Functions" on page 3-11
- "Internal Security" on page 4-18
- "Users" on page 4-20

Parameters

Takes two parameters:

- The security group to be checked.
- The access level to be checked.

Output

- Returns TRUE if the current user does have the specified access.
- Returns FALSE if the user does not have the specified access.

Example

Evaluates whether the user has the specified role:

```
<$userHasGroupPrivilege("Public", "R")$>
<$userHasGroupPrivilege("Secure", "A")$>
```

userHasRole

Checks if the current user has a particular role.

Type and Usage

- "Global Functions" on page 3-11
- "Internal Security" on page 4-18
- "Users" on page 4-20

Parameters

The only parameter is the name of a role.

Output

- Returns TRUE if the current user does not have the specified role.
- Returns FALSE if the user does not have the specified role.

Example

Evaluates whether the user has the specified role:

```
<$userHasRole("admin")$>
```

UserIsAdmin

Checks if the current user has rights to any administration applets.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

- Returns TRUE if the user has any administrative rights.
- Returns FALSE if the user does not have any administrative rights.

Example

Evaluates administrative rights of user:

```
<$UserIsAdmin$>
```

See Also

- "AdminAtLeastOneGroup" on page 6-5
- "UserAppRights" on page 8-341

UseSearchCache

When set to TRUE, enables the search cache.

The search cache improves search performance by caching recent search engine queries.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

As a configuration variable:

```
UseSearchCache=true
```

If this variable is set to FALSE and used on the CGI line of a browser URL request doing a query, it forces the query to execute each time. This makes it possible to measure the speed of the query being executed and to verify the cache results (because a new search result is generated each time).

```
http://...?IdcService=GET_SEARCH_RESULTS&...UseSearchCache=false
```

UseSecureLdap

Specifies whether SSL is used with the LDAP provider.

- When set to TRUE, SSL is used. You must have the appropriate certificates installed on the LDAP server. Once SSL is initiated, the certificates will secure communication between the LDAP server and the Content Server.
- When set to FALSE, SSL is not used.

Default is FALSE.

Type and Usage

- Section 3.2, "Configuration Variables"
- Section 4.27.2, "External Security"

Location

- Add/Edit LDAP Provider page, Use SSL
- *instance_dir*/data/providers/*provider_name*/provider.hda

Example

Used as a configuration entry:

```
UseSecureLdap=true
```

See Also

- "LdapPort" on page 6-144

UseSSL

Enables the secure sockets layer (SSL).

- This setting affects the variables HttpWebRoot andHttpCgiPath.
- Use the Secure Sockets Layer only if you are using an SSL-enabled Web server.
- When set to TRUE, SSL is used (*https* instead of *http*).

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "External Security" on page 4-19
- "Web Server" on page 4-21

Location

- System Properties, Internet tab, Use Secure Sockets Layer
- *IntradocDir*/config/config.cfg

Example

As a configuration setting, enables SSL:

```
UseSSL=true
```

As script, returns the value of the configuration setting:

```
<$UseSSL$>
```

See Also

- "HttpCgiPath" on page 8-130
- "HttpWebRoot" on page 8-141

UseXmlUrl

Checks if search keyword highlighting for XML documents is enabled. When set to TRUE, an XML URL is constructed when performing a keyword search.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Searching" on page 4-17

Output

- Returns TRUE if XML keyword highlighting is enabled.
- Returns FALSE if XML keyword highlighting is not enabled.

Example

Returns the status of search keyword highlighting for XML documents.

```
<$UseXmlUrl$>
```

See Also

- "UseHtmlOrTextHighlightInfo" on page 6-261
- "EnableDocumentHighlight" on page 6-45

WebServerAuthOnly

Specifies whether the Web server filter will authenticate users that are defined in the Content Server database.

- When set to TRUE, the Web server filter will not validate users against the internal user names and passwords from the Content Server database.
- If Basic Authentication is enabled in IIS, then this setting must be TRUE. This setting enables browsers to use Basic authentication when logging into the Microsoft Network; users can log in using a Windows user name and password using the Netscape browser.
- If this option is enabled, new users can be added only by adding them to a Windows network domain, since users are not authenticated against the Content Server user database. However, if the database happens to have a user of the same name as a Windows network user, then the Content Server security profile is used instead of the Windows network profile.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Web Server" on page 4-21
- "External Security" on page 4-19

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry to enable Web server authorization only:

```
WebServerAuthOnly=true
```

wfAction

Retrieves the action that is currently being performed on the revision.

- This variable is set after the Exit event of a workflow step, so it is normally evaluated during the Entry event of the next step.
- The possible actions are:
  - APPROVE
  - REJECT
  - CHECKIN
  - CONVERSION
  - META_UPDATE
  - TIMED_UPDATE
  - RESUBMIT

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the name of the action as a string.

Example

If the revision is in the process of being rejected, notify the original author:

```
<$if wfAction like "REJECT"$>
    <$wfSet("wfJumpName", "notifyAuthor")$>
    <$wfSet("wfJumpEntryNotifyOff", "0")$>
    <$wfNotify(dDocAuthor, "user")$>
<$endif$>
```

If the revision is currently being approved but the metadata value of *xDept* is not HR, terminate the approval and display an error page:

```
<$if wfAction like "APPROVE" and not(xDept like "HR")$>
    <$abortToErrorPage("The revision is not in HR.")$>
<$endif$>
```

wfAddActionHistoryEvent

Inserts an event into the WorkflowActionHistory table, in the workflow's companion data. The three parameters to this function are required. The rest of the row's values are computed or inherited from local data.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes three parameters:

- The first parameter is the workflow action.
- The second parameter is the message associated with the action.
- The third parameter is a comma-delimited list of users for the current workflow step,. See `wfComputeStepUserList` for a function which computes the current list.

See Also

- "wfComputeStepUserList" on page 6-279

wfAdditionalExitCondition

Retrieves the exit condition that has been defined for the current step.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the exit condition as a string.

Example

The following code in the *IdcHomeDir*/resources/core/templates/workflow_info.htm template page is used to display the exit condition on the Workflow Step Information page:

```
<$wfDisplayCondition(dWfName, currentStepName, "wfAdditionalExitCondition")$>
```

Typical exit condition output for this variable would look like:

```
dSecurityGroup like "Secure"
```

wfAddUser

Adds a user, alias, or workflow token to the list of reviewers for a workflow step. This function can only be used inside a token.

Type and Usage

- "Workflow Functions" on page 3-21
- "Users" on page 4-20

Parameters

Takes two parameters:

- The first parameter indicates the user name, alias, or token name.
- The second parameter indicates the type, either *user* or *alias*.

Example

Adds the user `mjones` as a reviewer:

```
<$wfAddUser("mjones", "user")$>
```

Adds the original author token and the `hr` alias as reviewers:

```
<$wfAddUser(dDocAuthor, "user")$>
<$wfAddUser("hr", "alias")$>
```

wfComputeStepUserList

Computes the list of users from the current step in the workflow.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Output

Returns a comma-delimited list of users.

See Also

- "wfAddActionHistoryEvent" on page 6-276

wfCurrentGet

Retrieves a local state value from the companion file.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

The only parameter is the key.

Output

Returns the local state value from the companion file.

Example

Returns the value of the local key `<step_name>@<workflow_name>.myKey`:

```
<$wfCurrentGet("myKey")$>
```

See Also

- "wfGet" on page 6-285

wfCurrentSet

Sets the local state value of a key in the companion file.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes two parameters:

- The first parameter is the key.
- The second parameter is the value.

Example

Sets the key `<step_name>@<workflow_name>.myKey` to `myValue`:

```
<$wfCurrentSet("myKey", "myValue")$>
```

See Also

- "wfSet" on page 6-301

wfCurrentStep

Retrieves the name of a step relative to the current step.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

The only parameter is an integer that represents the number of steps relative to the current step.

Output

- Returns a step name.
- Reaching the beginning or the end of the workflow returns the name of the first or last step, respectively.

Example

Returns the current step name:

```
<$wfCurrentStep(0)$>
```

Returns the previous step name:

```
<$wfCurrentStep(-1)$>
```

Returns the next step name:

```
<$wfCurrentStep(1)$>
```

To actually move to the next step you must set up a jump, as in this example:

```
<$wfSet("wfJumpName", "move to next step")$>
<$wfSet("wfJumpTargetStep", wfCurrentStep(1))$>
<$wfSet("wfJumpEntryNotifyOff", "0")$>
```

wfDisplayCondition

Retrieves the exit condition for a workflow step.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes three parameters:

- The first parameter is the workflow name.
- The second parameter is the step name.
- The third parameter is the exit condition to be displayed.

Output

Returns the exit condition expression.

Example

The following code in the *IdcHomeDir*/resources/core/templates/workflow_info.htm template page is used to display the exit condition on the Workflow Step Information page:

```
<$wfDisplayCondition(dWfName, currentStepName, "wfAdditionalExitCondition")$>
```

where:

- dWfName is the internal variable for the workflow name.
- *currentStepName* is set earlier in the template page to be equal to dWfStepName, which is the internal variable for the step name.
- wfAdditionalExitCondition is the internal variable for the exit condition expression. Typical exit condition output would look like:

  ```
  dSecurityGroup like "Secure"
  ```

wfExit

Exits a workflow step. This function moves the revision to a particular step in a workflow according to the function parameters and resets the parent list information. To completely exit a workflow, use `wfExit(100,100)` or any parameters that ensure that the revision returns to the parent workflow and then gets moved past the last step in that workflow.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes two parameters:

- The first parameter indicates the number of jumps to rewind.
- The second parameter indicates the target step relative to the step determined by the first parameter.

Example

Exits to the parent step in the workflow:

```
<$wfExit(0,0)$>
```

Returns to the previous jump step:

```
<$wfExit(1,0)$>
```

Returns to the previous jump step and moves to the next step in that workflow:

```
<$wfExit(1,1)$>
```

wfGet

Retrieves a state value from the companion file.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

The only parameter is the state key.

Output

Returns the state value from the companion file.

Example

Returns the current jump name:

```
<$wfGet("wfJumpName")$>
```

See Also

- "wfCurrentGet" on page 6-280

wfGetStepTypeLabel

This function takes an internal workflow step type value and turns it into a human readable label. For example, :C:CA:CE: is the three states :C: (contribution), :CA: (autocontribute or original author step) and :CE: (Edit revision). Applying the Idoc script function the label might become "Auto Contributor, Edit Revision".

Here are the possible current states.

States	Description
:CA:	Auto contribution step or step that occurs before all administrator defined workflow steps ("original author handles document rejection").
:CE:	Edit current revision. All edits replace current revision.
:CN:	Create new revision. All edit create new revision.
:R:	Review. Document can be approved or rejected.
:C:	Contribution. The document can be edited.

A workflow document step state has all the states that are true for it combined as a single fields with multiple values separated by colons (:) (redundant colons are eliminated). So a contributor/reviewer step that creates new revisions would have the state :R:C:CE:. Order does not matter. But :C: must be present even though it can be presumed by the presence of :CE: (:CE: also implies that the step is a contributor step). The value :C: accurately describes one capability of the current workflow step and the Content Server does not chase implication rules (:CE: -> :C:) so the Content Server will see the absence of :C: (even with :CE: present) as a statement that the workflow step does not allow core contributor types of activities (such as checkout or undo checkout).

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Parameters

The only parameter is the internal workflow step type value.

Example

```
<$wfGetStepTypeLabel(stepType)$>
```

wfIsFinishedDocConversion

Indicates if the document will not be in GENWWW after the current action finishes.

If this Idoc Script function returns true, the conversion process has finished and the content item is no longer in GENWWW.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Output

- Returns TRUE if the document has finished conversion.
- Returns FALSE if the content item is still in GENWWW and has not finished conversion.

Example

```
<$wfIsFinishedDocConversion()$>
```

wfIsNotifyingUsers

Returns TRUE if this workflow is currently suppressing all workflow notifications for this particular workflow event. Suppression includes notification into both the workflow in queue and email until either the current action is ended or the `wfSetIsNotifyingUsers` function is used to re-enable notification.

Suppression of notifications is temporary. If notification is enabled or allowed to remain on for a later workflow action, all notifications that were not sent out for the current step are then sent out and workflows in queues are appropriated updated.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Output

- Returns TRUE if notifications are suppressed.
- Returns FALSE if notifications are being suppressed.

Example

```
<$wfIsNofifyingUsers()$>
```

See Also

- "wfSetIsNotifyingUsers" on page 6-302

wfIsReleasable

Indicates if the document is released as far as the workflow is concerned. The document may be still in GENWW or DONE, but if this Idoc Script function returns true, then the workflow is not preventing the release of the document. If it returns false, then the document will not be released until the workflow allows it to be released. This Idoc script function takes no parameters. It evaluates the active release state value for any document info that may be present (in Idoc script terms it checks if `#active.dReleaseState` is not the value `E`).

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Output

- Returns TRUE if the document is available to be released.
- Returns FALSE if the content item is not available to be released. For example, the content item has not completed the check in and/or conversion process.

Example

```
<$wfIsReleasable()$>
```

wfJumpEntryNotifyOff

Turns the jump entry notification on and off.

- If this variable is TRUE, reviewers will not be notified when the jump is entered.
- If this variable is FALSE, reviewers will be notified when the jump is entered.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns TRUE or FALSE.

Example

The following code sets the step entry notification to 0, which means that reviewers will be notified upon jump entry:

```
<$wfSet("wfJumpEntryNotifyOff","0")$>
```

wfJumpMessage

Defines a message that will be included in the notification e-mail that is sent to users when a jump is entered.

- If no message is specified, the e-mail message will include only the information in the e-mail template.
- The jump message can include Idoc Script, which must be executed using the eval function. For example:

  ```
  <$eval(dDocName)$> is ready for your review.
  ```

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the message as a string.

Example

The following code includes the message *This is the message text* in the e-mail message that notifies users upon jump entry:

```
<$wfSet("wfJumpMessage","This is the message text")$>
```

See Also

- "eval" on page 8-71

wfJumpName

Retrieves the name of the current jump.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the jump name as a string.

Example

The following code sets the name of the current jump to *MyJump*:

```
<$wfSet("wfJumpName","MyJump")$>
```

wfJumpReturnStep

Retrieves the name of the step in the parent workflow that the revision returns to upon exiting a workflow after the current jump.

- The return step applies only if the revision properly completes the last step in the workflow that was jumped to and exits the workflow normally. Consequently, the return step is *not* applied when the revision jumps to another workflow.
- In the companion file, the return step is stored in the local key:

  ```
  <step_name>@<workflow_name>.returnStep
  ```

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the step name as a string.

Example

The following code sets the return step to be the next step in the parent workflow:

```
<$wfSet("wfJumpReturnStep",wfCurrentStep(1))$>
```

wfJumpTargetStep

Retrieves the name of the step the revision will jump to if the condition is met.

The target step can be referred to symbolically (such as *wfCurrentStep(1)*) or explicitly (such as *MyStep@MyWorkflow*). It is strongly recommended that you use symbolic references in step event scripts. They make the script easier to modify and reuse.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the step name as a string.

Example

The following code sets the target step explicitly. When the jump criteria is met, the revision jumps to a step named `step_1` in the `Marketing` workflow:

```
<$wfSet("wfJumpTargetStep","step_1@Marketing")$>
```

The following code sets the target step symbolically. When the jump criteria is met, the revision jumps to the first step in the current workflow:

```
<$wfSet("wfJumpTargetStep",WfStart)$>
```

wfLoadDesign

This function is used to obtain information about the existing steps in a workflow or about exit conditions in a workflow.

This function loads the <*instance_dir*/data/workflow/design/*workflowname*.hda file and returns a ResultSet containing design information for a workflow. *workflowname* corresponds to the value for the `dWfName` variable, usually available on workflow pages, email templates, and jump scripts.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes one parameter, the name of the workflow.

Output

Returns the information as a ResultSet.

Example

```
<$wfLoadDesign(WorkflowName)$>
```

wfMailSubject

Defines the subject line of a workflow e-mail notification.

If no subject line is specified, the e-mail will use the default subject for the type of notification (review, reject, or workflow started).

Idoc Script can be included in the subject string.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the subject line as a string.

Example

Notifies the original author with the subject line *"Content ID* has been released":

```
<$wfMailSubject="File <$dDocName$> has been released"$>
<$wfNotify(dDocAuthor, "user")$>
```

See Also

- "wfNotify" on page 6-298
- "wfMessage" on page 6-297

wfMessage

Defines a message that will be included in a workflow e-mail notification.

If no message is specified, the e-mail will use the default message for the type of notification (review, reject, or workflow started).

Idoc Script can be included in the message string.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the message as a string.

Example

Notifies the original author with the message *"Content ID* has been released":

```
<$wfMessage="File <$dDocName$> has been released"$>
<$wfNotify(dDocAuthor, "user")$>
```

See Also

- "wfNotify" on page 6-298
- "wfMailSubject" on page 6-296

wfNotify

Sends an e-mail message to a specified user, alias, or workflow token.

The wfMailSubject and wfMessage variables can be set to customize the notification message.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes two parameters and an optional third parameter:

- The first parameter specifies the user name, alias, or token to be notified.
- The second parameter indicates the type, either *user, alias* or token.
- The optional third parameter specifies the name of the e-mail template to use for constructing the message. (See the *IdcHomeDir*/resources/core/templates/templates.hda file for template definitions.)

Example

Notifies the original author:

```
<$wfNotify(dDocAuthor, "user")$>
```

Notifies all users in the `myAlias` alias, using the *IdcHomeDir*/resources/core/templates/reject_mail.htm file as a template:

```
<$wfNotify("myAlias", "alias", "WF_REJECT_MAIL")$>
```

See Also

- "wfMailSubject" on page 6-296
- "wfMessage" on page 6-297

wfParentList

Retrieves a list (parent list) of the workflow steps that the revision has visited. This tells the system where jumps occurred and how to unwind the steps during an error, a reject, or an exit.

This variable can be used to create conditional statements, but it should not be hard-coded or altered.

If the parent list is unwound due to an error, reject, or exit, steps are removed from the list, so the parent list may not reflect the complete step history.

The parent list is global, and is not localized with a step name.

Steps in the parent list are listed with the most recent step first. Steps are separated with a pound sign (#). An asterisk before a step name indicates that it is a jump step. For example:

```
Step_B@Workflow_2#*Step_A@Workflow_2#Step_1@Workflow_1
```

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Output

Returns the parent list as a string.

Example

One use for the parent list is to simulate the jumps that a content item has visited when you are testing a workflow script. For example, on the Test tab or the Edit Script screen, selecting a content item and clicking Load Item's Workflow State could result in the following line:

```
wfParentList=step_1@Marketing#contribution@Marketing
```

wfReleaseDocument

Causes a workflow to release all outstanding document revisions for a document that are currently being locked by the workflow. Such released revisions are free to be evaluated by the indexing engine and one of the released revisions may be full-text indexed as a result.

This function can only be used in the context of the evaluation of Idoc Script for a workflow step going through a workflow event. The function will have no effect if the document has already been released by the workflow. Note that this function does not cause the document to exit workflow.

Any new revision that is created for the document will be back in a "locked" ("unreleasable") state. In other words, this function releases current revisions but has no implications for any new revision that may be created. This function takes no parameters and returns no result. It acts on the current active workflow document.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Example

```
<$wfReleaseDocument()$>
```

wfSet

Sets a key with a particular value in the companion file.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes two parameters:

- The first parameter is the key.
- The second parameter is the value.

Example

Sets the key `wfJumpName` to `MyJump`:

```
<$wfSet("wfJumpName", "MyJump")$>
```

See Also

- "wfCurrentSet" on page 6-281

wfSetIsNotifyingUsers

Sets an internal flag indicating if workflow notification will be sent out during the current action (check in, update, resubmit, and so on).

The suppression is on both email and updates to the workflow in queue. An additional use for this function is to suppress workflow notification until after a document has been converted. This prevents a document from advancing out of the auto-contributor workflow step when the document finishes a conversion.

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes one parameter, the internal flag. If set to 1 (TRUE), notifications are not suppressed. If set to 0 (FALSE), notifications are suppressed.

Example

```
<$wfSetIsNotifyingUsers(0)$>
```

See Also

- "wfIsNotifyingUsers" on page 6-288

WfStart

Sends the revision to the first step in the current workflow. Note that this variable begins with an uppercase W.

Type and Usage

- "Workflow Variables" on page 3-21
- "Workflow" on page 4-21

Example

Sets the target step for a jump to restart the workflow:

```
<$wfSet("wfJumpTargetStep",WfStart)$>
```

wfUpdateMetaData

Defines a metadata value for the current content item revision in a workflow.

The *wfUpdateMetaData* function can be used only for updating custom metadata fields. You cannot use this function to update standard, predefined fields

Type and Usage

- "Workflow Functions" on page 3-21
- "Workflow" on page 4-21

Parameters

Takes two parameters:

- The first parameter is the name of the metadata field.
- The second parameter is the value of the metadata field.

Example

Defines "This is my comment." as the value of the Comments field:

```
<$wfUpdateMetaData("xComments", "This is my comment.")$>
```

WorkflowDir

Defines the path to the *workflow* directory, which contains workflow data.

Default is *instance_dir/data/workflow/*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9
- "Workflow" on page 4-21

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
WorkflowDir=c:/workflow_info/
```

WorkflowIntervalHours

Defines the time between workflow update cycles, in hours.

The update event scripts for any revisions in a workflow are executed at this interval.

Default is 1 hour.

Type and Usage

- "Configuration Variables" on page 3-2
- "Workflow" on page 4-21

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
WorkflowIntervalHours=2
```

XMLEncodingMode

Used to specify international characters in URLs in the browser used for the Content Server.

Setting this variable to FULL may be useful in the following instances:

- If a Content Server is being used in a browser that is using UTF-8 (that is, with the setting "Always send URLs as UTF-8" enabled). This usually occurs only in older versions of Microsoft browsers.
- If a custom component is created that puts Content Server URLs into Javascript variables. In that case, the URLs may need to be fully encoded.
- If a Content Server is being accessed through a PDA-type device (PALM, handheld computer, and so on). If a component is used that provides accessibility through these devices, then the URLs may need to be fully encoded. These devices may not support non-7-bit characters in their URLs.

Default is PARTIAL.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
XMLEncodingMode=full
```

7

Management and Configuration Options by Type and Usage

This chapter describes the Idoc Script variables and functions used to manage and configure Content Server, and groups the variables and functions by *type*.

This chapter also groups Idoc Script variables and functions by *common usage*.

- "Users" on page 7-17

7.1 Conditional Dynamic Variables

Some dynamic variables are conditional and can only be used within a conditional statement such as *if*, *while*, *elseif*, or *loop*. For a list of conditional dynamic variables and details about their use, see "Conditional Dynamic Variables" on page 3-1.

The following is a list of Idoc Script conditional dynamic variables relating to configuration:

"AfterLogin" on page 8-5
"HeavyClient" on page 8-121
"IsDynamic" on page 8-159
"IsExternalUser" on page 8-162
"IsMac" on page 8-174
"IsMaxRows" on page 8-175
"IsSun" on page 8-188
"IsUserEmailPresent" on page 8-192
"IsWindows" on page 8-194
"UserLanguageID" on page 8-347
"UserLocaleId" on page 8-348

7.2 Configuration Variables

The following configuration variables are used when customizing and configuring the Content Server.

"AccessListPrivilegesGrantedWhenEmpty" on page 8-3
"AllowMatchesInDateCheck" on page 8-6
"ARCHIVE_COLLECTION" on page 8-7
"ArchiverDoLocks" on page 8-8
"ArchiverTouchMonitorInterval" on page 8-9
"ArchiverTraceLockMaintenance" on page 8-10
"ArchiverTraceLocks" on page 8-11
"BatchLoaderPath" on page 8-13
"BatchLoaderUserName" on page 8-14
"BinDir" on page 8-15
"ChunkedRequestTrace" on page 8-22
"CleanUp" on page 8-23
"CollectionLocation" on page 8-25
"ConfigDir" on page 8-26
"DataDir" on page 8-30
"DateOutputFormat" on page 8-32
"DisableSharedCacheChecking" on page 8-59
"DirectoryLockingLogPath" on page 8-56
"DownloadApplet" on page 8-63
"EnableEnvironmentPackager" on page 8-69
"EnableErrorFile" on page 8-70
"FileEncoding" on page 8-92
"ForceDocTypeChoice" on page 8-93
"ForceSecurityGroupChoice" on page 8-96
"getFreeMemory" on page 8-109
"getTotalMemory" on page 8-114
"HTMLEditorPath" on page 8-122
"IdcAnalyzeLogDir" on page 8-142

"lcCaption" on page 8-202
"IdcLocale" on page 8-144
"IdcServerBindAddress" on page 8-145
"IdcServerSocketQueueDepth" on page 8-146
"IntradocDir" on page 8-152
"isActiveTrace" on page 8-153
"IsAutoArchiver" on page 8-154
"IsJspServerEnabled" on page 8-173
"IsOverrideFormat" on page 8-179
"isPhysicallySplitDir" on page 8-180
"isVerboseTrace" on page 8-193
"IsXml" on page 8-195
"JspAdminQuery" on page 8-197
"JspDefaultIndexPage" on page 8-198
"JspEnabledGroups" on page 8-199
"LocaleArchiverDateFormat" on page 8-207
"LogMessagePrefix" on page 8-209
"LogName" on page 8-210
"MaxArchiveErrorsAllowed" on page 8-212
"MaxErrorsAllowed" on page 8-213
"MacSupportsSignedApplets" on page 8-211
"MultiUpload" on page 8-216
"NotationForRequiredFields" on page 8-220
"OdbcDateFormat" on page 8-221
"ProviderDir" on page 8-236
"RequestAuditAdditionalVerboseFieldsList" on page 8-237
"RequestAuditIntervalSeconds1" on page 8-238
"RequestAuditIntervalSeconds2" on page 8-239
"RequestAuditListDepth1" on page 8-240
"RequestAuditListDepth2" on page 8-241
"RequestAuditSubClassificationField" on page 8-242
"rptDisplayMapValue" on page 8-244
"SearchDir" on page 6-228
"showDocActions" on page 8-292
"ShowRevisionsColumn" on page 8-293
"StrConfineOverflowChars" on page 8-299
"StyleForRequiredFields" on page 8-316
"SubjectsDir" on page 8-317
"SysAdminAddress" on page 8-318
"SystemDateFormat" on page 8-319
"SystemLocale" on page 8-320
"SystemTimeZone" on page 8-321
"TempDir" on page 8-322
"TraceResourceConflict" on page 8-330
"TraceResourceLoad" on page 8-331
"TraceResourceOverride" on page 8-332
"UseFourDigitYear" on page 8-338
"UploadApplet" on page 8-334
"UserCache" on page 8-342
"UserCacheTimeout" on page 8-343
"UserLanguageID" on page 8-347
"UserLocaleId" on page 8-348
"UserPublishCacheDir" on page 8-350
"UseRedirectedOutput" on page 8-345
"WarnAboutTimeZone" on page 8-359

"WebBrowserPath" on page 8-360

7.3 Global Functions

Idoc Script has many built-in global functions. For a list of conditional dynamic variables and details about their use, see "Global Functions" on page 3-11.

The following is a list of Idoc Script global functions relating to configuration and usage:

"abortToErrorPage" on page 8-2
"break" on page 8-16
"cacheInclude" on page 8-19
"dateCurrent" on page 8-31
"ddAppendIndexedColumnResultSet" on page 8-33
"ddAppendResultSet" on page 8-35
"ddApplyTableSortToResultSet" on page 8-36
"ddGetFieldList" on page 8-37
"ddIncludePreserveValues" on page 8-38
"ddLoadIndexedColumnResultSet" on page 8-40
"ddLoadResultSet" on page 8-41
"ddMergeIndexedColumnResultSet" on page 8-42
"ddMergeResultSet" on page 8-44
"ddMergeUsingIndexedKey" on page 8-46
"ddSetLocal" on page 8-48
"ddSetLocalByColumnsFromFirstRow" on page 8-49
"ddSetLocalByColumnsFromFirstRowIndexed" on page 8-50
"ddSetLocalEmpty" on page 8-51
"ddSetLocalEmptyByColumns" on page 8-52
"docLoadResourceIncludes" on page 8-60
"docRootFilename" on page 8-62
"dpGet" on page 8-65
"dpPromote" on page 8-66
"dpPromoteRs" on page 8-67
"dpSet" on page 8-68
"eval" on page 8-71
"executeService" on page 8-73
"forceExpire" on page 8-94
"formatDate" on page 8-97
"formatDateDisplay" on page 8-98
"formatDateOnly" on page 8-99
"formatDateOnlyDisplay" on page 8-100
"formatDateOnlyFull" on page 8-101
"formatDateWithPattern" on page 8-102
"formatTimeOnly" on page 8-103
"formatTimeOnlyDisplay" on page 8-104
"getDebugTrace" on page 8-106
"getErrorTrace" on page 8-107
"getFieldViewValue" on page 8-108
"getFreeMemory" on page 8-109
"getRequiredMsg" on page 8-112
"getTextFile" on page 8-113
"getTotalMemory" on page 8-114
"getUserValue" on page 8-115
"getValue" on page 8-116
"idocTestForInclude" on page 8-147

"inc" on page 8-148
"incGlobal" on page 8-149
"incTemplate" on page 8-151
"isActiveTrace" on page 8-153
"isFalse" on page 8-163
"isTrue" on page 8-189
"js" on page 8-196
"lc" on page 8-201
"lcCaption" on page 8-202
"loadDocMetaDefinition" on page 8-203
"loadDocumentProfile" on page 8-204
"loadSearchOperatorTables" on page 8-205
"loadUserMetaDefinition" on page 8-206
"optList" on page 8-228
"parseDataEntryDate" on page 8-231
"parseDateWithPattern" on page 8-234
"pneNavigation" on page 8-235
"rptDisplayMapValue" on page 8-244
"rs" on page 8-245
"rsAddFields" on page 8-246
"rsAddFieldsWithDefaults" on page 8-247
"rsAddRowCountColumn" on page 8-249
"rsAppend" on page 8-250
"rsAppendNewRow" on page 8-251
"rsAppendRowvalues" on page 8-252
"rsCopyFiltered" on page 8-253
"rsCreateReference" on page 8-254
"rsCreateResultSet" on page 8-255
"rsDeleteRow" on page 8-256
"rsExists" on page 8-258
"rsFieldByIndex" on page 8-259
"rsFieldExists" on page 8-260
"rsFindRowPrimary" on page 8-261
"rsFirst" on page 8-262
"rsInsertNewRow" on page 8-263
"rsIsRowPresent" on page 8-264
"rsLoopInclude" on page 8-265
"rsLoopSingleRowInclude" on page 8-266
"rsMakeFromList" on page 8-267
"rsMakeFromString" on page 8-269
"rsMerge" on page 8-271
"rsMergeDelete" on page 8-272
"rsMergeReplaceOnly" on page 8-273
"rsNext" on page 8-274
"rsNumFields" on page 8-275
"rsNumRows" on page 8-276
"rsRemove" on page 8-277
"rsRename" on page 8-278
"rsRenameField" on page 8-279
"rsSetRow" on page 8-280
"rsSort" on page 8-281
"rsSortTree" on page 8-282
"setContentType" on page 8-286
"setExpires" on page 8-287
"setHttpHeader" on page 8-288

"setMaxAge" on page 8-289
"setValue" on page 8-291
"setResourceInclude" on page 8-290
"strCenterPad" on page 8-296
"strCommaAppendNoDuplicates" on page 8-297
"strConfine" on page 8-298
"strEquals" on page 8-300
"strEqualsIgnoreCase" on page 8-301
"strGenerateRandom" on page 8-302
"strIndexOf" on page 8-303
"strLeftFill" on page 8-304
"strLeftPad" on page 8-305
"strLength" on page 8-306
"strLower" on page 8-307
"strRemoveWs" on page 8-308
"strReplace" on page 8-309
"strReplaceIgnoreCase" on page 8-310
"strRightFill" on page 8-311
"strRightPad" on page 8-312
"strSubstring" on page 8-313
"strTrimWs" on page 8-314
"strUpper" on page 8-315
"toInteger" on page 8-327
"trace" on page 8-328
"url" on page 8-335
"urlEscape7Bit" on page 8-336
"utGetValue" on page 8-353
"utLoad" on page 8-354
"utLoadDocumentProfiles" on page 8-355
"utLoadResultSet" on page 8-356
"xml" on page 8-362

7.4 Page Variables

Page variables are set on a particular Content Server Web page to enable specific page attributes or functionality. For details about page display variables, see "Page Variables" on page 3-15.

This section includes these topics:

- "Page Display Variables" on page 7-6
- "Field Display Variables" on page 7-7

7.4.1 Page Display Variables

The following is a list of Idoc Script page display variables:

"generateUniqueId" on page 8-105
"isCheckin" on page 8-155
"isDocPage" on page 8-157
"isEditMode" on page 8-160
"isFormSubmit" on page 8-168
"isInfo" on page 8-170
"isQuery" on page 8-181
"isUpdate" on page 8-190
"isUploadFieldScript" on page 8-191

"localPageType" on page 8-208
"noMCPrefill" on page 8-218

7.4.2 Field Display Variables

Field display variables can be grouped into the following types:

- "Field Information Variables" on page 7-7
- "Common Field Display Variables" on page 7-7
- "Other Field Display Variables" on page 7-7

For details about these variables, see "Field Display Variables" on page 3-15.

7.4.2.1 Field Information Variables

The following variables define information about a metadata field. The variable values are loaded or computed for each metadata field.

"fieldCaption" on page 8-76
"fieldDefault" on page 8-79
"fieldIsOptionList" on page 8-84
"fieldName" on page 8-86
"fieldOptionListType" on page 8-87
"fieldType" on page 8-88
"fieldValue" on page 8-89
"fieldWidth" on page 8-91
"isRequired" on page 8-184
"requiredMsg" on page 8-243
"valueStyle" on page 8-357

7.4.2.2 Common Field Display Variables

There are several commonly used page variables that affect the display of metadata fields on Content Server pages.

The following is a list of the common field display variables:

"isExcluded" on page 8-161
"isHidden" on page 8-169
"isInfoOnly" on page 8-171
"isRelocated" on page 8-182
"maxLength" on page 8-214
"optionListScript" on page 8-225

7.4.2.3 Other Field Display Variables

Several other variables are available to affect the display of metadata fields.

"addEmptyOption" on page 8-4
"captionEntryWidth" on page 8-20
"captionFieldWidth" on page 8-21
"defaultFieldInclude" on page 8-53
"defaultOptionListScript" on page 8-54
"fieldCaptionInclude" on page 8-77
"fieldCaptionStyle" on page 8-78
"fieldEditWidth" on page 8-80
"fieldEntryInclude" on page 8-81
"fieldExtraScriptInclude" on page 8-82
"fieldInclude" on page 8-83

"fieldMaxLength" on page 8-85
"fieldValueStyle" on page 8-90
"hasOptionList" on page 8-120
"isFieldExcluded" on page 8-164
"isFieldHidden" on page 8-165
"isFieldInfoOnly" on page 8-166
"isFieldMemo" on page 8-167
"isMultiOption" on page 8-176
"isStrictList" on page 8-187
"NotationForRequiredFields" on page 8-220
"optionListKey" on page 8-222
"optionListName" on page 8-223
"optionListResultSet" on page 8-224
"optionListValueInclude" on page 8-226
"optionsAllowPreselect" on page 8-227
"StyleForRequiredFields" on page 8-316

7.5 Read-Only Variables

Read-only variables can be used to gather information about the current template, the user who is currently logged in, or other current settings. These variables are read-only and cannot be assigned a value.

- "Template Read-Only Variables" on page 7-8
- "User Read-Only Variables" on page 7-8
- "Other Read-Only Variables" on page 7-8

7.5.1 Template Read-Only Variables

Template-related read-only variables make it possible to create conditional content in a template based on the identity of the template. For details and a list of all template read-only variables, see "Template Read-Only Variables" on page 3-18.

"TemplateClass" on page 8-323
"TemplateFilePath" on page 8-324
"TemplateName" on page 8-325
"TemplateType" on page 8-326

7.5.2 User Read-Only Variables

User-related read-only variables make it possible to gather information about the current user.

"UserAddress" on page 8-340
"UserFullName" on page 8-346
"UserName" on page 8-349

7.5.3 Other Read-Only Variables

The following variable is set only as internal Content Server flags, so it can be retrieved but not set directly.

"SafeDir" on page 8-283

7.6 Setable Variables

Setable variables can be set within script or used within a CGI string. For details about setable variables, see "Setable Variables" on page 3-19.

"ClientControlled" on page 8-24
"coreContentOnly" on page 8-27
"IsJava" on page 8-172
"IsSoap" on page 8-186
"IsXml" on page 8-195
"ScriptDebugTrace" on page 8-284
"ScriptErrorTrace" on page 8-285

7.7 Value Variables

A *value* variable is evaluated once at the beginning of a service call and that value is used throughout the service call. For details about value variables, see "Value Variables" on page 3-21.

"AuthorAddress" on page 8-12
"BrowserVersionNumber" on page 8-17
"CURRENT_DATE" on page 8-28
"CURRENT_ROW" on page 8-29
"htmlRefreshTimeout" on page 8-123
"htmlRefreshUrl" on page 8-124
"HttpBrowserFullCgiPath" on page 8-129
"HttpWebRoot" on page 8-141
"MSIE" on page 8-215
"PageParent" on page 8-230

7.8 Archiver

The following Idoc Script variables are related to the Archiver.

"AllowMatchesInDateCheck" on page 8-6
"ARCHIVE_COLLECTION" on page 8-7
"ArchiverDoLocks" on page 8-8
"ArchiverTouchMonitorInterval" on page 8-9
"ArchiverTraceLockMaintenance" on page 8-10
"ArchiverTraceLocks" on page 8-11
"CollectionLocation" on page 8-25
"IsAutoArchiver" on page 8-154
"LocaleArchiverDateFormat" on page 8-207
"MaxArchiveErrorsAllowed" on page 8-212
"OdbcDateFormat" on page 8-221

7.9 Batch Loader

The following Idoc Script variables are related to the Batch Loader.

"BatchLoaderPath" on page 8-13
"BatchLoaderUserName" on page 8-14
"CleanUp" on page 8-23
"EnableErrorFile" on page 8-70
"MaxErrorsAllowed" on page 8-213

7.10 Clients

The following Idoc Script variables are related to client applications.

"BrowserVersionNumber" on page 8-17
"ClientControlled" on page 8-24
"DownloadApplet" on page 8-63
"HeavyClient" on page 8-121
"HttpBrowserFullCgiPath" on page 8-129
"IsMac" on page 8-174
"IsSun" on page 8-188
"IsWindows" on page 8-194
"MacSupportsSignedApplets" on page 8-211
"MSIE" on page 8-215
"MultiUpload" on page 8-216
"showDocActions" on page 8-292
"UploadApplet" on page 8-334

7.11 Content Profiles

The following functions are used to manage Content Profiles.
"dpGet" on page 8-65
"dpPromote" on page 8-66
"dpPromoteRs" on page 8-67
"dpSet" on page 8-68
"DisableContentProfiles" on page 8-57
"getFieldViewValue" on page 8-108
"getRequiredMsg" on page 8-112
"IsDpSubmitErrorFatal" on page 8-158
"loadDocumentProfile" on page 8-204
"utLoadDocumentProfiles" on page 8-355

7.12 Date and Time

The following Idoc Script variables and functions are related to the formatting and manipulation of dates and times.

"CURRENT_DATE" on page 8-28
"DateOutputFormat" on page 8-32
"dateCurrent" on page 8-31
"formatDate" on page 8-97
"formatDateOnly" on page 8-99
"formatDateOnlyDisplay" on page 8-100
"formatDateOnlyFull" on page 8-101
"formatDateWithPattern" on page 8-102
"formatTimeOnly" on page 8-103
"parseDataEntryDate" on page 8-231
"parseDate" on page 8-232
"parseDateWithPattern" on page 8-234
"SystemDateFormat" on page 8-319
"SystemTimeZone" on page 8-321
"UseFourDigitYear" on page 8-338
"WarnAboutTimeZone" on page 8-359

7.13 Debugging

The following Idoc Script variables and functions are related to debugging.

"ArchiverTraceLocks" on page 8-11
"ArchiverTraceLockMaintenance" on page 8-10
"ChunkedRequestTrace" on page 8-22
"DirectoryLockingLogPath" on page 8-56
"EnableEnvironmentPackager" on page 8-69
"EnableErrorFile" on page 8-70
"getDebugTrace" on page 8-106
"getErrorTrace" on page 8-107
"getFreeMemory" on page 8-109
"getTotalMemory" on page 8-114
"IdcAnalyzeLogDir" on page 8-142
"isActiveTrace" on page 8-153
"IsJava" on page 8-172
"IsSoap" on page 8-186
"isVerboseTrace" on page 8-193
"IsXml" on page 8-195
"LogMessagePrefix" on page 8-209
"LogName" on page 8-210
"RequestAuditAdditionalVerboseFieldsList" on page 8-237
"RequestAuditIntervalSeconds1" on page 8-238
"RequestAuditIntervalSeconds2" on page 8-239
"RequestAuditListDepth1" on page 8-240
"RequestAuditListDepth2" on page 8-241
"RequestAuditSubClassificationField" on page 8-242
"ScriptDebugTrace" on page 8-284
"ScriptErrorTrace" on page 8-285
"TempDir" on page 8-322
"trace" on page 8-328
"TraceResourceConflict" on page 8-330
"TraceResourceLoad" on page 8-331
"TraceResourceOverride" on page 8-332
"UseRedirectedOutput" on page 8-345

7.14 Directories and Paths

The following Idoc Script variables and functions are related to directories and file paths.

"BinDir" on page 8-15
"CollectionLocation" on page 8-25
"ConfigDir" on page 8-26
"DataDir" on page 8-30
"docRootFilename" on page 8-62
"DownloadSuggestedName" on page 8-64
"HTMLEditorPath" on page 8-122
"HttpAbsoluteCgiPath" on page 8-125
"HttpAbsoluteWebRoot" on page 8-127
"HttpAbsoluteCgiRoot" on page 8-126
"HttpBrowserFullCgiPath" on page 8-129
"HttpAdminCgiPath" on page 8-128
"HttpCgiPath" on page 8-130
"HttpCommonRoot" on page 8-131

"HttpEnterpriseCgiPath" on page 8-132
"HttpHelpRoot" on page 8-133
"HttpImagesRoot" on page 8-134
"HttpLayoutRoot" on page 8-135
"HttpRelativeAdminRoot" on page 8-136
"HttpRelativeCgiRoot" on page 8-137
"HttpRelativeWebRoot" on page 8-138
"HttpServerAddress" on page 8-139
"HttpSystemHelpRoot" on page 8-140
"HttpWebRoot" on page 8-141
"IdcAnalyzeLogDir" on page 8-142
"IntradocDir" on page 8-152
"isPhysicallySplitDir" on page 8-180
"ProviderDir" on page 8-236
"SafeDir" on page 8-283
"SearchDir" on page 6-228
"SubjectsDir" on page 8-317
"TempDir" on page 8-322
"UserCache" on page 8-342
"WebBrowserPath" on page 8-360

7.15 Dynamichtml

The following Idoc Script functions are related to the use of dynamichtml.

"ddAppendIndexedColumnResultSet" on page 8-33
"ddAppendResultSet" on page 8-35
"ddApplyTableSortToResultSet" on page 8-36
"ddGetFieldList" on page 8-37
"ddIncludePreserveValues" on page 8-38
"ddLoadIndexedColumnResultSet" on page 8-40
"ddLoadResultSet" on page 8-41
"ddMergeIndexedColumnResultSet" on page 8-42
"ddMergeResultSet" on page 8-44
"ddMergeUsingIndexedKey" on page 8-46
"ddSetLocal" on page 8-48
"ddSetLocalByColumnsFromFirstRow" on page 8-49
"ddSetLocalByColumnsFromFirstRowIndexed" on page 8-50
"ddSetLocalEmpty" on page 8-51
"ddSetLocalEmptyByColumns" on page 8-52
"idocTestForInclude" on page 8-147

7.16 Field Display

The following Idoc Script variables and functions are related to the display of metadata fields on Content Server Web pages.

"addEmptyOption" on page 8-4
"captionEntryWidth" on page 8-20
"captionFieldWidth" on page 8-21
"defaultFieldInclude" on page 8-53
"defaultOptionListScript" on page 8-54
"fieldCaption" on page 8-76
"fieldCaptionInclude" on page 8-77
"fieldCaptionStyle" on page 8-78
"fieldDefault" on page 8-79

"fieldEditWidth" on page 8-80
"fieldEntryInclude" on page 8-81
"fieldExtraScriptInclude" on page 8-82
"fieldInclude" on page 8-83
"fieldIsOptionList" on page 8-84
"fieldMaxLength" on page 8-85
"fieldName" on page 8-86
"fieldOptionListType" on page 8-87
"fieldType" on page 8-88
"fieldValue" on page 8-89
"fieldValueStyle" on page 8-90
"fieldWidth" on page 8-91
"ForceDocTypeChoice" on page 8-93
"ForceSecurityGroupChoice" on page 8-96
"hasOptionList" on page 8-120
"isExcluded" on page 8-161
"isFieldExcluded" on page 8-164
"isFieldHidden" on page 8-165
"isFieldInfoOnly" on page 8-166
"isFieldMemo" on page 8-167
"isHidden" on page 8-169
"isInfoOnly" on page 8-171
"isMultiOption" on page 8-176
"IsOverrideFormat" on page 8-179
"isRelocated" on page 8-182
"isRequired" on page 8-184
"isStrictList" on page 8-187
"maxLength" on page 8-214
"NotationForRequiredFields" on page 8-220
"optionListKey" on page 8-222
"optionListName" on page 8-223
"optionListResultSet" on page 8-224
"optionListScript" on page 8-225
"optionListValueInclude" on page 8-226
"optionsAllowPreselect" on page 8-227
"optList" on page 8-228
"requiredMsg" on page 8-243
"ShowRevisionsColumn" on page 8-293
"StyleForRequiredFields" on page 8-316
"valueStyle" on page 8-357

7.17 Idoc Script

The following Idoc Script functions and keywords are not related to any specific Content Server functionality.

"break" on page 8-16
"c" on page 8-18
"cacheInclude" on page 8-19
"docLoadResourceIncludes" on page 8-60
"eval" on page 8-71
"exec" on page 8-72
"executeService" on page 8-73
"forceExpire" on page 8-94
"getTextFile" on page 8-113

"getUserValue" on page 8-115
"getValue" on page 8-116
"inc" on page 8-148
"incGlobal" on page 8-149
"include" on page 8-150
"incTemplate" on page 8-151
"isFalse" on page 8-163
"isTrue" on page 8-189
"setExpires" on page 8-287
"setHttpHeader" on page 8-288
"setMaxAge" on page 8-289
"setResourceInclude" on page 8-290

7.18 JSP

The following Idoc Script variables are related to JSP support.

"IsJspServerEnabled" on page 8-173
"JspAdminQuery" on page 8-197
"JspDefaultIndexPage" on page 8-198
"JspEnabledGroups" on page 8-199

7.19 Localization

The following Idoc Script variables and functions are related to localization.

"FileEncoding" on page 8-92
"IdcLocale" on page 8-144
"lcCaption" on page 8-202
"lc" on page 8-201
"LocaleArchiverDateFormat" on page 8-207
"rptDisplayMapValue" on page 8-244
"SystemDateFormat" on page 8-319
"SystemLocale" on page 8-320
"SystemTimeZone" on page 8-321
"UserLanguageID" on page 8-347
"UserLocaleId" on page 8-348
"WarnAboutTimeZone" on page 8-359

7.20 Page Display

The following Idoc Script variables and functions are related to the display of Content Server Web pages.

"abortToErrorPage" on page 8-2
"AfterLogin" on page 8-5
"coreContentOnly" on page 8-27
"docLoadResourceIncludes" on page 8-60
"generateUniqueId" on page 8-105
"getHelpPage" on page 8-110
"getOptionListSize" on page 8-111
"htmlRefreshTimeout" on page 8-123
"htmlRefreshUrl" on page 8-124
"isCheckin" on page 8-155
"IsCurrentNav" on page 8-156
"isDocPage" on page 8-157

"IsDynamic" on page 8-159
"isEditMode" on page 8-160
"isFormSubmit" on page 8-168
"isInfo" on page 8-170
"IsMaxRows" on page 8-175
"IsMultiPage" on page 8-177
"isNew" on page 8-178
"isQuery" on page 8-181
"IsSavedQuery" on page 8-185
"isUpdate" on page 8-190
"isUploadFieldScript" on page 8-191
"localPageType" on page 8-208
"noInfoColumn" on page 8-217
"noMCPrefill" on page 8-218
"noRevisionsColumn" on page 8-219
"PageParent" on page 8-230
"pneNavigation" on page 8-235
"ResultsTitle" on page 6-207
"showDocActions" on page 8-292
"showSearchResultsCheckoutColumn" on page 8-294
"StdPageWidth" on page 8-295
"TemplateClass" on page 8-323
"TemplateFilePath" on page 8-324
"TemplateName" on page 8-325
"TemplateType" on page 8-326
"UseBellevueLook" on page 8-337
"UseXpedioLook" on page 8-352

7.21 Personalization

The following Idoc Script functions are related to user topic (personalization) files.

"utGetValue" on page 8-353
"utLoad" on page 8-354
"utLoadResultSet" on page 8-356

7.22 ResultSets

The following Idoc Script variables and functions are related to ResultSets.

"CURRENT_ROW" on page 8-29
"getValue" on page 8-116
"loadDocMetaDefinition" on page 8-203
"loadSearchOperatorTables" on page 8-205
"loadUserMetaDefinition" on page 8-206
"rs" on page 8-245
"rsAddFields" on page 8-246
"rsAddFieldsWithDefaults" on page 8-247
"rsAddRowCountColumn" on page 8-249
"rsAppend" on page 8-250
"rsAppendNewRow" on page 8-251
"rsAppendRowvalues" on page 8-252
"rsCopyFiltered" on page 8-253
"rsCreateReference" on page 8-254
"rsCreateResultSet" on page 8-255
"rsDeleteRow" on page 8-256

"rsExists" on page 8-258
"rsFieldByIndex" on page 8-259
"rsFieldExists" on page 8-260
"rsFindRowPrimary" on page 8-261
"rsFirst" on page 8-262
"rsInsertNewRow" on page 8-263
"rsIsRowPresent" on page 8-264
"rsLoopInclude" on page 8-265
"rsLoopSingleRowInclude" on page 8-266
"rsMakeFromList" on page 8-267
"rsMakeFromString" on page 8-269
"rsMerge" on page 8-271
"rsMergeDelete" on page 8-272
"rsMergeReplaceOnly" on page 8-273
"rsNext" on page 8-274
"rsNumFields" on page 8-275
"rsNumRows" on page 8-276
"rsRemove" on page 8-277
"rsRename" on page 8-278
"rsRenameField" on page 8-279
"rsSetRow" on page 8-280
"rsSort" on page 8-281
"rsSortTree" on page 8-282
"setContentType" on page 8-286
"setValue" on page 8-291
"utLoadResultSet" on page 8-356

7.23 Strings

The following Idoc Script variables and functions are related to strings.

"js" on page 8-196
"strCenterPad" on page 8-296
"strCommaAppendNoDuplicates" on page 8-297
"strConfine" on page 8-298
"StrConfineOverflowChars" on page 8-299
"strEquals" on page 8-300
"strEqualsIgnoreCase" on page 8-301
"strGenerateRandom" on page 8-302
"strIndexOf" on page 8-303
"strLeftFill" on page 8-304
"strLeftPad" on page 8-305
"strLength" on page 8-306
"strLower" on page 8-307
"strRemoveWs" on page 8-308
"strReplace" on page 8-309
"strReplaceIgnoreCase" on page 8-310
"strRightFill" on page 8-311
"strRightPad" on page 8-312
"strSubstring" on page 8-313
"strTrimWs" on page 8-314
"strUpper" on page 8-315
"toInteger" on page 8-327
"url" on page 8-335
"urlEscape7Bit" on page 8-336

"xml" on page 8-362

7.24 Templates

The following Idoc Script variables are related to Content Server templates.

"TemplateClass" on page 8-323
"TemplateFilePath" on page 8-324
"TemplateName" on page 8-325
"TemplateType" on page 8-326

7.25 Users

The following Idoc Script variables and functions are related to Content Server users.

"AuthorAddress" on page 8-12
"DelimitedUserRoles" on page 8-55
"ExternalUserAccounts" on page 8-74
"ExternalUserRoles" on page 8-75
"getUserValue" on page 8-115
"getValueForSpecifiedUser" on page 8-118
"IsExternalUser" on page 8-162
"IsLoggedIn" on page 6-120
"IsUserEmailPresent" on page 8-192
"SysAdminAddress" on page 8-318
"UserAccounts" on page 8-339
"UserAddress" on page 8-340
"UserAppRights" on page 8-341
"UserCacheTimeout" on page 8-343
"UserDefaultAccount" on page 8-344
"UserFullName" on page 8-346
"UserName" on page 8-349
"UserRoles" on page 8-351

8

Management and Configuration Variables and Functions

This chapter lists Idoc Script variables and functions in alphabetical order, and provides details about each item.

abortToErrorPage

Aborts the current page and displays an error message.

- This function evaluates the StatusCode variable, and if a negative numeric value (-1) is returned, the display of the current page is aborted and is substituted with an error page.
- The StatusMessage variable can be used as the error message string.

Type and Usage

- "Global Functions" on page 3-11
- "Page Display" on page 4-14

Parameters

The only parameter is the error message string.

Output

Returns the error message string on an error page.

Example

Aborts the current page and displays *Access Denied* as an error message:

```
<$abortToErrorPage("Access Denied")$>
```

Aborts the current page and displays the value of the *StatusMessage* variable as an error message:

```
<$abortToErrorPage("<$StatusMessage$>")$>
```

See Also

- "executeService" on page 8-73
- "IsRequestError" on page 8-183
- "StatusCode" on page 6-249
- "StatusMessage" on page 6-250

AccessListPrivilegesGrantedWhenEmpty

This configuration variable changes the default behavior of core collaboration, where an empty access list allows access to everyone, to instead deny access to everyone.

Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Parameters

There are no parameters, just the setting of 0 to deny access to everyone.

Example

This example denies access to everyone when the access list is empty:

```
AccessListPrivilegesGrantedWhenEmpty=0
```

addEmptyOption

Specifies that a metadata field option list has blank value as the first option in the list.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the first value in the option is blank.
- Returns FALSE if the first value in the option list is not blank.

Example

Specifies that the first value in the option list is blank:

```
<$if ForceDocTypeChoice and isTrue(ForceDocTypeChoice)$>
    <$addEmptyOption = 1$>
<$endif$>
```

See Also

- "fieldIsOptionList" on page 8-84

AfterLogin

Specifies whether the current page was created immediately after a login.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Page Display" on page 4-14

Output

- Returns TRUE if the page was created immediately after a login.
- Returns FALSE if the page was not created immediately after a login.

Example

Displays an alternate URL if the page was not created immediately after a login:

```
<$if not AfterLogin$>
    <a href="<$redirect$>">
<$endif$>
```

AllowMatchesInDateCheck

When set to TRUE, the date validation code is changed so that dates that match a prior revision date are valid.

- Setting this variable allows an archive import and batch load process to import revisions of a content item that have matching creation dates or release dates.
- Standard date validation in the archiver normally disallows an import of a revision that matches or is within a minute of the prior revision of a document.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2

Location

IntradocDir/config/config.cfg

Example

```
AllowMatchesInDateCheck=TRUE
```

ARCHIVE_COLLECTION

Defines the path to the default archive collection directory.

Default is *DomainHome*/ucm/cs/archives/.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2

Location

- *IntradocDir*/config/config.cfg
- *DomainHome*/ucm/cs/bin/intradoc.cfg

This setting must be included in both configuration files.

Example

As a configuration setting:

```
ARCHIVE_COLLECTION=f:/domain_archives/
```

ArchiverDoLocks

Activates a locking algorithm on archive files at the file system level.

- If you do automated imports or transfers with multiple Content Servers sharing an instance, set this variable to TRUE.
- If only one instance is doing automated imports or transfers in a load balancing configuration, then this variable does not need to be set.
- There is no default setting.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
ArchiverDoLocks=true
```

See Also

- "ArchiverTraceLocks" on page 8-11
- "ArchiverTraceLockMaintenance" on page 8-10
- "ArchiverTouchMonitorInterval" on page 8-9

ArchiverTouchMonitorInterval

Defines the time that a contending Content Server will wait before it will treat an archive lock file whose timestamp has not changed as having expired. The Content Server will also touch any locks it is maintaining twice during this interval.

- An archive lock is created by placing a lock file into an appropriate directory. In order for the lock to persist, the timestamp on the file must change periodically; otherwise any contending Content Server will treat the lock as an expired lock.
- The smaller the value of this setting, the quicker that locks can alternate between Content Servers. The larger the value, the more likely that high activity (which could keep a touch request delayed for many seconds) on the network drive will cause a Content Server to falsely report a lock as having expired. If a lock is falsely expired, it can cause duplicated effort (although it should not have any long term impact on the state of the Content Server).
- The default is 12000 milliseconds (12 seconds).

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
ArchiverTouchMonitorInterval=15000
```

See Also

- "ArchiverDoLocks" on page 8-8
- "ArchiverTraceLocks" on page 8-11
- "ArchiverTraceLockMaintenance" on page 8-10

ArchiverTraceLockMaintenance

Dumps trace information to the console for timestamp maintenance on any outstanding archive locks and the discovery of timestamp changes on locks maintained by other servers.

- An archive lock is created by placing a lock file into an appropriate directory. In order for the lock to persist, the timestamp on the file must change periodically; otherwise any contending Content Server will treat the lock as an expired lock.
- There is no default setting.

> **Note:** This variable is obsolete as of the Content Server 7.0 release. The functionality is now available in the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
ArchiverTraceLockMaintenance=true
```

See Also

- "ArchiverDoLocks" on page 8-8
- "ArchiverTraceLocks" on page 8-11
- "ArchiverTouchMonitorInterval" on page 8-9

ArchiverTraceLocks

Dumps trace information to the console when locks on an archive are acquired and released.

There is no default setting.

> **Note:** This variable is obsolete as of the Content Server 7.0 release. The functionality is now available in the System Audit Information page.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
ArchiverTraceLocks=true
```

See Also

- "ArchiverDoLocks" on page 8-8
- "ArchiverTraceLockMaintenance" on page 8-10
- "ArchiverTouchMonitorInterval" on page 8-9

AuthorAddress

Specifies the e-mail address of the author of a content item.

Type and Usage

- "Value Variables" on page 3-21
- "Users" on page 4-20

Output

Returns a string or Boolean value depending on use.

- Standard use: Returns the e-mail address of the content item's author as a string.
- Used in a conditional statement:
  - Returns TRUE if the content item author has a defined e-mail address.
  - Returns FALSE if the content item author has no e-mail address.

Example

Can be used to alert the content item author through e-mail when a revision is made.

```
<$AuthorAddress$>
```

BatchLoaderPath

Defines the path to the default batch load text file.

Returns the file path as a string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Batch Loader" on page 4-2

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

- As a Windows configuration entry:

  ```
  BatchLoaderPath=c:/domain/BatchLoader/batchfile.txt
  ```

- As a Solaris/UNIX configuration entry:

  ```
  BatchLoaderPath=/u1/intradoc3/batLd/batchfile
  ```

- As script, returns the file path as a string:

  ```
  <$BatchLoaderPath$>
  ```

BatchLoaderUserName

Defines the authorized user name for the Batch Loader utility. The user name must belong to the *admin* role.

- Default is *sysadmin*.
- Returns the Batch Loader user name as a string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Batch Loader" on page 4-2

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
BatchLoaderUserName=sysadmin
```

BinDir

Defines the path to the Content Server launch executable files, such as IdcServer.exe.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
BinDir=c:/domain/bin/
```

break

Often used to terminate a loop.

- The break instruction causes the innermost loop to be exited.
- Control resumes with the first statement following the end of the loop.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

BrowserVersionNumber

Retrieves the version number of the client browser.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Clients" on page 4-2

Output

Returns the browser version number as a string.

Example

Can be used to ensure that the user has a browser version compatible with Content Server.

```
<$BrowserVersionNumber$>
```

c

Specifies a comment in Idoc Script code.

Type and Usage

- "Idoc Script Comments" on page 2-2
- "Idoc Script" on page 4-12

Example

Places a comment in the code:

```
<$c = "Sets the variables to empty strings."$>
```

cacheInclude

This function acts similar to the "inc" IdocScript function. It will evaluate the dynamic html include corresponding to 'includeName', and display it on the page. The difference is that if possible, it will pull the rendered html from a cache, instead of evaluating it again.

See "Keywords Versus Functions" on page 2-12 for more information.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

Takes three required parameters and two optional parameters. For example:

```
cacheInclude(includeName, scope, lifeSpan [, cacheName, key])
```

Parameter	Description
includeName	The name of the dynamic html element to evaluate.
scope	Set to 'session' if each user should see different html, or 'application' if all users will see the same thing.
lifeSpan	The lifespan of this include in the cache, in seconds.
cacheName	Optional: if you want to place this data into a named cache instead of the default cache. If an empty string is passed, it will cache the include into the default cache for the session.
key	Optional: if you do not want automatic name-scoping of your cache to prevent conflicts, you can specify a unique key here.

Example

This will cache the "std_page_begin" include for each user for ten minutes. This is about 10k per user in the cache.

```
<$cacheInclude("std_page_begin", "session", 600)$>
```

See Also

- "inc" on page 8-148
- "forceExpire" on page 8-94
- "setExpires" on page 8-287
- "setHttpHeader" on page 8-288
- "setMaxAge" on page 8-289

captionEntryWidth

Specifies the width of a metadata field, in percent.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the width of the current metadata field in percent.

Example

Used as script:

```
<$if isInfo$>
    <$captionFieldWidth="30%"$>
    <$captionEntryWidth="70%"$>
<$elseif isEditMode$>
    <$captionFieldWidth="20%"$>
    <$captionEntryWidth="80%"$>
<$endif$>
```

See Also

- "captionFieldWidth" on page 8-21

captionFieldWidth

Specifies the width of a metadata field caption, in percent.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the width of the current metadata field caption in percent.

Example

Used as script:

```
<$if isInfo$>
    <$captionFieldWidth="30%"$>
    <$captionEntryWidth="70%"$>
<$elseif isEditMode$>
    <$captionFieldWidth="20%"$>
    <$captionEntryWidth="80%"$>
<$endif$>
```

See Also

- "fieldCaptionStyle" on page 8-78
- "captionEntryWidth" on page 8-20

ChunkedRequestTrace

Outputs chunked file transfer requests on the Content Server Output page in the Admin Server.

- When set to TRUE, chunking requests are output to the Content Server Output page.
- When set to FALSE, chunking requests are not output.
- For this setting to output data, the Chunking function must be enabled using DisableHttpUploadChunking.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
ChunkedRequestTrace=true
```

See Also

- "AppletChunkSize" on page 6-14
- "AppletChunkThreshold" on page 6-15
- "DisableHttpUploadChunking" on page 6-36

CleanUp

Determines whether files should be deleted from the file system as they are being batch loaded.

- When set to TRUE, deletes each batch loaded file from the hard drive after it is successfully checked in or updated.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Batch Loader" on page 4-2

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
CleanUp=true
```

ClientControlled

Checks if the page was accessed from the ODMA Client.

This value is passed by the string for controlling the update process that is provided by the client.

Type and Usage

- "Setable Variables" on page 3-19
- "Clients" on page 4-2

Output

- Returns TRUE if the page was accessed from the ODMA Client.
- Returns FALSE if the page was not accessed from the ODMA Client.

Example

Checks if the page was accessed from the ODMA Client:

```
<$ClientControlled$>
```

See Also

- "HasLocalCopy" on page 8-119
- "IsNotLatestRev" on page 6-121

CollectionLocation

Defines where the default Archive collection is located.

Default is *DomainHome*/ucm/cs/archives.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2
- "Directories and Paths" on page 4-9

Location

- *IntradocDir*/config/config.cfg
- *DomainHome*/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
CollectionLocation=c:/domain/new/archives/
```

ConfigDir

Defines the path to the *config* directory.

Default is *IntradocDir*/config.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

As a configuration setting:

```
ConfigDir=c:/domain/config/
```

coreContentOnly

Set this variable in the URL of a service request to display only the form. The `std_page_begin` and `std_page_end` include files will not be displayed.

Type and Usage

- "Setable Variables" on page 3-19
- "Page Display" on page 4-14

Output

None.

CURRENT_DATE

Returns the current date and time.

This variable is similar to dateCurrent, which is used more frequently.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Date and Time" on page 4-8

Output

Returns a string formatted according to the user locale.

Example

Used as script:

```
<$CURRENT_DATE$>
```

See Also

- "dateCurrent" on page 8-31

CURRENT_ROW

Evaluates which row of a ResultSet you are in.

The first row in a ResultSet is row zero (0).

Type and Usage

- "Dynamic Variables" on page 3-10
- "ResultSets" on page 4-15

Output

Returns the row number.

Example

Used as script:

```
<$CURRENT_ROW$>
```

DataDir

Defines the path to the *data* directory.

Default is *IntradocDir*/data.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
DataDir=c:/domain/data/
```

dateCurrent

Returns the current date and time.

Can be used to return the current date and time to the user or to create commands using date evaluations.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only optional parameter is an integer, which adjusts the date by the specified number of days relative to the current date.

Output

Returns a date formatted according to the user locale.

Example

In the following examples, dates are formatted according to the default English-US locale:

```
m/d/yy h:mm XM
```

The following returns the current date and the current time (for example, 8/12/01 1:55 PM):

```
<$dateCurrent()$>
```

Returns the date ten days in the future and the current time (for example, 8/22/01 1:55 PM):

```
<$dateCurrent(10)$>
```

Returns the date ten days in the past and the current time (for example, 8/2/01 1:55 PM):

```
<$dateCurrent(-10)$>
```

See Also

- "CURRENT_DATE" on page 8-28

DateOutputFormat

Sets the date output format for the Indexer.

- There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Date and Time" on page 4-8
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DateOutputFormat=M/d/yyyy hh:mm:ss a
```

ddAppendIndexedColumnResultSet

This function loads a dynamicdata table into a result set. It is very similar to ddLoadIndexedColumnResultSet. The main difference is that if the Idoc Script result set already exists, the new result set created from the dynamicdata table is appended to it. Any fields found in the dynamicdata table, but not in the target result set, are automatically added.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for this function.

Parameters	Description
`dataTableName`	The name of the dynamicdata table to load.
`idocTableName`	The name of the result set into which the dynamicdata table should be appended.
`indexColumn`	The name of an indexed column in the dynamicdata table.
`indexValue`	The value to use to select a section of the original table. This value will be checked in a case-insensitive manner against the indexColumn in each row. If it matches, that row is loaded into the result set; otherwise it is skipped.
`mappingTableName`	The name of a dynamicdata table used to rename the columns of the final result set. The renaming is done by mapping the column names in the specified mapping table to the values of the first row in the same table. This is done exactly the same in ddLoadResultSet.

Output

If the dynamicdata table is successfully loaded, it returns True.

Example

```
<@dynamicdata MyDataTable@>
<?commatable indexedColumns="fieldA"?>
fieldA, fieldB, fieldC
1,      2,       3
4,      5,       6
1,      3,       9
<@end@>


<$exec rsCreateResultSet("MyResultSet", "fieldA,fieldB")$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.fieldA = "first value"$>
<$MyResultSet.fieldB = "second value"$>
<$exec ddLoadIndexedColumnResultSet("MyDataTable", "MyResultSet", "fieldA", "1")$>
<$exec rsFirst("MyResultSet")$>
<$foo = MyResultSet.fieldB$>    [[% (foo == 'second value') %]]
<$exec rsNext("MyResultSet")$>
<$bar = MyResultSet.fieldC$>    [[% (bar == '3') %]]
```

See Also

- "ddLoadIndexedColumnResultSet" on page 8-40

ddAppendResultSet

This function loads a dynamicdata table into a result set. It is very similar to ddLoadResultSet. The main difference is that if the Idoc Script result set already exists, the new result set created from the dynamicdata table is appended to it. Any fields found in the dynamicdata table, but not in the target result set, are automatically added.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
`dataTableName`	The name of the dynamicdata table to load.
`idocTableName`	The name of the result set into which the dynamicdata table should be appended.
`mappingTableName`	The name of a dynamicdata table used to rename the columns of the final result set. The renaming is done by mapping the column names in the specified mapping table to the values of the first row in the same table. This is done exactly the same in ddLoadResultSet.

Output

If the dynamicdata table is successfully loaded, it returns True.

Example

```
<@dynamicdata MyDataTable@>
field1, field2, field3
a,      b,       c
d,      e,       f
<@end@>


<$exec rsCreateResultSet("MyResultSet", "field1,field2")$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.field1 = "first value"$>
<$MyResultSet.field2 = "second value"$>
<$exec ddAppendResultSet("MyDataTable", "MyResultSet")$>
<$exec rsFirst("MyResultSet")$>
<$foo = MyResultSet.field1$>    [[% (foo == 'first value') %]]
<$exec rsNext("MyResultSet")$>
<$bar = MyResultSet.field3$>    [[% (bar == 'c') %]]
```

See Also

- "ddLoadResultSet" on page 8-41
- "ddAppendIndexedColumnResultSet" on page 8-33

ddApplyTableSortToResultSet

This function sorts an existing result set using the rules defined in a particular dynamicdata table.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
dataTableName	The name of the dynamicdata table to use for its sorting rules.
resultSetName	The name of the result set into which the dynamicdata table should be appended.

Output

If the sorting is successfully performed, it returns True.

Example

```
<@dynamicdata MyDataTable@>
<?commatable sortcolumn="order" sortType="int" sortOrder="asc"?>
user,    service,            order
jane,    DOC_INFO,           10
bob,     GET_SEARCH_RESULTS, 20
annette, CHECKIN_NEW,        30
<@end@>


<$exec rsCreate("MyResultSet")$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.user = "james"$>
<$MyResultSet.service = "GET_FILE"$>
<$MyResultSet.order = 75$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.user = "zoe"$>
<$MyResultSet.service = "DOC_INFO_BY_NAME"$>
<$MyResultSet.order = 20$>
<$exec ddApplyTableSortToResultSet("MyDataTable", "MyResultSet")$>
```

ddGetFieldList

This function takes a dynamicdata table and returns a comma-separated string containing the names of the columns in the table. It is expected that many dynamicdata tables consist only of field names without any rows just to supply comma-separated lists of values to code in the Content Server.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

This function has one parameter. `dataTableName` is the name of the dynamicdata table to load.

Output

If the dynamictable exists, it returns True.

Example

```
<@dynamicdata MyDataTable@>
foo,bar,baz
<@end@>


<$fieldList = ddGetFieldList("MyDataTable")$>
<$trace(fieldList, "#console")$> [[% Outputs: foo,bar,baz %]]
```

ddIncludePreserveValues

This function executes a resource include, but protects values specified by a dynamicdata table from being changed. The column names in the data table are used as the list of variables names that must be protected. These variables are protected by temporarily caching them, calling the include, and then resetting those variables back to the cached values. If a variable was null, it is set to blank.

If one of the column names in the table starts with a dollar symbol ($), then the string that follows is assumed to be the name of a result set. In that case, it is the pointer to the result set that is temporarily cached in memory and then replaced after the resource include has finished executing. If the result set did not exist at the time of caching, then any result set that exists with that key at the end are removed. If the result set is active at the time it has its pointer cached, new result sets of the same name can be created during the call of the include and the previously active result set will be recovered appropriately without disturbing the loop. The one side effect is that if a new result set is created that temporarily replaces the active result set, then the algorithm for variable substitution that retrieves values first from active result sets can find values for a variable from the cached active result set (assuming the variable is not found as a field in an active result set with higher precedence). The cached active result set maintains its place in the active result set stack,

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
`includeName`	The name of the resource include to execute.
`dataTableName`	The name of the dynamicdata table to use for preserving local data and result sets..

Output

A string representing the output of the executed resource include.

Example

```
<@dynamicdata MyPreservedFields@>
foo, bar, $baz
<@end@>


<@dynamichtml my_include@>
    <$foo = "tempValue1"$>
    <$bar = "tempValue2"$>
<@end@>


<$foo = 5$>
<$ddIncludePreserveValues("my_include", "MyPreservedFields")$>
```

```
<$trace(foo, "#console")$>   [[% Outputs: 5 %]]
```

ddLoadIndexedColumnResultSet

This function loads a dynamicdata table into a result set. This function is similar to ddLoadResultSet except that it only loads those rows whose values for indexColumn match indexValue. The comparisons made on this column are case-insensitive.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
`dataTableName`	The name of the dynamicdata table to load.
`idocTableName`	The name of the result set into which the dynamicdata table should be loaded.
`indexColumn`	The name of an indexed column in the dynamicdata table.
`indexValue`	The value to use to select a section of the original table. This value will be checked in a case-insensitive manner against the *indexColumn* in each row. If it matches, that row is loaded into the result set; otherwise it is skipped.
`mappingTableName`	The name of a dynamicdata table used to rename the columns of the final result set. The renaming is done by mapping the column names in the specified mapping table to the values of the first row in the same table. This is done exactly the same in ddLoadResultSet.

Output

If the dynamicdata table is successfully loaded, it returns True.

Example

```
<@dynamicdata MyDataTable@>
<?commatable indexedColumns="fieldA"?>
fieldA, fieldB, fieldC
1,      2,       3
4,      5,       6
1,      3,       9
<@end@>


<$exec ddLoadIndexedColumnResultSet("MyDataTable", "MyResultSet", "fieldA", "1")$>
<$exec rsFirst("MyResultSet")$>
<$foo = MyResultSet.fieldB$>    [[% (foo == '2') %]]
<$exec rsNext("MyResultSet")$>
<$bar = MyResultSet.fieldC$>    [[% (bar == '9') %]]
```

See Also

- "ddLoadResultSet" on page 8-41

ddLoadResultSet

This function loads a dynamicdata table into a result set.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
`dataTableName`	The name of the dynamicdata table to load.
`idocTableName`	The name of the result set into which the dynamicdata table should be loaded.
`mappingTableName`	The name of a dynamicdata table used to rename the columns of the final result set. The renaming is done by mapping the column names in the specified mapping table to the values of the first row in the same table. This is done exactly the same in ddLoadResultSet.

Output

If the dynamicdata table is successfully loaded, it returns True.

Example

```
<@dynamicdata MyDataTable@>
fieldA, fieldB, fieldC
1,      2,       3
4,      5,       6
1,      3,       9
<@end@>


<$exec ddLoadResultSet("MyDataTable", "MyResultSet")$>
<$exec ddLoadResultSet("MyDataTable", "MyAliasedResultSet")$>
<$exec rsFirst("MyResultSet")$>
<$exec rsFirst("MyAliasedResultSet")$>
<$foo = MyResultSet.field1$>        [[% (foo == 'a') %]]
<$bar = MyAliasedResultSet.alias2$>  [[% (bar == 'c') %]]
```

See Also

- "ddLoadIndexedColumnResultSet" on page 8-40

ddMergeIndexedColumnResultSet

This function merges a dynamicdata table into a result set. This function is similar to ddMergeResultSet except that it only merges those rows whose values for indexColumn match indexValue. The comparisons made on this column are case-insensitive.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
`dataTableName`	The name of the dynamicdata table to load.
`idocTableName`	The name of the result set into which the dynamicdata table should be merged.
`indexColumn`	The name of an indexed column in the dynamicdata table.
`indexValue`	The value to use to select a section of the original table. This value is checked in a case-insensitive manner against the indexColumn in each row. If it matches, that row is loaded into the result set; otherwise it is skipped.
`mergeType`	The type of merge to perform. Set this to `replace` to prevent the merge from appending any new rows. Any other value for this parameter is treated as `append`.
`mappingTableName`	The name of a dynamicdata table which should be used to rename the columns of the final result set. This renaming is done by mapping the column names in this mapping table to the values of the first row in this same table. This is done exactly as it is done in ddLoadResultSet.

Output

If the dynamicdata table is successfully merged, it returns True.

Example

```
<@dynamicdata MyDataTable@>
fieldA, fieldB, fieldC
1,      2,       3
4,      5,       6
1,      3,       9
<@end@>


<$exec rsCreateResultSet("MyResultSet", "fieldA,fieldB,fieldC")$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.fieldA = "4"$>
<$MyResultSet.fieldB = "8"$>
<$MyResultSet.fieldB = "23"$>
<$exec rsAppendNewRow("MyResultSet")$>
```

```
<$MyResultSet.fieldA = "8"$>
<$MyResultSet.fieldB = "21"$>
<$MyResultSet.fieldB = "59"$>
<$exec ddMergeIndexedColumnResultSet("MyDataTable", "MyResultSet", "fieldA", "4",
"replace")$>
<$exec rsFirst("MyResultSet")$>
<$foo = MyResultSet.fieldB$>    [[% (foo == '5') %]]
<$exec rsNext("MyResultSet")$>
<$bar = MyResultSet.fieldC$>    [[% (bar == '59') %]]
```

See Also

- "ddMergeResultSet" on page 8-44

ddMergeResultSet

This function merges a dynamicdata table into an existing result set. If the target result set does not exist, then this function acts exactly like ddLoadResultSet and the target result set is simply created. Otherwise, the merge is performed using the first column of the new result set as the merge key. By default, the merge appends any rows from the new result set that do not match any rows in the existing Idoc Script table, unless the mergeType parameter is set to replace, in which case no new rows are added to the Idoc Script result set.

Note that this merge does not replace all rows matched by a particular row of the new result set, just the first one that it finds. The matching is case-sensitive. The mergeType parameter is optional and defaults to null.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
`dataTableName`	The name of the dynamicdata table to load.
`idocTableName`	The name of the result set into which the dynamicdata table should be merged.
`mergeType`	The type of merge to perform. Set this to `replace` to prevent the merge from appending any new rows. Any other value for this parameter is treated as `append`.
`mappingTableName`	The name of a dynamicdata table which should be used to rename the columns of the final result set. This renaming is done by mapping the column names in this mapping table to the values of the first row in this same table. This is done exactly as it is done in ddLoadResultSet.

Output

If the dynamicdata table is successfully merged, it returns True.

Example

```
<@dynamicdata MyDataTable@>
fieldA, fieldB, fieldC
1,      2,       3
4,      5,       6
<@end@>


<$exec rsCreateResultSet("MyResultSet", "fieldA,fieldB,fieldC")$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.fieldA = "4"$>
<$MyResultSet.fieldB = "52"$>
<$MyResultSet.fieldC = "18"$>
<$exec ddMergeResultSet("MyDataTable", "MyResultSet", "replace")$>
```

```
<$exec rsFirst("MyResultSet")$>
<$foo = MyResultSet.fieldB$>    [[% (foo == 52) %]]
<$bar = MyResultSet.#numRows$>  [[% (bar == 1) %]]
```

See Also

- "ddLoadResultSet" on page 8-41

ddMergeUsingIndexedKey

This function pulls selective rows from a dynamicdata table and use them to replace particular row values in a target result set. The values of a particular column in the target result set are used as values for an index look up into the dynamicdata table. The sub table retrieved is used to replace column values in the target result set that have matching column names. The dynamicdata table is assumed to have only one row in the sub table selected out by the index value. If it has more than one row, only the first row is used to merge in values into the target result set.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
`dataTableName`	The name of the dynamicdata table to load.
`idocTableName`	The name of the result set into which the dynamicdata table should be merged.
`indexColumn`	The name of an indexed column in the dynamicdata table and the name of the column in the target result set from which values are retrieved to select out sub tables in the dynamicdata table.
`mappingTableName`	The name of a dynamicdata table which should be used to rename the columns of the final result set. This renaming is done by mapping the column names in this mapping table to the values of the first row in this same table. This is done exactly as it is done in ddLoadResultSet.

Output

If the dynamicdata table is successfully merged, it returns True.

Example

```
<@dynamicdata MyDataTable@>
<?commatable indexedColumns="fieldA"?>
fieldA, fieldB, fieldC
1,      2,       3
4,      5,       6
7,      8,       9
<@end@>


<$exec rsCreateResultSet("MyResultSet", "fieldA,fieldB,fieldC")$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.fieldA = "4"$>
<$MyResultSet.fieldB = "8"$>
<$MyResultSet.fieldB = "23"$>
 <$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.fieldA = "7"$>
<$MyResultSet.fieldB = "20"$>
```

```
<$MyResultSet.fieldB = "41"$>
<$exec rsAppendNewRow("MyResultSet")$>
<$MyResultSet.fieldA = "8"$>
<$MyResultSet.fieldB = "21"$>
<$MyResultSet.fieldB = "59"$>
<$exec ddMergeIndexedColumnResultSet("MyDataTable", "MyResultSet", "fieldA")$>
<$exec rsFirst("MyResultSet")$>
<$foo = MyResultSet.fieldB$>    [[% (foo == '5') %]]
<$exec rsNext("MyResultSet")$>
<$bar = MyResultSet.fieldC$>    [[% (bar == '9') %]]
<$exec rsNext("MyResultSet")$>
<$bar = MyResultSet.fieldC$>    [[% (bar == '59') %]]
```

ddSetLocal

This function takes a dynamicdata table and iterates over it, setting local data values for each row. The first column of the table is used as the list of keys, while the second column is used as the list of values.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

This function has one parameter, `dataTableName`, which is the name of the dynamicdata table to load.

Output

If the dynamicdata table exists, it returns True.

Example

```
<@dynamicdata MyDataTable@>
key, value
foo, 15
bar, 23
baz, 77
<@end@>


<$exec ddSetLocal("MyDataTable")$>
<$tmp1 = #local.foo$>   [[% (tmp1 == '15') %]]
<$tmp2 = #local.bar$>   [[% (tmp2 == '23') %]]
<$tmp3 = #local.baz$>   [[% (tmp3 == '77') %]]
```

See Also

- "ddSetLocalByColumnsFromFirstRow" on page 8-49
- "ddSetLocalByColumnsFromFirstRowIndexed" on page 8-50

ddSetLocalByColumnsFromFirstRow

This function takes the first row of a dynamicdata table and, using the column names as keys, sets a local value for each column and its corresponding value. If there is no first row, then this function does nothing. This can be a useful method for quickly setting a lot of local values. The `filterInclude` and `includeColumns` properties of the dynamicdata table are ignored.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

This function has one parameter, `dataTableName`, which is the name of the dynamicdata table to load.

Output

If the dynamicdata table was successfully loaded, it returns True.

Example

```
<@dynamicdata MyDataTable@>
fieldA, fieldB, fieldC
foo,    bar,    baz
<@end@>


<$exec ddSetLocalByColumnsFromFirstRow("MyDataTable")$>
<$tmpStr1 = #local.fieldA$>   [[% (tmpStr1 == 'foo') %]]
<$tmpStr2 = #local.fieldC$>   [[% (tmpStr2 == 'baz') %]]
```

See Also

- "ddSetLocal" on page 8-48
- "ddSetLocalByColumnsFromFirstRowIndexed" on page 8-50

ddSetLocalByColumnsFromFirstRowIndexed

This function takes the first row of a dynamicdata table, and, using the column names as keys, sets a local value for each column and its corresponding value. If there is no first row, then this function does nothing. This function is almost identical to ddSetLocalByColumnsFromFirstRow. The only difference is that this function uses the first row given an indexed column and value instead of the very first row of the dynamicdata table. If no indexed row is found then this function does nothing.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

The following table lists parameters for the function.

Parameters	Description
dataTableName	The name of the dynamicdata table to load.
indexColumn	The name of an indexed column in the dynamicdata table.
indexValue	The value to use to select a row of the original table. This value will be checked in a case-insensitive manner against the indexColumn in each row. If it matches, that row will be used and the rest of the table will be ignored.

Output

If the dynamicdata table is successfully loaded, it returns True.

Example

```
<@dynamicdata MyDataTable@>
<?commatable indexedColumns="fieldB"?>
fieldA, fieldB, fieldC
1,      2,      3
4,      5,      6
7,      8,      9
3,      5,      2
<@end@>


<$exec ddSetLocalByColumnsFromFirstRowIndexed("MyDataTable", "fieldB", 5)$>
<$tmpStr1 = #local.fieldA$>   [[% (tmpStr1 == '4') %]]
<$tmpStr2 = #local.fieldC$>   [[% (tmpStr2 == '6') %]]
```

See Also

- "ddSetLocal" on page 8-48
- "ddSetLocalByColumnsFromFirstRow" on page 8-49

ddSetLocalEmpty

This function takes a dynamicdata table and iterates over it, clearing local data. The first column is used as the keys to clear.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

This function has one parameter, `dataTableName`, which is the name of the dynamicdata table to use.

Output

If the dynamicdata table exists, it returns True.

Example

```
<@dynamicdata MyDataTable@>
key
foo
bar
baz
<@end@>


<$foo = 1$>
<$bar = 2$>
<$baz = 3$>
<$exec ddSetLocalEmpty("MyDataTable")$>
<$if foo or bar or baz$>
    [[% This will not be executed as foo, bar, and baz are all empty. %]]
<$endif$>
```

See Also

- "ddSetLocal" on page 8-48

ddSetLocalEmptyByColumns

This function takes a dynamicdata table and uses the columns to clear values in local data.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

This function has one parameter, `dataTableName`, which is the name of the dynamicdata table to use.

Output

If the dynamicdata table exists, it returns True.

Example

```
<@dynamicdata MyDataTable@>
foo, bar, baz
<@end@>


<$bar = "asdf"$>
<$exec ddSetLocalEmptyByColumns("MyDataTable")$>
<$if bar$>
    [[% This will not execute as bar will be empty. %]]
<$endif$>
```

See Also

- "ddSetLocal" on page 8-48
- "ddSetLocalEmpty" on page 8-51

defaultFieldInclude

Specifies the include to use to display the metadata field.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Example

Sets the default field display include for a hidden field on a query page to *std_query_hidden_field*:

```
<$if isFieldHidden$>
    <$if isQuery and not (fieldType like "Date") and not (fieldType like "Int")$>
        <$defaultFieldInclude = "std_query_hidden_field"$>
    <$endif$>
```

See Also

- "fieldCaptionInclude" on page 8-77
- "fieldEntryInclude" on page 8-81

defaultOptionListScript

Defines a piece of Idoc Script that displays a standard option list field.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

None.

Example

Generates an option list using the optList function:

```
<$if optionsAllowPreselect and fieldValue$>
    <$defaultOptionListScript = "<$optList " & optionListName & ":fieldValue$>"$>
<$else$>
    <$defaultOptionListScript = "<$optList " & optionListName & ":noselected$>"$>
<$endif$>
```

Loops on the current option list ResultSet to generate an option list:

```
<@dynamichtml compute_resultset_option_list_script@>
<$if not optionListKey$>
    <$optionListKey = fieldName$>
<$endif$>
<$defaultOptionListScript = "<$loop " & optionListResultSet & "$>" & "<$inc('std_
resultset_option_list_item')$>" & "<$endloop$>"$>
<@end@>
```

See Also

- "optionListScript" on page 8-225
- "optList" on page 8-228

DelimitedUserRoles

Retrieves a comma-delimited, colon-delimited list of roles the current user belongs to.

Type and Usage

- "Configuration Variables" on page 3-2
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

Returns the user role list as a string.

Example

Returns a list of roles formatted as follows:

```
:guest:,:PublicContributor:,:ClassifiedConsumer:
```

See Also

- "UserRoles" on page 8-351

DirectoryLockingLogPath

Enables logging and defines the path for the log file during temporary locking of directories.

- Any errors that occur during temporary locking of directories will be reported to this log file.
- If this setting has no value, logging is disabled.
- There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DirectoryLockingLogPath=C:/temp/locking.log
```

DisableContentProfiles

Used to disable Content Profile functionality but does not remove profiles which have already been created. Profiles remain available through the user interface.

This configuration variable can be used to help determine if specific behavior is caused by Content Profiles or by another component.

After debugging has been completed, remove this variable from the config.cfg file and restart the Content Server to use Content Profiles again.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Profiles" on page 4-3

Location

IntradocDir/config/config.cfg

Example

DisableContentProfiles=true

DisableGzipCompression

Disables gzip compression of HTML pages and log files. Pages are compressed only if the client browser supports compressed HTML. Compression makes pages an average of 80% smaller and is supported by any HTTP 1.1 compatible browser.

- When set to TRUE, pages will not be compressed before being sent to a client Web browser.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Conversion" on page 4-5

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DisableGzipCompression=true
```

See Also

- "HTTP_ACCEPT_ENCODING" on page 6-74

DisableSharedCacheChecking

Disables timestamp checking on all loaded resources.

- When set to TRUE, resource files defined in components will be refreshed only when the Content Server is restarted.
- This setting does not stop timestamp checking on HCST and HCSP pages.
- This setting can be used to limit file system activity, typically when the Content Server file system is on a network share. This setting is particularly important in production environments where file checking and timestamp overhead can seriously degrade performance. In a development environment this setting can be optional.
- See the clustering documentation for more information.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DisableSharedCacheChecking=true
```

docLoadResourceIncludes

Loads all the includes in a specified content item for use in the display of the current page.

- The content item specified must have the file extension *idoc*.
- This function sets *StatusCode* as a side effect. Use the abortToErrorPage function if the specified file must successfully load for the page to correctly display.

Type and Usage

- "Global Functions" on page 3-11
- "Page Display" on page 4-14
- "Idoc Script" on page 4-12

Parameters

Takes a CGI-encoded parameter list that specifies a content item that is checked into the Content Server. The parameter options are listed in the following table.

Optional Parameters	Description
dID	If *dID* is not present, *dDocName* and *RevisionSelectionMethod* must be present. A rendition of the revision of the content item with this ID will be returned, if it exists, and the *RevisionSelectionMethod* parameter does not exist or has the value *Specific*.
dDocName	It is recommended that *dDocName* be present in all requests for content items where the *dDocName* is known. Error messages assume that it is present, as do other features such as forms. ■ If *dDocName* is not present, *dID* must be present and *RevisionSelectionMethod* must not be present. ■ If *RevisionSelectionMethod* is present, a rendition of a revision of the content item with this name with be returned, if it exists. ■ If *RevisionSelectionMethod* is not present, *dDocName* may be used in error messages.
RevisionSelectionMethod	If present, *dDocName* must be present. The value of this variable is the method used to compute a *dID* from the specified *dDocName*. The value may be *Specific*, *Latest*, or *LatestReleased*. ■ **Specific:** The *dDocName* is ignored, *dID* is required and is used to get a specific revision. ■ **Latest:** The latest revision of the content item (including revisions in a workflow) is used to compute the *dID*. ■ **LatestReleased:** The latest released revision of the content item is used to compute the *dID*.
Rendition	■ If not present, *Rendition* defaults to *Primary*. This parameter specifies the rendition of the content item. ■ If the value is *Primary*, *Web,* or *Alternate*, the primary, web-viewable, or alternate rendition of the selected revision is returned.

Note: When used in .hcsp pages, the ampersand (&) character in the CGI-encoded parameter list must be changed to the &amp; character.

Output

None.

Example

Loads the resource includes in the primary vault rendition of the latest revision of *mydoc*.

```
<$docLoadResourceIncludes("dDocName=mydoc&RevisionSelectionMethod=Latest")$>
```

See Also

- "abortToErrorPage" on page 8-2

docRootFilename

Retrieves the file name of a file without the extension or directory path.

- This function is typically used to extract the Content ID (*dDocName*) part of a Content Server-controlled static URL.

Type and Usage

- "Global Functions" on page 3-11
- "Directories and Paths" on page 4-9

Parameters

Takes one parameter:

- The only parameter is a path and file name.

Output

Returns the file name as a string.

Example

Returns the value *mydoc*:

```
docRootFilename("/groups/public/documents/adacct/mydoc.pdf")
```

DownloadApplet

Enables the multiple file Download Applet.

- When set to TRUE, the Download Applet is enabled so that multiple files can be downloaded from a search results page.
- When set to FALSE, the Download Applet is disabled.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Clients" on page 4-2

Location

- System Properties, Options tab, Enable download applet
- Admin Server, General Configuration, Enable download applet
- *IntradocDir*/config/config.cfg

Example

As a configuration setting, enables the Download Applet:

```
DownloadApplet=true
```

As script, evaluates the condition of the Download Applet:

```
<$DownloadApplet$>
```

See Also

- "MultiUpload" on page 8-216
- "UploadApplet" on page 8-334

DownloadSuggestedName

Retrieves the default path and suggested name for a file being downloaded.

The suggested name is based on the original file name of the content item.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Items" on page 4-3
- "Directories and Paths" on page 4-9

Output

Returns the path and suggested name for the downloaded file as a string.

Example

Returns the path and suggested name for the downloaded file:

```
<$DownloadSuggestedName$>
```

dpGet

Function used to return the value set by the dpSet function.

Both dpGet and dpSet are used to control the logic of Content Profile rules that are to be displayed. Commonly, a user may create an activation condition that sets a value for a specified key. Even if the condition evaluates to false (that is, the rule fails to fire), the key is set. Another rule can now access the key and retrieve the value when evaluating its activation condition. By using these two functions, a user can create a dependency between rules. For example, a user may want a rule to be evaluated only when a preceding rule is evaluated.

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

This function takes one parameter:

- Key: The designation for the value to be returned.

Example

```
<$myValue = dpGet("myKey")$>
```

See Also

- "dpSet" on page 8-68

dpPromote

Function used to 'promote' values from a rule evaluation context into the request context. After the evaluation of all rules, the key and value specified by this function are pushed into the local data.

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

This function takes two parameters:

- The first parameter is the designation for the value to be returned.
- The second parameter is the value to be returned.

Example

This example changes the value for AutoNumberPrefix for one request to be the value for dDocType:

```
<$dpPromote("AutoNumberPrefix", dDocType)$>
```

This example causes the Title field to be hidden on checkin pages:

```
<$dpPromote("dDocTitle:isHidden", "1")$>
```

See Also

- "dpPromoteRs" on page 8-67

dpPromoteRs

Similar to dpPromote, this function allows a ResultSet that is generated in a profile rule to be promoted to the request context. The ResultSet is pushed into the databinder after the evaluation of all rules. The values are not evaluated during rule activation or manipulation but are available for page presentation.

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

This function can take two parameters:

- The first parameter is the name of the ResultSet to push into the data.
- Optional. The second parameter is an alternate name for the ResultSet.

See Also

- "dpPromote" on page 8-66

dpSet

Function used to set temporary values during the evaluation of activation conditions or rule values anywhere Idoc Script is evaluated.

The values are set into a working area and can be retrieved by the `dpGet` function call. The key and value that is set does not persist, but is globally available for the profile being evaluated. All key/value pairs can be accessed by the rules of the profile.

Both `dpGet` and `dpSet` are used to control the logic of Content Profile rules that are to be displayed. Commonly, a user may create an activation condition that sets a value for a specified key. Even if the condition evaluates to false (that is, the rule fails to fire), the key is set. Another rule can now access the key and retrieve the value when evaluating its activation condition. By using these two functions, a user can create a dependency between rules. For example, a user may want a rule to be evaluated only when a preceding rule is evaluated.

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

This function takes two parameters:

- Key: The designation for the value to be returned.
- Value: Value associated with the key.

Example

```
<$dpSet("myKey", "1")$>
```

See Also

- "dpGet" on page 8-65

EnableEnvironmentPackager

Enables the Environment Packager debugging function.

- This function enables an administrator to package all files in the *admin*, *config*, and *bin* directories into a Zip archive, along with a text file called *environment.txt*. This text file contains additional information about the Java Virtual Machine, and the operating system of the server. The Zip file, *server_environment.zip*, is placed in the root directory of the Content Server.
- When set to TRUE, the Environment Packager function is enabled.
- When set to FALSE, the Environment Packager function is disabled.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
EnableEnvironmentPackager=true
```

EnableErrorFile

Generates a Batch Loader error file.

- If errors are encountered during a batch load, an error file is generated with those problem records added.
- When set to TRUE, an error file is generated.
- There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Batch Loader" on page 4-2
- "Debugging" on page 4-9

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
EnableErrorFile=true
```

eval

Evaluates a variable definition as if it were Idoc Script. Can be used to recursively evaluate a literal string.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is a variable that defines a string to be evaluated as Idoc Script.

Output

Returns the parameter string, with any Idoc Script in the variable definition resolved.

Example

Variable one is assigned the string *CompanyName*, and variable two is assigned a string that includes variable one within Idoc Script delimiters. On a page, variable one presents the string *CompanyName*, variable two presents the string *Welcome to <$one$>* and eval(two) presents the string `Welcome to CompanyName`.

```
<$one="CompanyName"$>
<$two="Welcome to <$one$>"$>
<$one$><br>
<$two$><br>
<$eval(two)$>
```

Say you wanted to exclude all metadata fields that have the prefix *xPrefix* in their names. You could use the *eval* function to dynamically write and evaluate Idoc Script for this purpose:

```
<$loop DocMetaDefinition$>
<$if strIndexOf("xPrefix", dName) >= 0$>
    <$myScript = "<$" & dName & ":isExcluded=1$>"$>
    <$eval(myScript)$>
<$endif$>
<$endloop$>
```

See Also

- "Keywords Versus Functions" on page 2-12
- "exec" on page 8-72
- "setResourceInclude" on page 8-290

exec

Executes an Idoc Script expression and suppresses the output (does not display the expression on the page).

- In earlier versions of Idoc Script, the *exec* keyword was required to suppress the *value* of any variable from appearing in the output file. In the current version, the *exec* keyword is needed only to suppress an *expression* from appearing in the output.

  For example, the first line below is equivalent to the last two lines:

  ```
  <$varA="stringA", varB ="stringB"$>
  <$exec varA="stringA"$>
  <$exec varB="stringB"$>
  ```

- The *exec* keyword is typically used to evaluate behind-the-scenes code, such as specifying an include to be used later in the page.

Type and Usage

- "Special Keywords" on page 2-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is an Idoc Script expression.

Output

Returns the value of the expression, but does not display the expression on the page.

Example

Evaluates the *inc* global functions to incorporate the specified includes without displaying their code on the page:

```
<@dynamichtml std_definitions@>
    <$exec inc("std_page_variable_definitions")$>
    <$exec inc("define_image_files")$>
<@end@>
```

See Also

- "Keywords Versus Functions" on page 2-12
- "eval" on page 8-71

executeService

Executes a Content Server service.

- This function allows the specified service to be executed while the page is being constructed. Generally, services are executed using a tool such as IdcCommand or the CGI URL on the browser.
- Used with dynamic server pages.
- Services that can be called with the `executeService` function must be scriptable, meaning that they do not require parameter input. Scriptable services have an access level of 32 or more.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is the name of the service to be executed. The live data in the DataBinder is used as parameters to the specified service.

Output

- Returns a StatusCode of `0` if the service was executed successfully.
- Returns a StatusCode of `-1` if an error in the service execution occurred.
- All output of the executed service is suppressed, but any ResultSets and loaded values are available.

Example

Executes a service when given a service name:

```
<$executeService("servicename")$>
```

See Also

- "StatusCode" on page 6-249

ExternalUserAccounts

Retrieves the default roles for users who are defined by an external user base (NTLM, Active Directory, or LDAP).

Type and Usage

- "User Read-Only Variables" on page 3-19
- "External Security" on page 4-19
- "Users" on page 4-20

Output

- Returns a comma-delimited list of accounts, with permissions for each account specified in parentheses after the account name.
- The `#none` entry grants privileges to content items that have no account assigned. The `#all` entry grants privileges to all accounts.

Example

As script, returns the default account information as a string:

```
<$ExternalUserAccounts$>
```

See Also

- "ExternalUserRoles" on page 8-75
- "UserAccounts" on page 8-339
- "Default Accounts" on page 6-31

ExternalUserRoles

Retrieves the default roles for users who are defined by an external user base (NTLM, Active Directory, or LDAP).

Type and Usage

- "User Read-Only Variables" on page 3-19
- "External Security" on page 4-19
- "Users" on page 4-20

Output

- Returns the list of roles as a string.

Example

As script, returns the default role information as a string:

```
<$ExternalUserRoles$>
```

See Also

- "ExternalUserAccounts" on page 8-74
- "UserRoles" on page 8-351

fieldCaption

Specifies the caption label for a metadata field.

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the caption of the metadata field as a string.

Example

Defines the caption for the *dDocTitle* metadata field as the *wwTitle* localized string:

```
<$fieldName = "dDocTitle", fieldCaption = lc("wwTitle"), isRequired = 1, fieldType
= "BigText", requiredMsg = lc("wwSpecifyTitle")$>
```

fieldCaptionInclude

Specifies the include to use to display the caption for the metadata field.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

None.

Example

Sets the caption display include to *std_field_caption*:

```
<$fieldCaptionInclude = "std_field_caption"$>
```

See Also

- "defaultFieldInclude" on page 8-53
- "fieldEntryInclude" on page 8-81
- "fieldCaptionStyle" on page 8-78

fieldCaptionStyle

Specifies the style of the caption for the metadata field.

The following SPAN styles are typically used for captions. They are defined in the *std_style_declaration* include in the *IdcHomeDir*/resources/core/std_page.htm resource file:

- searchLabel
- infoLabel
- tableEntry
- requiredField

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the name of the caption style.

Example

Sets the caption style to *requiredField*:

```
<$if isRequired and not suppressCaptionHighlighting$>
    <$fieldCaptionStyle = "requiredField"$>
<$endif$>
Generates the standard field caption:
<@dynamichtml std_field_caption@>
    <span class=<$fieldCaptionStyle$>><$fieldCaption$><$if not
    isFormSubmit$><$":"$><$endif$></span>
<@end@>
```

See Also

- "fieldCaption" on page 8-76
- "fieldCaptionInclude" on page 8-77
- "fieldValueStyle" on page 8-90
- "NotationForRequiredFields" on page 8-220
- "StyleForRequiredFields" on page 8-316

fieldDefault

Specifies the default value for a metadata field.

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the default value of the metadata field as a string.

Example

Defines the default value for the *dDocTitle* metadata field:

```
<$fieldName = "dDocTitle", fieldCaption = lc("wwTitle"), isRequired = 1, fieldType
= "BigText", requiredMsg = lc("wwSpecifyTitle"), fieldDefault="Enter a Title"$>
```

fieldEditWidth

Specifies the character width of the metadata input field on the HTML page. It is set in `compute_namevalue_edit_widths`, which is included in the resource `compute_std_field_includes`.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

None.

fieldEntryInclude

Specifies the include to use to display the value for the metadata field.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Example

Sets the value display include to *std_checkbox_entry*:

```
<$fieldEntryInclude = "std_checkbox_entry"$>
```

See Also

- "defaultFieldInclude" on page 8-53
- "fieldCaptionInclude" on page 8-77
- "fieldValueStyle" on page 8-90

fieldExtraScriptInclude

Specifies the name of the include file containing additional JavaScript validation for a specific metadata field. This is set in the resource `compute_std_field_includes`.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

fieldInclude

Specifies the name of the include to display a metadata field. The include file is usually `std_namevalue_field`, but it can be set to another include.

This is usually set at the top of the page with `xFieldName:include` but can also be set by overriding the resource `compute_std_field_includes`.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

fieldIsOptionList

Specifies whether a metadata field has an option list.

- When set to TRUE, the field has an option list.
- When set to FALSE, the field does not have an option list.
- Default is FALSE.

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

- Returns TRUE if the field has an option list.
- Returns FALSE if the field does not have an option list.

Example

Defines the *dDocAccount* metadata field as an option list if predefined accounts exist:

```
<$if HasPredefinedAccounts$>
    <$fieldIsOptionList = 1, optionListName = "docAccounts",
    fieldOptionListType = "combo"$>
<$endif$>
```

See Also

- "hasOptionList" on page 8-120
- "fieldOptionListType" on page 8-87
- "optionListName" on page 8-223

fieldMaxLength

Specifies the maximum length of the metadata field. This is not the size of the field but the maximum number of characters that the field can contain. It is usually set at the top of the page with `xFieldName:include` but can also be set by overriding the resource `compute_std_field_includes`.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

fieldName

Specifies the internal name of a metadata field.

- Predefined metadata fields begin with a lowercase d (for example, *dDocName*).
- Custom metadata fields begin with a lowercase x (for example, *xComments*).

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the name of the metadata field as a string.

Example

Defines the *dDocTitle* metadata field:

```
<$fieldName = "dDocTitle", fieldCaption = lc("wwTitle"), isRequired = 1, fieldType
= "BigText", requiredMsg = lc("wwSpecifyTitle")$>
```

fieldOptionListType

Specifies the type of option list for a metadata field.

Possible values for this variable are:

fieldOptionListType Value	Corresponding Option in Configuration Manager
strict	Select List Validated or Select List Not Validated
combo	Edit and Select List
multi	Edit and Multiselect List
access	None (Used in Collaboration Server for member access lists)

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the option list type as a string.

Example

Defines the *dDocAccount* option list type as *combo* (Edit and Select List):

```
<$if HasPredefinedAccounts$>
    <$fieldIsOptionList = 1, optionListName = "docAccounts",
    fieldOptionListType = "combo"$>
<$endif$>
```

See Also

- "fieldIsOptionList" on page 8-84
- "optionListName" on page 8-223

fieldType

Specifies the type of metadata field.

Possible values for this variable are:

fieldType Value	Corresponding Option in Configuration Manager
Text	Text
BigText	Long Text
Int	Integer
Date	Date
Memo	Memo

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the metadata field type as a string.

Example

Defines the *dDocTitle* metadata field as a *BigText* (Long Text) field:

```
<$fieldName = "dDocTitle", fieldCaption = lc("wwTitle"), isRequired = 1,
    fieldType = "BigText", requiredMsg = lc("wwSpecifyTitle")$>
```

fieldValue

Specifies the value for a metadata field.

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the value of the current metadata field.

Example

Generates the standard field value:

```
<@dynamichtml std_value_label@>
    <span class="<$fieldValueStyle$>"><$fieldValue$></span><!--""-->
<@end@>
```

See Also

- "fieldEntryInclude" on page 8-81
- "fieldCaptionStyle" on page 8-78
- "fieldValueStyle" on page 8-90
- "optionListName" on page 8-223

fieldValueStyle

Specifies the style of the value for the metadata field.

The following SPAN styles are typically used for values. They are defined in the *std_style_declaration* include in the *IdcHomeDir*/resources/core/std_page.htm resource file:

- tableEntry
- xxsmall
- strongHighlight

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the name of the value style.

Example

Sets the value style:

```
<$if isFieldInfoOnly$>
    <$if valueStyle$>
<$fieldValueStyle = valueStyle$>
    <$else$>
<$fieldValueStyle = "tableEntry"$>
    <$endif$>
<$endif$>
```

Generates the standard field value:

```
<@dynamichtml std_value_label@>
    <span class="<$fieldValueStyle$>"><$fieldValue$></span><!--'"-->
<@end@>
```

See Also

- "fieldEntryInclude" on page 8-81
- "fieldCaptionStyle" on page 8-78
- "valueStyle" on page 8-357

fieldWidth

Specifies the width of a metadata field, in characters.

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the width of the current metadata field.

Example

Generates the *Extension* field with a width of 10 characters:

```
<$fieldName = "dExtension", fieldCaption = lc("wwNativeExtension"),
fieldWidth = 10$>
```

See Also

- "fieldCaptionStyle" on page 8-78

FileEncoding

Specifies the file encoding to be used.

If the Content Server is configured to use languages other than operating system's native language (for example, using both English and Japanese locales on an English system), this variable must be set to **UTF8** in both the Content Server and the Inbound Refinery.

Type and Usage

- "Configuration Variables" on page 3-2
- "Localization" on page 4-13
- "Content Server" on page 4-4
- "Inbound Refinery" on page 4-6

Location

- Content Server: *DomainHome*/ucm/cs/bin/intradoc.cfg
- Inbound Refinery: *Intradoc*/data/configuration/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
FileEncoding=UTF8
```

ForceDocTypeChoice

Adds a blank option as the default for the Type option list on checkin pages. This requires the user to select a Type.

- Used on pages as a display option.
- When set to TRUE, the checkin page displays the Type option list with a blank option.
- Default setting is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Field Display" on page 4-11

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ForceDocTypeChoice=true
```

forceExpire

This function can be used to force the cache for a particular include to expire. This is useful in the case of a change in the page display, such as adding new metadata fields, or if the user has saved a new query, or altered which links are viewable.

If the value for 'includeName' is null, or an empty string, then the entire cache will be eliminated. This is useful when all includes for a particular page are placed into the same cache, and need to be expired at the same time.

See Keywords Versus Functions for more information

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

Takes two required parameters and two optional parameters. For example:

```
forceExpire(includeName, scope [, cacheName, key])
```

Parameter	Description
includeName	The name of the dynamic html element to evaluate.
scope	Set to 'session' if each user should see different html, or 'application' if all users will see the same thing.
cacheName	Optional: if you want to place this data into a named cache instead of the default cache. If an empty string is passed, it will cache the include into the default cache for the session.
key	Optional: if you do not want automatic name-scoping of your cache to prevent conflicts, you can specify a unique key here.

Example

This can be used as a replacement for the 'std_page_begin' include. It will verify if the user executed one of the standard service calls to modify the links on the standard left navigation bar (saving a query, altering the portal design). If so, the cached html for the standard page begin will be invalidated. Then, the next 'cacheInclude' function will reevaluate the include, and place it back into the cache.

```
<$if strEquals(IdcService, "PNE_SAVE_QUERY") or strEquals(IdcService, "PNE_UPDATE_
PORTAL_INFO")$>
<$forceExpire("std_main_page_begin", "session", "", "std_main_page_begin")$>
<$endif$>
<$cacheInclude("std_main_page_begin", "session", 600, "", "std_main_page_begin")$>
```

See Also

- "inc" on page 8-148
- "cacheInclude" on page 8-19
- "setExpires" on page 8-287
- "setHttpHeader" on page 8-288

- "setMaxAge" on page 8-289

ForceSecurityGroupChoice

Adds a blank option as the default for the Security Group option list on checkin pages. This requires the user to select a Security Group.

Used on pages as a display option.

When set to TRUE, the checkin page displays the Security Group option list with a blank option.

Default setting is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Field Display" on page 4-11

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ForceSecurityGroupChoice=true
```

formatDate

Reformats a date/time to the default date/time format.

Database-formatted dates cannot be evaluated (for example, 2001-06-15).

Long-formatted dates cannot be evaluated (for example, June 15, 2001).

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is a string that specifies a date/time.

Output

- Returns the date/time in the format used by dateCurrent (for example, 6/15/01 1:55 PM).
- Returns null if the parameter cannot be evaluated.
- If a time is not provided, returns a default time of 12:00 AM.

Example

The following example formats the date and time and displays it as 12/14/99 2:00 PM:

```
<$formatDate("12/14/1999 02:00 PM")$>
```

The following example formats the date, assigns the default time, and displays it as 9/15/03 12:00 AM:

```
<$formatDate("09/15/2003")$>
```

The following script formats and displays a specified date and time. Line one evaluates an alternate date and time format and assigns it to a custom variable. Line two displays this date to a user.

```
<$my_customDateTime = formatDate("06/14/2004 15:05:34")$>
Final Approval: <$my_customDateTime$>
```

Output:

Final Approval: 6/14/04 3:05 PM

formatDateDisplay

Reformats a date/time to a date/time format for display to the user. Uses the "Display Date Format" in System Properties to format the date.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is a string that specifies a date/time.

Output

- Returns the date/time in the format used.
- Returns null if the parameter cannot be evaluated.

Example

```
<$formatDateDisplay$>
```

See Also

- "formatDateOnlyDisplay" on page 8-100

formatDateOnly

Reformats a date/time to the default date format and strips out the time.

Database-formatted dates cannot be evaluated (for example, 2000-02-02).

Long-formatted dates cannot be evaluated (for example, June 12, 2001).

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is a string that specifies a date/time.

Output

- Returns the date only in the format used by dateCurrent (for example, 7/12/00).
- Returns null if the parameter cannot be evaluated.

Example

Returns the current date only (for example, 9/12/01):

```
<$formatDateOnly(dateCurrent())$>
```

Returns the date ten days in the future (for example, 9/22/01):

```
<$formatDateOnly(dateCurrent(10))$>
```

Formats the date and time and displays the date only as 1/17/00:

```
<$formatDateOnly("01/17/2000 2:00 PM")$>
```

This script displays the current date and a date 100 days in the future. Line one assigns the current date only to a custom variable. Line two assigns a date 100 days in the future to a second custom variable. Line three displays these dates to a user (for example, Start Date: 10/12/01 and End Date: 1/20/02):

```
<$my_startDate = formatDateOnly(dateCurrent())$>
<$my_endDate = formatDateOnly(dateCurrent(100))$>
Start Date: <$my_startDate$> and End Date: <$my_endDate$>
```

formatDateOnlyDisplay

Reformats a date to a date format for display to the user. Uses the "Display Date Format" in System Properties to format the date.

Similar to formatDateDisplay but only formats the date.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is a string that specifies a date.

Output

- Returns the date in the format used.
- Returns null if the parameter cannot be evaluated.

Example

```
<$formatDateOnlyDisplay$>
```

See Also

- "formatDateDisplay" on page 8-98

formatDateOnlyFull

Reformats a date/time to the long date format and strips out the time. Database-formatted dates cannot be evaluated (for example, 2000-02-02).

This setting is deprecated for version 5.1 and later. Use formatDateWithPattern

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is a string that specifies a date/time.

Output

- Returns a long-formatted date:

  ```
  month d, yyyy
  ```

- Returns null if the parameter cannot be evaluated.

Example

Returns the current date in long format:

```
<$formatDateOnlyFull(dateCurrent())$>
```

Returns the date 365 days in the future in long format (for example, September 12, 2002):

```
<$formatDateOnlyFull(dateCurrent(365))$>
```

Formats the date only and displays as June 12, 2001:

```
<$formatDateOnlyFull("6/12/01 3:00 PM")$>
```

formatDateWithPattern

Reformats a date/time to a specified date/time pattern.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

Takes two parameters:

- The first parameter is a date string used by the Content Server, or a date object created with the parseDate or dateCurrent functions.
- The second parameter is the date/time pattern, such as MM/dd/yyyy.

  The capital letter `Z` denotes the use of a UTC time zone for the entry. The lowercase `zzzz` denotes the time offset (HHMM) from the UTC time, preceded by a plus (+) or minus (-) sign to indicate the offset.

Output

- Returns the date/time in the format specified by the pattern parameter.
- Returns null if the parameter cannot be evaluated.

Example

Displays Wed, 5 Jul 2006 12:08:56 -0700:

```
<$formatDateWithPattern(dateCurrent(),"EEE, d MMM yyyy HH:mm:ss zzzz")$>
```

Displays 2006-07-05 14:30:33Z:

```
<$formatDateWithPattern(dateCurrent(),"yyyy-MM-dd HH:mm:ssZ")$>
```

formatTimeOnly

Reformats a date/time to the default time format and strips out the date.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is a string or variable that specifies a date/time.

Output

- Returns a time only in the format used by dateCurrent (for example, 1:15 PM).
- Returns null if the parameter cannot be evaluated.

Example

Returns the current time only:

```
<$formatTimeOnly(dateCurrent())$>
```

Formats the time only and displays as 5:00 PM:

```
<$formatTimeOnly("2/2/99 5:00 PM")$>
```

Formats the time only and displays as 6:14 PM:

```
<$formatTimeOnly("04/21/2001 18:14:00")$>
```

formatTimeOnlyDisplay

Reformats a time to a time format for display to the user. Uses the "Display Date Format" in System Properties to format the time.

Similar to formatDateDisplay but only formats the time.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is a string that specifies the time.

Output

- Returns the time in the format used.
- Returns null if the parameter cannot be evaluated.

Example

```
<$formatTimeOnlyDisplay$>
```

See Also

- "formatDateDisplay" on page 8-98

generateUniqueId

This function returns the unique ID for an HTML page. This is used with the `id` attribute in an HTML element to guarantee a unique ID.

Type and Usage

- "Page Variables" on page 3-15
- "Page Display" on page 4-14

Parameters

The only parameter is the field name for which the ID will be generated.

Output

Returns the ID for an HTML page.

getDebugTrace

Retrieves the output of the debug trace.

Type and Usage

- "Global Functions" on page 3-11
- "Debugging" on page 4-9

Output

- Returns the output of the accumulated debug trace for the page being constructed.
- Returns an empty string if ScriptDebugTrace has not been set.

Example

Retrieves the output of the debug trace and outputs the information to a page:

```
<$getDebugTrace()$>
```

See Also

- "ScriptDebugTrace" on page 8-284

getErrorTrace

Retrieves the output of the error trace.

Error trace output is encoded for display in HTML pages. For example, the < and > delimiters are HTML-escaped and carriage returns are converted to <*br*> tags.

Type and Usage

- "Global Functions" on page 3-11
- "Debugging" on page 4-9

Output

- Returns the output of the accumulated error trace for the page being constructed.
- The function returns an empty string if ScriptErrorTrace has not been set.

Example

Retrieves the output of the error trace and outputs the information to a page:

```
<$getErrorTrace()$>
```

See Also

- "ScriptErrorTrace" on page 8-285

getFieldViewValue

This function is used when defining a content rule for a field. The value of a field for a content profile can be made dependent on a view by using this function. Because a field can have both a default and a derived value, the view can be used to create an interdependency between fields.

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

Takes three parameters:

- Field. The name of a metadata field. This field must have an associated view.
- Value. A lookup key for value.
- Column name. A column in the view's table.

Output

Returns the value in the column specified using the field value as a lookup.

Example

In the following example, the view table for the field `xEmployee` has a column called `type`. Using the value as specified in the `xEmployee` metadata field, this function provides a lookup into this table and returns the `type` column value.

```
getFieldViewValue("xEmployee", #active.xEmployee, "type")
```

getFreeMemory

This function returns the amount of free memory in the Java Virtual Machine (JVM). This is a performance auditing function used on the System Audit Information page.

Type and Usage

- "Global Functions" on page 3-11
- "Debugging" on page 4-9

Output

The amount of free JVM memory in megabytes.

Example

```
<$freeMem = getFreeMemory()$>
```

See Also

- "getTotalMemory" on page 8-114

getHelpPage

This function returns a relative URL to a help page based on the name of the page.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Page Display" on page 4-14

Parameters

The name of the page used.

Example

```
<input type=Button value="<$lc("wwQuickHelp")$>"
    onClick="QuickHelp('<$getHelpPage("QH_AdvancedSearch")$>', 'Search')">
```

getOptionListSize

This function returns the size of an option list. It is used to determine whether to display custom user metadata fields on the User Profile page.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Page Display" on page 4-14

Parameters

The only parameter is the option list to be sized.

Output

Returns the size of the specified option list.

getRequiredMsg

This function is used on presentation pages to populate the Java script with the desired error message. The error message is set from the content rule user interface for a field that is designated as required. It returns a required message based on the following algorithm:

1. The function looks for the fieldname:requiredMsg.
2. If the value exists, it is localized.
3. If the value is undefined or the message is empty, the function returns the default message and does not localize it. To localize the message, the caller of the function must use the `lc` function on the default message before passing it to this function (`getRequiredMsg`).

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

This function takes two parameters:

- Fieldname: Field associated with the error message.
- Message: Default message to be used.

See Also

- "lc" on page 8-201

getTextFile

Gets the Web version of a text file and returns its contents to a string.

- This can be used only with files that have a format containing text, such as text/html, text/plain, or text/xml.
- There must be enough information to determine the content item's Web URL. The dID value is the only required metadata field.

  The following information must be present:

  – dDocName, dWebExtension, dSecurityGroup, dRevLabel, and dDocType.

  – If Accounts are enabled, dDocAccount must also be specified.

  This information is available automatically after a GET_SEARCH_RESULTS call or after a DOC_INFO call.
- This function should be wrapped with a `cacheInclude` call for greater performance. That can, however, create a security issue if all docs go to the same cache.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

This function does not take parameters but uses variables on the page to determine the document's URL.

Example

```
<$loop SearchResults$>
    <$fullDocument=getTextFile()$>
    <h2><$dDocTitle$></h2>
    <div><$fullDocument$></div>
<$endloop$>
```

See Also

- "cacheInclude" on page 8-19

getTotalMemory

This function returns the amount of total memory in the Java Virtual Machine (JVM). This is a performance auditing function used on the System Audit Information page to determine how much memory the server is using.

Type and Usage

- "Global Functions" on page 3-11
- "Debugging" on page 4-9

Output

The total JVM memory in megabytes.

Example

```
<$totalMem = getTotalMemory()$>
```

See Also

- "getFreeMemory" on page 8-109

getUserValue

Retrieves the value of a user metadata field for the current user.

The parameter must refer to a column in the Users database table. Unlike the user personalization functions that have no support for global reference, information assigned to the user in the Users table can be available to the Content Server.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12
- "Users" on page 4-20

Parameters

The only parameter is a user-related variable.

Output

- Returns the value of the metadata field for the current user.
- Evaluates to TRUE if the user value was retrieved.
- Evaluates to FALSE if an error in retrieval occurred.
- Returns a StatusCode of `-1` if the value is unspecified or unknown.

Example

Returns the User Type of the currently logged on user:

```
<$getUserValue('dUserType')$>
```

getValue

This function has multiple uses:

Retrieves the value of a particular metadata field from local, active, or environment data.

Retrieves the value of a particular column from a specific ResultSet.

Retrieves information about ResultSet rows.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is either the type of data or the name of a ResultSet.
- The second parameter is the name of a metadata field, the column name, or a ResultSet row parameter.

Variations

You can also use a shorthand format that returns results identical to the `getValue(arg1,arg2)` function. The format uses the form `<$arg1.arg2$>`, where *arg1* and *arg2* are the literal string arguments "arg1" and "arg2" to the *getValue* function.

> **Note:** Data types (local, active, and env) and ResultSet parameters must start with the # symbol.

The following parameter combinations can be used:

getValue and Shorthand Formats	Description
getValue("#local",*fieldName*) <$#local.*fieldName*$>	Retrieves the value of the specified metadata field from the local data.
getValue("#active",*fieldName*) <$#active.*fieldName*$>	Attempts to retrieve the value of the specified metadata field from the data in the following order: **1.** Local data **2.** Active ResultSets **3.** All other ResultSets **4.** Environment settings
getValue("#env",*fieldName*) <$#env.*fieldName*$>	Retrieves the value of the specified metadata field from the environment settings.
getValue(*ResultSetName,fieldName*) <$*ResultSetName.fieldName*$>	Retrieves the value of the specified metadata field from the current row of the specified ResultSet.

getValue and Shorthand Formats	Description
getValue(*ResultSetName*,*columnName*) <$*ResultSetName*.*columnName*$>	Retrieves the value of the specified column from the current row of the specified ResultSet.
getValue(*columnName*) <$*columnName*$>	Retrieves the value of the specified column from the current row of the current ResultSet.
getValue(*ResultSetName*,"#row") <$*ResultSetName*.#row$>	Retrieves the number of the current row in the specified ResultSet. The first row is 0.
getValue(*ResultSetName*,"#numRows") <$*ResultSetName*.#numRows$>	Retrieves the total number of rows in the specified ResultSet.
getValue(*ResultSetName*,"#isRowPresent") <$*ResultSetName*.#isRowPresent$>	Checks if at least one row is present in the specified ResultSet. This is useful when looping manually with rsNext.
getValue(*ResultSetName*,"#isEmpty") <$*ResultSetName*.#isEmpty$>	Checks if the specified ResultSet does not contain any rows.

Output

- See Variations for output values.
- Returns an empty string if a value is not found.

Example

- Gets the Content ID from the ResultSet named *DOC_INFO*:

  ```
  <$name = getValue("DOC_INFO","dDocName")$>
  ```

  or

  ```
  <$name = <$DOC_INFO.dDocName$>
  ```

- Checks to see if the passed parameter *dDocType* (which is in the local data) equals the value in the active ResultSet:

  ```
  <$loop DocTypes$>
  <$if strEquals(#active.dDocType, getValue("#local","dDocType"))$>
  <!--do special HTML for selected document type-->
  <$endif$>
  <!-- additional statement-->
  <$endloop$>
  ```

See Also

- "Special Keywords" on page 2-11

getValueForSpecifiedUser

Retrieves the value of a user attribute for a specific user. This function can be useful for defining extended workflow functionality.

Type and Usage

- "Global Functions" on page 3-11
- "Users" on page 4-20
- "Workflow" on page 4-21

Parameters

Takes two parameters:

- The first parameter is the user name.
- The second parameter is the name of a column in the Users database table that specifies a user attribute (such as *dFullName*, *dUserType*, or *dEmail*).

Output

- Returns the value of the user attribute for the specified user.
- Returns an empty string if the value is not found.

Example

Retrieves the full name for *mjones*.

```
<$getValueForSpecifiedUser('mjones', 'dFullName')$>
```

HasLocalCopy

Checks whether the client computer has a copy of the requested content item in the download target directory.

Generally used to query the user whether to overwrite when downloading. For example, this variable is used by the Oracle ODMA Client.

Type and Usage

- "Setable Variables" on page 3-19
- "Content Items" on page 4-3
- "Clients" on page 4-2

Output

- Returns TRUE if a local copy is detected.
- Returns FALSE if a local copy is not detected.

Example

Checks for a local copy of the content item:

```
<$HasLocalCopy$>
```

See Also

- "ClientControlled" on page 8-24
- "IsNotLatestRev" on page 6-121

hasOptionList

Specifies that the metadata field has an option list.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the field has an option list.
- Returns FALSE if the field does not have an option list.

Example

Specifies that the field has an option list:

```
<$hasOptionList=1$>
```

Generates the option list values if the field has an option list:

```
<$if hasOptionList$>
    <$if isQuery$></td><td><$endif$>
    <$include std_option_list_entry$>
<$endif$>
```

See Also

- "fieldIsOptionList" on page 8-84

HeavyClient

Checks if the checkin is through the Content Server ODMA client or Upload applet.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Clients" on page 4-2

Output

- Returns TRUE if the ODMA client or Upload applet is being used for checkin.
- Returns FALSE if the ODMA client and Upload applet are not being used for checkin.

Example

Checks for check in method:

```
<$HeavyClient$>
```

HTMLEditorPath

Defines the path for the executable that is used to edit component files from within the Component Wizard.

This path is defined in the Component Wizard.

You should specify a text editor (such as WordPad) rather than a graphical HTML editor (such as FrontPage). Graphical editors can insert or change HTML tags and may cause Idoc Script tags to be converted into a string of characters that will not be recognized by the Content Server.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

- *IntradocDir*/config/config.cfg
- *DomainHome*/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
HTMLEditorPath=C:/Program Files/Windows NT/Accessories/wordpad.exe
```

htmlRefreshTimeout

Similar to `DefaultHtmlRefreshTimeoutInSeconds`. Defines the time, in seconds, that a Work In Progress page, My Checked-Out Content page or My Workflow Assignments page refreshes.

`htmlRefreshTimeout` can be set in the URL or the service's databinder. This can be used to set a different refresh time for different pages. If that is needed, do not set `DefaultHtmlRefreshTimeoutInSeconds` in the config.cfg file, but instead set `htmlRefreshTimeout` in the URL or databinder.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Page Display" on page 4-14

Example

In the page's URL:

```
&htmlRefreshTimeout=90
```

See Also

- "DefaultHtmlRefreshTimeoutInSeconds" on page 6-32
- "htmlRefreshUrl" on page 8-124

htmlRefreshUrl

Used to set the URL of the page to load after a Work In Progress page, My Checked-Out Content page or My Workflow Assignments page is refreshed. The default is the current page. This variable can be set in the URL of the page or the databinder of the service.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Page Display" on page 4-14

Example

In the page's URL:

```
&htmlRefreshUrl=http://www.home.com
```

See Also

- "DefaultHtmlRefreshTimeoutInSeconds" on page 6-32
- "htmlRefreshTimeout" on page 8-123

HttpAbsoluteCgiPath

Retrieves the Content Server CGI path as a complete URL.

This variable cannot be modified directly; to change the path, use the HttpAbsoluteCgiRoot configuration setting.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the path as a string.

Example

Returned for a Content Server (*domain*):

```
HttpAbsoluteCgiPath=http://localhost/domain/idcplg/
```

See Also

- "HttpCgiPath" on page 8-130
- "HttpAbsoluteCgiRoot" on page 8-126
- "HttpBrowserFullCgiPath" on page 8-129

HttpAbsoluteCgiRoot

Defines the Content Server CGI path as a complete URL.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Location

IntradocDir/config/config.cfg

Example

For a Content Server:

```
HttpAbsoluteCgiRoot=http://my_computer/idcplg/idc_cgi_isapi-instance.dll
```

See Also

- "HttpAbsoluteCgiPath" on page 8-125

HttpAbsoluteWebRoot

Defines the Web server root directory as a complete URL.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Directories and Paths" on page 4-9
- "Web Server" on page 4-21

Location

IntradocDir/config/config.cfg

Example

For a Content Server:

```
HttpAbsoluteWebRoot=http://my_computer/domain/
```

See Also

- "HttpRelativeWebRoot" on page 8-138

HttpAdminCgiPath

Retrieves the Admin Server CGI path as a relative URL.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the Admin Server CGI path as a string.

Example

Returned for a Content Server:

```
HttpAdminCgiPath=/idcplg/idc_cgi_isapi-instance.dll/cs-admin/pxs
```

HttpBrowserFullCgiPath

This variable is used to set explicit control over the CGI path construction made for applets and the Admin Server.

The Content Server evaluates the HTTP address in the address bar of the browser. If the `HttpIgnoreWebServerInternalPortNumber` configuration entry is not sufficient to assist the evaluation, `HttpBrowserFullCgiPath` can be set with an explicit answer.

Specify a relative path to the Content Server (`idc1` in the example below).

Type and Usage

- "Dynamic Variables" on page 3-10
- "Directories and Paths" on page 4-9
- "Clients" on page 4-2

Example

```
HttpBrowserFullCgiPath=http://localhost/idc1/idcplg
```

See Also

- "HttpIgnoreWebServerInternalPortNumber" on page 6-81

HttpCgiPath

Retrieves the Content Server CGI path.

The following configuration settings are used to determine the value of this variable:

Variable	Description
UseSSL	When set to TRUE, the secure sockets layer (SSL) is used (*https* instead of *http*).
isAbsoluteCgi	Defines whether the complete URL is used instead of a relative path. This is an internal flag set by the Content Server and is not intended for user configuration.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the path as a string.

Example

Returned for a Content Server:

```
HttpCgiPath=/domain/idcplg
```

See Also

- "HttpAbsoluteCgiPath" on page 8-125
- "HttpEnterpriseCgiPath" on page 8-132
- "UseSSL" on page 6-272

HttpCommonRoot

Retrieves the URL of the *common* directory.

Multiple content servers can share resources from one content server installation. This variable defines the URL path to the *DomainHome*/ucm/cs/common/ directory of the content server whose resources are being shared. For example, the *HttpCommonRoot* defines the prefix to use for accessing the shared common directory where Web applets are located.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the relative URL as a string. If the URL is external, the complete URL is returned rather than the relative path.

Example

Returned for a Content Server:

```
HttpCommonRoot=/domain/common/
```

See Also

- "HttpHelpRoot" on page 8-133
- "HttpImagesRoot" on page 8-134
- "HttpWebRoot" on page 8-141

HttpEnterpriseCgiPath

Retrieves the CGI path of a Content Server as a relative URL.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the Content Server CGI path as a string.

Example

Returned for a Content Server:

```
HttpEnterpriseCgiPath=/idcplg/idc_cgi_isapi-instance.dll
```

See Also

- "HttpCgiPath" on page 8-130

HttpHelpRoot

Retrieves the URL of the *help* directory.

Multiple content servers can share online help files from one Content Server installation. This variable defines the URL path to the *DomainDir*/ucm/cs/weblayout/help directory of the content server whose help files are being shared.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the relative URL as a string. If the URL is external, the complete URL is returned rather than the relative path.

Example

Returned for a Content Server:

```
HttpHelpRoot=/domain/help/
```

See Also

- "HttpCommonRoot" on page 8-131
- "HttpImagesRoot" on page 8-134
- "HttpWebRoot" on page 8-141

HttpImagesRoot

Retrieves the URL of the *images* directory.

Multiple content servers can share user interface images from one Content Server installation. This variable defines the URL path to the *DomainHome*/ucm/cs/weblayout/images directory of the content server whose image files are being shared.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the relative URL as a string. If the URL is external, the complete URL is returned rather than the relative path.

Example

Returned for a Content Server:

```
HttpImagesRoot=/domain/images/
```

See Also

- "HttpCommonRoot" on page 8-131
- "HttpHelpRoot" on page 8-133
- "HttpWebRoot" on page 8-141

HttpLayoutRoot

Retrieves the URL of the folder containing the current layout files. This is useful if you have additional JavaScript resources specific to the current layout.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

HttpRelativeAdminRoot

Retrieves the relative URL of the Admin Server.

Defaults to `/cs-admin/` during installation.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the relative URL as a string.

Example

Returned for a Content Server:

```
HttpRelativeAdminRoot=/cs-admin/
```

HttpRelativeCgiRoot

Defines the Content Server CGI path as a relative URL.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Location

IntradocDir/config/config.cfg

Example

For a Content Server:

```
HttpRelativeCgiRoot=/idcplg/idc_cgi_isapi-instance.dll
```

See Also

- "HttpAbsoluteCgiRoot" on page 8-126

HttpRelativeWebRoot

Defines the Web server root directory as a relative URL.

- A relative root such as */domain/* is used rather than a full root such as *http://www.mycomputer.com/domain/*.
- Returns the relative Web root directory as a string.
- There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9
- "Web Server" on page 4-21

Location

- System Properties, Internet tab, Http Relative Web Root
- Content Server: *IntradocDir*/config/config.cfg

Example

As a configuration setting, defines the relative Web root:

```
HttpRelativeWebRoot=/domain/
```

As script, returns the relative Web root as a string:

```
<$HttpRelativeWebRoot$>
```

See Also

- "HttpAbsoluteWebRoot" on page 8-127

HttpServerAddress

Defines the Web server address as a partial URL.

A partial URL such as *mycomputer* is used rather than a full address such as *http://www.mycomputer.com/*.

Returns the Web server address as a string.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9
- "Web Server" on page 4-21

Location

- System Properties, Internet tab, HTTP Server Address
- Content Server: *IntradocDir*/config/config.cfg

Example

As a configuration setting, defines the Web server address:

```
HttpServerAddress=mycomputer
```

As script, returns the Web server address as a string:

```
<$HttpServerAddress$>
```

See Also

- "IdcCommandServerHost" on page 6-86
- "IntradocServerHostName" on page 6-102

HttpSystemHelpRoot

Similar to the `HttpHelpRoot` variable except this variable returns the path to the help files for the default system language.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Content Server" on page 4-4
- "Directories and Paths" on page 4-9

Output

Returns the path to the help files.

Example

```
HttpSystemHelpRoot=/domain/help/
```

See Also

- "HttpCommonRoot" on page 8-131
- "HttpHelpRoot" on page 8-133
- "HttpImagesRoot" on page 8-134
- "HttpWebRoot" on page 8-141

HttpWebRoot

Retrieves the URL path of the *weblayout* directory.

Multiple content servers can share resources from one Content Server installation. This variable defines the URL path to the *DomainHome*/ucm/cs/weblayout/ directory of the content server whose resource files are being shared.

The following configuration settings determine the value of this variable:

Variable	Description
UseSSL	When set to TRUE, the secure sockets layer (SSL) is used (*https* instead of *http*).
isAbsoluteWeb	Defines whether the complete URL is used instead of a relative path. This is an internal flag set by the Content Server and is not intended for user configuration.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Directories and Paths" on page 4-9

Output

Returns the relative URL as a string. If the URL is external, the complete URL is returned rather than the relative path.

Example

Returned for a Content Server:

```
HttpWebRoot=/domain/
```

See Also

- "UseSSL" on page 6-272
- "HttpCommonRoot" on page 8-131
- "HttpHelpRoot" on page 8-133
- "HttpImagesRoot" on page 8-134

IdcAnalyzeLogDir

Specifies the location where the Content Server Analyzer utility writes output files and transfers extra files detected during the file system analysis process.

You can specify a partial or complete directory path.

- If a partial directory path (such as *analyzer_logs/*) is specified, this subdirectory is created in the *DomainHome*/ucm/cs/bin/ directory.
- If a complete directory path (such as c:/*analyzer_logs/*) is specified, this is the log directory.

The log file name is specified by the LogName setting.

The default is *DomainHome*/ucm/cs/bin/logs/.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9
- "Directories and Paths" on page 4-9

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

As a configuration setting:

```
IdcAnalyzeLogDir=f:/domain/analyzer_logs/
```

See Also

- "LogName" on page 8-210
- "SafeDir" on page 8-283
- "TempDir" on page 8-322

IdcHomeDir

Defines the location of the server media for Oracle Universal Content Management (Oracle UCM), which can run media for any of the following: Content Server, Inbound Refinery, and Universal Records Management.

The location can not be modified by an administration after Oracle UCM is deployed; it is a fixed location based on the current location of the media

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

DomainHome/ucm/idc

Example

Used as a configuration setting:

```
IdcHomeDir=<MW_HOME>/user_projects/domains/<base_domain>/ucm/idc
```

IdcLocale

Sets the value that is placed in the HTTP cookie of the client to keep track of the current user's locale.

Type and Usage

- "Configuration Variables" on page 3-2
- "Localization" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IdcLocale=Deutsch
```

IdcServerBindAddress

Used in clustered environments to bind the Content Server to a specific IP address.

Type and Usage

- "Configuration Variables" on page 3-2

Location

Set in the config.cfg file of the Content Server listening for requests:

IntradocDir/config/config.cfg

Example

Used as a configuration variable:

```
IdcServerBindAddress=IPaddress
```

IdcServerSocketQueueDepth

Specifies the depth of the TCP/IP socket queue.

This setting can be used to tune Content Server performance, particularly in a clustered environment. Decreasing this value might limit Content Server throughput.

Default is 200.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IdcServerSocketQueueDepth=256
```

idocTestForInclude

This function tests to find out if a dynamichtml resource exists.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamicdata" on page 4-10

Parameters

This function has one parameter: `includeName` is the name of the dynamichtml resource.

Output

If the resource exists, then it returns True.

Example

```
<$if idocTestForInclude("std_page_begin")$>
    <$include std_page_begin$>
<$endif$>
```

inc

Adds the code from an include resource to the page.

This function does the same thing as the *include* keyword, except that it can take an Idoc Script variable as the parameter. See Keywords Versus Functions for more information.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is a variable that defines the name of an include.

Output

Displays the code that is defined in the specified include resource.

Example

Say you wanted to execute some Idoc Script for some, but not all, of your custom metadata fields. You could dynamically create includes based on the field names (such as *specific_include_xComments*) by executing this Idoc Script:

```
<$loop DocMetaDefinition$>
    <$myInclude = "specific_include_" & dName$>
    <$exec inc(myInclude)$>
<$endloop$>
```

Note the use of the `exec` keyword, which suppresses the output of the include specified by the `inc` function. If you don't use *exec* before the `inc` function, the HTML inside the specified include will be displayed on the page.

Note that if the `specific_include_xComments` does not exist, this code will not throw an error because the output is not being displayed.

See Also

- "Keywords Versus Functions" on page 2-12
- "include" on page 8-150
- "setResourceInclude" on page 8-290

incGlobal

Includes the entire contents of a text file in the display of the current page.

This function is used to generate the default portal page.

A global include file is a text file that contains HTML code, Idoc Script code, or both. It is loaded on server startup.

Global include files must have a *.inc* extension, and must be located in the *DomainHome*/ucm/cs/data/pages/ directory.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is the name of a global include file without the .inc file extension.

Output

Displays the code that is defined in the specified global include file.

Example

When you change the portal page using the Update Portal function in the Web Layout Editor, a global include file called *portal_message.inc* is created to contain your modified text. This file is then referenced in the *pne_home_page.htm* template file as follows:

```
<$incGlobal("portal_message")$>
```

include

Adds the code from an include resource (defined by `<@dynamichtml include_name@>`) to the page.

This keyword is the most commonly used command in Idoc Script, as it allows you to reuse small chunks of code on multiple template pages.

See Includes for more information.

Type and Usage

- "Special Keywords" on page 2-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is the name of the include.

Output

Displays the code that is defined in the specified include resource.

Example

The following includes are used to create the beginning and end of most Content Server Web pages. These includes are defined in the following file:

IdcHomeDir/resources/core/standard_page.htm

```
<$include std_html_head_declarations$>
<$include body_def$>
<$include std_page_begin$>
    Hello World!
<$include std_page_end$>
```

See Also

- "Keywords Versus Functions" on page 2-12
- "Includes" on page 2-3
- "setResourceInclude" on page 8-290
- "inc" on page 8-148

incTemplate

Adds the contents of a Content Server template to a page, after evaluating any Idoc Script.

You can use this function to include the content of an entire template. However, this usage is discouraged because resource includes are usually sufficiently flexible to support all requirements for the sharing of Idoc Script between pages.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is the name of a Content Server template file without the .htm file extension.

Output

Displays the code that is defined in the specified template.

Example

Retrieves the *new_look* template file.

```
<$incTemplate("new_look")$>
```

IntradocDir

The directory containing the configuration and data specific to a Content Server instance.

There is no default value. The value depends on which configuration is specified: `cs` (for Content Server), `ibr` (for Inbound Refinery), or `urm` (for Universal Records Management).

It is recommended that you move the directory onto a network share if you are managing a load balancing cluster.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

- Content Server*DomainHome*/ucm/cs/bin/intradoc.cfg
- Inbound Refinery: *DomainHome*/ucm/ibr/bin/intradoc.cfg
- Universal Records Management: *DomainHome*/ucm/urm/bin/intradoc.cfg

Example

Used as a configuration entry:

```
IntradocDir=c:/ucm/cs1
```

isActiveTrace

Checks if a section is being traced in the core.

Type and Usage

- "Global Functions" on page 3-11
- "Debugging" on page 4-9

Output

Returns active trace results.

Example

```
<$isActiveTrace()$>
```

See Also

- "isVerboseTrace" on page 8-193

IsAutoArchiver

Enables the automatic import or transfer of content items.

When set to TRUE, the Content Server will automatically import and transfer archives.

When set to FALSE, the Content Server will not automatically import or transfer archives.

The NoAutomation setting overrides this setting.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IsAutoArchiver=false
```

See Also

- "NoAutomation" on page 6-179

isCheckin

Specifies if the current page is a checkin page.

When set to TRUE, the current page is identified as a checkin page.

When set to FALSE, the current page is not identified as a checkin page.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the page is a checkin page.
- Returns FALSE if the page is not a checkin page.

Example

Set at the top of a page:

```
isCheckin=1
```

Used as script:

```
<$if isCheckin or isUpdate or isQuery$>
    <$isFormSubmit = 1$>
    <$if not isQuery$>
        <$isEditMode = 1$>
        <$formName = "Checkin"$>
    <$endif$>
    <$else$>
        <$isFormSubmit = ""$>
        <$isEditMode = ""$>
<$endif$>
```

IsCurrentNav

Checks if the page currently being displayed is the same as the page being looped over while building the Next/Previous navigation on search results pages.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Page Display" on page 4-14
- "Searching" on page 4-17

Output

- Returns TRUE if the navigation loop is at the same page as the current page.
- Returns FALSE if the navigation loop is not at the same page as the current page.

Example

If the navigation loop is at the current page, the page number is displayed as plain text. If the navigation loop is not at the current page, the page number is displayed as a hypertext link:

```
<$loop NavigationPages$>
<$if IsCurrentNav$>
    <$HeaderPageNumber$>
<$else$>
    <a href="<$strRemoveWs(inc("searchapi_navigation_specific_page"))$>">
    <$HeaderPageNumber$></a>
<$endif$>
<$endloop$>
```

isDocPage

Specifies if the current page is a page that displays metadata (such as search, checkin, and content information pages).

When set to TRUE, the current page is identified as a document page.

When set to FALSE, the current page is not identified as a document page.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the page is a document page.
- Returns FALSE if the page is not a document page.

Example

Set at the top of a page:

```
isDocPage=1
```

Used as script:

```
<$if isDocPage$>
    <!--statement-->
<$endif$>
```

IsDpSubmitErrorFatal

Used with Content Profiles. When set to TRUE, enables the Content Server to issue a fatal error when a global rule violation is encountered.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Profiles" on page 4-3

Location

IntradocDir/config/config.cfg

Example

```
IsDpSubmitErrorFatal=true
```

See Also

- *Managing Repository Content Guide*

IsDynamic

Checks if the page is presented dynamically to the user.

Most pages viewed by the user are dynamic. However, some static pages are designed to be delivered to the user without changes. Examples are the guest portal page and the content of some auto generated e-mails.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Page Display" on page 4-14

Output

- Returns TRUE if the page is being presented dynamically to the user.
- Returns FALSE if the page is static or cannot be displayed.

Example

Evaluates whether the page is presented dynamically:

```
<$if IsDynamic$>
    <a href="<$redirect$>">
<$endif$>
```

isEditMode

Specifies if metadata fields on the current page can be edited.

This variable is set on checkin and content information update pages.

When set to TRUE, metadata fields on the current page can be edited.

When set to FALSE, metadata fields on the current page cannot be edited.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if metadata fields on the current page can be edited.
- Returns FALSE if metadata fields on the current page cannot be edited.

Example

Set at the top of a page:

```
isEditMode=1
```

Used as script:

```
<$if isCheckin or isUpdate or isQuery$>
    <$isFormSubmit = 1$>
    <$if not isQuery$>
        <$isEditMode = 1$>
        <$formName = "Checkin"$>
    <$endif$>
    <$else$>
        <$isFormSubmit = ""$>
        <$isEditMode = ""$>
<$endif$>
```

isExcluded

Completely excludes the metadata field from the page.

Type and Usage

- "Common Field Display Variables" on page 3-16
- "Field Display" on page 4-11

Output

- Returns TRUE if the metadata field is excluded from the page.
- Returns FALSE if the metadata field is not excluded from the page.

Example

Excludes the current metadata field from the page:

```
<$isExcluded=1$>
```

Sets a custom variable depending on the active value of *isExcluded* for the current metadata field:

```
<$isCustomExcluded = getValue("#active", fieldName & ":isExcluded")$>
$isCustomRelocated = getValue("#active", fieldName & ":isRelocated")$>
<$if isCustomExcluded or (isCustomRelocated and not isRelocated) or isExcluded or
(isFieldHidden and not isFormSubmit)$>
    <$isFieldExcluded = 1$>
<$endif$>
```

See Also

- "isFieldExcluded" on page 8-164
- "isHidden" on page 8-169
- "isInfoOnly" on page 8-171
- "isRelocated" on page 8-182
- "optionListScript" on page 8-225

IsExternalUser

Checks if the user is accessing the Content Server from an external system, such as a LDAP system.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Users" on page 4-20

Output

- Returns TRUE if the user is accessing the Content Server from an external system.
- Returns FALSE if the user is accessing the Content Server directly.

Example

Checks if user is at an external location:

```
<$IsExternalUser$>
```

isFalse

Checks if a string or expression evaluates to FALSE.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is the string or expression to be evaluated.

Output

If the parameter is a string:

- Returns TRUE if the string begins with F, f, N, n, or is 0.
- Returns FALSE if the string begins with any other character.

If the parameter is not a string:

- Returns TRUE if the value is 0.
- Returns FALSE if the value is not 0.

Example

Evaluates the string `false` and returns TRUE (1):

```
<$isFalse("false")$>
```

Evaluates that the integer five is greater than one and returns FALSE (0):

```
<$isFalse(5>1)$>
```

Evaluates the result of the equation as zero and returns TRUE (1):

```
<$isFalse(1-1)$>
```

Evaluates the string equality statement as true and returns FALSE (0):

```
<$isFalse(strEquals("abc","abc"))$>
```

See Also

- "isTrue" on page 8-189

isFieldExcluded

Specifies that the metadata field is excluded.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the field is excluded.
- Returns FALSE if the field is not excluded.

Example

Determines if the metadata field is excluded:

```
<$if isCustomExcluded or (isCustomRelocated and not isRelocated) or isExcluded or
(isFieldHidden and not isFormSubmit)$>
    <$isFieldExcluded = 1$>
<$endif$>
```

See Also

- "isExcluded" on page 8-161
- "isFieldHidden" on page 8-165
- "isFieldInfoOnly" on page 8-166
- "isFieldMemo" on page 8-167

isFieldHidden

Specifies that the metadata field is hidden.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the field is hidden.
- Returns FALSE if the field is not hidden.

Example

Determines if the metadata field is hidden:

```
<$if isHidden or isCustomHidden$>
    <$isFieldHidden = 1$>
<$else$>
    <$isFieldHidden = ""$>
<$endif$>
```

See Also

- "isHidden" on page 8-169
- "isFieldExcluded" on page 8-164
- "isFieldInfoOnly" on page 8-166
- "isFieldMemo" on page 8-167

isFieldInfoOnly

Specifies that the metadata field is an information only field.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the field is an information only field.
- Returns FALSE if the field is not an information only field.

Example

Determines if the metadata field is information only:

```
<$if isInfo or isCustomInfo or isFieldHidden or isInfoOnly$>
    <$isFieldInfoOnly = 1$>
<$else$>
    <$isFieldInfoOnly = ""$>
<$endif$>
```

See Also

- "isInfoOnly" on page 8-171
- "isFieldExcluded" on page 8-164
- "isFieldHidden" on page 8-165
- "isFieldMemo" on page 8-167

isFieldMemo

Specifies that the metadata field is a memo field.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the field is a memo field.
- Returns FALSE if the field is not a memo field.

Example

Determines if the field is a memo field:

```
<@dynamichtml compute_std_entry_type_info@>
<$if not hasOptionList and #active.fieldType like "Memo"$>
    <$isFieldMemo = 1$>
<$else$>
    <$isFieldMemo = ""$>
<$endif$>
<@end@>
```

See Also

- "isFieldExcluded" on page 8-164
- "isFieldHidden" on page 8-165
- "isFieldInfoOnly" on page 8-166

isFormSubmit

Specifies if the current page is a submittable HTML form.

When set to TRUE, the current page is a submittable HTML form.

When set to FALSE, the current page is not a submittable HTML form.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the page is a submittable HTML form.
- Returns FALSE if the page is not a submittable HTML form.

Example

Set at the top of a page:

```
isFormSubmit=1
```

Adds a colon after the field captions if the page is not a submittable HTML form:

```
<@dynamichtml std_field_caption@>
    <span class=<$fieldCaptionStyle$>>
    <$fieldCaption$><$if not isFormSubmit$><$":"$><$endif$>
    </span>
<@end@>
```

isHidden

Hides the metadata field from the user but includes the field as hidden data on the page.

Type and Usage

- "Common Field Display Variables" on page 3-16
- "Field Display" on page 4-11

Output

- Returns TRUE if the metadata field is hidden on the page.
- Returns FALSE if the metadata field is not hidden the page.

Example

Hides the current metadata field from the user:

```
<$isHidden=1$>
```

Sets a custom variable depending on the active value of *isHidden* for the current metadata field:

```
<$isCustomHidden = getValue("#active", fieldName & ":isHidden")$>
    <$if isHidden or isCustomHidden$>
<$isFieldHidden = 1$>
<$else$>
    <$isFieldHidden = ""$>
<$endif$>
```

See Also

- "isExcluded" on page 8-161
- "isFieldHidden" on page 8-165
- "isInfoOnly" on page 8-171
- "isRelocated" on page 8-182
- "optionListScript" on page 8-225

isInfo

Specifies if the current page is an information-only page.

When set to TRUE, the current page is identified as an information-only page.

When set to FALSE, the current page is not identified as an information-only page.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the page is an information-only page.
- Returns FALSE if the page is not an information-only page.

Example

Set at the top of a page:

```
isInfo=1
```

Used as script:

```
<$if isInfo$>
    <$captionFieldWidth="30%"$>
    <$captionEntryWidth="70%"$>
<$elseif isEditMode$>
    <$captionFieldWidth="20%"$>
    <$captionEntryWidth="80%"$>
<$endif$>
```

isInfoOnly

Displays the metadata field as information only, not as an input field.

Type and Usage

- "Common Field Display Variables" on page 3-16
- "Field Display" on page 4-11

Output

- Returns TRUE if the metadata field is displayed as information only.
- Returns FALSE if the metadata field can be edited.

Example

Displays the current metadata field as information only:

```
<$isInfoOnly=1$>
```

Sets a custom variable depending on the active value of *isInfoOnly* for the current metadata field:

```
<<$isCustomInfo = getValue("#active", fieldName & ":isInfoOnly")$>
<$if isInfo or isCustomInfo or isFieldHidden or isInfoOnly$>
    <$isFieldInfoOnly = 1$>
<$else$>
    <$isFieldInfoOnly = ""$>
<$endif$>
```

See Also

- "isExcluded" on page 8-161
- "isFieldInfoOnly" on page 8-166
- "isHidden" on page 8-169
- "isRelocated" on page 8-182
- "optionListScript" on page 8-225

IsJava

Displays the local data of a Content Server Web page.

This variable can be set as a flag on a page or as a parameter to a service call.

Type and Usage

- "Setable Variables" on page 3-19
- "Debugging" on page 4-9

Output

When set to TRUE, returns the local data in the DataBinder.

Example

When included in the code on a Content Server Web page, displays the local data of a page:

```
<$IsJava=1$>
```

When included in a Content Server URL, displays the local data for the New Checkin page:

```
http://myinstance.com/idcplg/idc_cgi_isapi-instance.dll?IdcService=CHECKIN_NEW_
FORM&IsJava=1
```

IsJspServerEnabled

Enables Java Server Page functionality.

Java Server Page support enables developers to access and modify Content Server content, ResultSets, personalization and security definitions, and predefined variables and configuration settings through Java Server Pages rather than through standard Content Server component architecture. Content Server services and Idoc Script functions can also be executed from Java Server Pages, which reside as executable content in the Content Server.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "JSP" on page 4-13

Location

- System Properties, Server tab, Execute Java Server Page (Jsp)
- Admin Server, General Configuration, Enable Java Server Page (Jsp)
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
IsJspServerEnabled=true
```

See Also

- "JspEnabledGroups" on page 8-199

IsMac

Checks if the client browser is running on a Macintosh operating system.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Clients" on page 4-2

Output

- Returns TRUE if the client browser is running on a Mac.
- Returns FALSE if the client browser is not running on a Mac.

Example

Redirects to a different URL if the browser is running on a Mac:

```
<$if IsMac$>
    <a href="<$redirect$>">
<$endif$>
```

IsMaxRows

Checks if there are more results on a Work In Progress or Report page than the MaxQueryRows setting allows.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Page Display" on page 4-14

Output

- Returns TRUE if the number of results is greater than the number of rows permitted.
- Returns FALSE if the number of results is less than the number of rows permitted.

Example

Returns an error message if the maximum number of rows is exceeded:

```
<$elseif IsMaxRows$>
<table border=0 cellpadding=1 cellspacing=1 width="100%">
    <tr>
    <span class=smallHighlight><$lc("wwOutputLimitedByMaxRows")$>
    </span>
</tr>
</table>
```

isMultiOption

Specifies that a metadata field option list allows multiple values to be selected.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the field is a multiselect option list.
- Returns FALSE if the field is not a multiselect option list.

Example

Specifies that the field is a multiselect option list:

```
<$isMultiOption=1$>
```

Determines the type of option list:

```
<$if #active.fieldOptionListType like "*multi*"$>
    <$exec isMultiOption=1$>
<$elseif #active.fieldOptionListType like "access*"$>
    <$exec isAccessList=1$>
    <$exec isStrictList=1$>
<$elseif not (#active.fieldOptionListType like "*combo*")$>
    <!--Strict choice-->
    <$exec isStrictList=1$>
<$endif$>
```

See Also

- "isStrictList" on page 8-187

IsMultiPage

Checks if multiple pages are needed for search results. This variable depends on the number of rows displayed per page, which is defined by the `ResultCount` input variable (default is 25).

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Page Display" on page 4-14
- "Searching" on page 4-17

Output

- Returns TRUE if the number of search results is greater than the number of rows permitted per page.
- Returns FALSE if the number of search results is less than the number of rows permitted per page.

Example

Evaluates number of rows and determines if multiple pages are needed:

```
<$IsMultiPage$>
```

isNew

Checks if the content item is new or a revision. This variable is set to TRUE by the CHECKIN_NEW_FORM and CHECKIN_SIMILAR_FORM standard services.

Type and Usage

- "Value Variables" on page 3-21
- "Content Items" on page 4-3
- "Page Display" on page 4-14

Output

- Returns TRUE if the content item is new.
- Returns FALSE if the content item is a revision to an existing content item.

Example

If the content item is new, a specified service is performed:

```
<$if isNew$>
    <input type=hidden name=IdcService value="CHECKIN_NEW">
<$endif$>
```

If the content item is a revision, the original content item author is used:

```
<$if not isNew$>value="<$dDocAuthor$>"<$endif$>
```

If the content item is new, the default accounts for the current user are used:

```
<$if isNew$>
    <$defaultAccounts$>
<$endif$>
```

IsOverrideFormat

Enables users to select a different conversion format upon checkin.

When set to TRUE, a Format option list is displayed for the Primary File and Alternate File on the checkin page.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Field Display" on page 4-11

Location

- System Properties, Options tab, Allow override format on check in
- Admin Server, General Configuration, Allow override format on check in
- *IntradocDir*/config/config.cfg

Example

As a configuration setting:

```
IsOverrideFormat=true
```

As script, returns the value of the override format function:

```
<$IsOverrideFormat$>
```

isPhysicallySplitDir

Tells the Content Server that the *vault* and *weblayout* directories are on different file systems.

When set to TRUE, tells the server that the directories are on different file systems.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
isPhysicallySplitDir=true
```

See Also

- "WeblayoutDir" on page 8-361

isQuery

Specifies if the current page is a search page.

When set to TRUE, the current page is identified as a search page.

When set to FALSE, the current page is not identified as a search page.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the page is a search page.
- Returns FALSE if the page is not a search page.

Example

Set at the top of a page:

```
isQuery=1
```

Used as script:

```
<$if isCheckin or isUpdate or isQuery$>
    <$isFormSubmit = 1$>
    <$if not isQuery$>
<$isEditMode = 1$>
<$formName = "Checkin"$>
    <$endif$>
    <$else$>
<$isFormSubmit = ""$>
<$isEditMode = ""$>
<$endif$>
```

isRelocated

Excludes the metadata field unless the local variable is set to TRUE while the include is evaluated.

This variable is typically used to set a custom location for a metadata field. It allows a field to be defined on a page more than once, with the location of the field that is actually generated depending on the value of this variable.

Type and Usage

- "Common Field Display Variables" on page 3-16
- "Field Display" on page 4-11

Output

- Returns TRUE if the metadata field is included at that location on the page.
- Returns FALSE if the metadata field is excluded from that location on the page.

Example

Includes the current metadata field on the page:

```
<$isRelocated=1$>
```

Sets a custom variable depending on the active value of *isRelocated* for the current metadata field:

```
<$isCustomExcluded = getValue("#active", fieldName & ":isExcluded")$>
<$isCustomRelocated = getValue("#active", fieldName & ":isRelocated")$>
<$if isCustomExcluded or (isCustomRelocated and not isRelocated) or isExcluded or
(isFieldHidden and not isFormSubmit)$>
    <$isFieldExcluded = 1$>
<$endif$>
```

See Also

- "isExcluded" on page 8-161
- "isHidden" on page 8-169
- "isInfoOnly" on page 8-171
- "optionListScript" on page 8-225

IsRequestError

Checks if there is a request error condition present on the Content Server by evaluating the StatusCode variable.

If *StatusCode* is set to a negative numeric value (-1), there is a request error condition present on the Content Server.

The typical behavior when a request error condition is present is to abort the display of the current page and substitute an error page. See abortToErrorPage.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Debugging" on page 4-9
- "Content Server" on page 4-4

Output

- Returns TRUE if there is a request error condition present on the Content Server (`StatusCode = -1`).
- Returns FALSE if there is no request error condition present on the Content Server (*StatusCode* is other than a negative numeric value).

Example

Evaluates the request error condition:

```
<$IsRequestError$>
```

See Also

- "abortToErrorPage" on page 8-2
- "executeService" on page 8-73
- "StatusCode" on page 6-249

isRequired

Specifies if a value is required for a metadata field.

When set to TRUE, the metadata field is required.

When set to FALSE, the metadata field is optional.

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

- Returns TRUE if the metadata field is required.
- Returns FALSE if the metadata field is optional.

Example

Defines the *dDocTitle* metadata field as a required field:

```
<$fieldName = "dDocTitle", fieldCaption = lc("wwTitle"), isRequired = 1, fieldType
= "BigText", requiredMsg = lc("wwSpecifyTitle")$>
```

See Also

- "requiredMsg" on page 8-243

IsSavedQuery

Checks if a query has been saved to the current user's portal navigation bar.

Type and Usage

- "Setable Variables" on page 3-19
- "Page Display" on page 4-14
- "Searching" on page 4-17

Output

- Returns TRUE if the current query has been saved.
- Returns FALSE if the current query has not been saved or no query is found.

Example

Evaluates query status:

```
<$IsSavedQuery$>
```

IsSoap

Displays the local SOAP data of a Content Server Web page.

This variable can be set as a flag on a page or as a parameter to a service call.

Type and Usage

- "Setable Variables" on page 3-19
- "Debugging" on page 4-9

Output

When set to TRUE, returns the underlying SOAP code.

Example

When included in the code on a Content Server Web page:

```
<$IsSoap=1$>
```

When included in a Content Server URL:

```
http://myinstance.com/idcplg/idc_cgi_isapi-instance.dll?IdcService=CHECKIN_NEW_
FORM&IsSoap=1
```

isStrictList

Specifies that a metadata field option list does not allow multiple values to be selected.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the field is not a multiselect option list.
- Returns FALSE if the field is a multiselect option list.

Example

Specifies that the field is not a multiselect option list:

```
<$isStrictList=1$>
```

Determines the type of option list:

```
<$if #active.fieldOptionListType like "*multi*"$>
    <$exec isMultiOption=1$>
<$elseif #active.fieldOptionListType like "access*"$>
    <$exec isAccessList=1$>
    <$exec isStrictList=1$>
<$elseif not (#active.fieldOptionListType like "*combo*")$>
    <!--Strict choice-->
    <$exec isStrictList=1$>
<$endif$>
```

See Also

- "isMultiOption" on page 8-176

IsSun

Checks if the client browser is running on a Sun system.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Clients" on page 4-2

Output

- Returns TRUE if the client browser is running on Sun.
- Returns FALSE if the client browser is not running on Sun.

Example

Redirects to a different URL if the browser is running on Sun:

```
<$if IsSun$>
    <a href="<$redirect$>">
<$endif$>
```

isTrue

Checks if a string or expression evaluates to TRUE.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is the string or expression to be evaluated.

Output

If the parameter is a string:

- Returns TRUE if the string begins with T, t, Y, y, or is 1.
- Returns FALSE if the string begins with any other character.

If the parameter is not a string:

- Returns TRUE if the value is not 0.
- Returns FALSE if the value is 0.

Example

Evaluates the string `yes` and returns TRUE (1):

```
<$isTrue("yes")$>
```

Evaluates that the integer five is greater than one and returns TRUE (1):

```
<$isTrue(5>1)$>
```

Evaluates the result of the equation as zero and returns FALSE (0):

```
<$isTrue(1-1)$>
```

Evaluates the string equality statement as true and returns TRUE (1):

```
<$isTrue(strEquals("abc","abc"))$>
```

See Also

- "isFalse" on page 8-163

isUpdate

Specifies if the current page is a content information update page.

When set to TRUE, the current page is identified as a content information update page.

When set to FALSE, the current page is not identified as a content information update page.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the page is a content information update page.
- Returns FALSE if the page is not a content information update page.

Example

Set at the top of a page:

```
isUpdate=1
```

Used as script:

```
<$if isCheckin or isUpdate or isQuery$>
    <$isFormSubmit = 1$>
    <$if not isQuery$>
        <$isEditMode = 1$>
        <$formName = "Checkin"$>
    <$endif$>
    <$else$>
        <$isFormSubmit = ""$>
        <$isEditMode = ""$>
<$endif$>
```

isUploadFieldScript

Specifies that an include is being used inside JavaScript. It is used to determine how metadata fields are uploaded.

When set to TRUE, the include is being used inside JavaScript.

When set to FALSE, the include is being used inside JavaScript.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the include is being used inside JavaScript.
- Returns FALSE if the include is not being used inside JavaScript.

Example

Set in the include that defines the JavaScript wrapper:

```
<@dynamichtml std_upload_java_script@>
    <script language="JavaScript">
    // Java script for uploading.
    <$isUploadFieldScript = 1$>
    <$include std_upload_info_script$>
    <$isUploadFieldScript = ""$>
    </script>
<@end@>
```

Used as script:

```
<$if isUploadFieldScript$>
    <$defaultFieldInclude = "std_file_entry"$>
<$else$>
    <$defaultFieldInclude = "std_nameentry_row"$>
    <$fieldCaptionInclude = "std_field_caption"$>
    <$fieldEntryInclude = "std_file_entry"$>
<$endif$>
```

IsUserEmailPresent

Checks if an e-mail address is defined for the current user.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Users" on page 4-20

Output

- Returns TRUE if an e-mail address is defined for the user.
- Returns FALSE if an e-mail address is not defined for the user.

Example

Checks whether the user is logged in and has an e-mail address before performing a function.

```
<@dynamichtml subscription_action_script@>
    function allowSubscription(form)
    {
    <$if IsLoggedIn$>
    <$if IsUserEmailPresent$>
    <$else$>
    …
    }
<@end@>
```

isVerboseTrace

Checks if a section is being traced in the core. Verbose trace generates a full report.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Output

- Returns TRUE if tracing is set to Verbose.
- Returns FALSE if tracing is set to any other debug level.

Example

```
<$isVerboseTrace()$>
```

See Also

- "isActiveTrace" on page 8-153

IsWindows

Checks if the client browser is running on a Windows operating system.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Clients" on page 4-2

Output

- Returns TRUE if the client browser is running on Windows.
- Returns FALSE if the client browser is not running on Windows.

Example

Redirects to a different URL if the browser is running on Windows:

```
<$if IsWindows$>
    <a href="<$redirect$>">
<$endif$>
```

IsXml

Displays a raw dump of the HTML form when set to TRUE in a URL to a HCSP or HCSF file. This is useful for extracting form data or submitting forms from a remote application.

Type and Usage

- "Setable Variables" on page 3-19
- "Debugging" on page 4-9

Output

- Returns the XML data contained in the form.

See Also

- "IsJava" on page 8-172
- "IsSoap" on page 8-186

js

Formats a string for use in a "..." JavaScript literal string declaration.

This function performs string manipulation such as changing double quotes to single quotes.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string formatted for use in a JavaScript string declaration.

Example

Formats the string *variablestring* for use in a JavaScript string declaration:

```
<$js("variablestring")$>
```

JspAdminQuery

Defines which files are to be made available as Web application files.

This setting is generally used to define the query to find the Web application archiver (*.war* file) when deploying the Oracle JavaBean.

Type and Usage

- "Configuration Variables" on page 3-2
- "JSP" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
JspAdminQuery=dExtension <matches> war
```

See Also

- "IsJspServerEnabled" on page 8-173

JspDefaultIndexPage

Defines the default page for a Web application.

This is a comma-delimited list of file names that defines a search sequence. If the first file is not present, the second file will be the default page, and so on.

Default is *index.html,index.htm,index.jsp*.

Type and Usage

- "Configuration Variables" on page 3-2
- "JSP" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
JspDefaultIndexPage=index.html,index.htm,index.jsp
```

See Also

- "IsJspServerEnabled" on page 8-173

JspEnabledGroups

Defines the security groups to be enabled for Java Server Page functionality.

Because Java Server Pages usually have full privileges to any resource on the hosting system, it can be important to restrict Java Server Pages to security groups which allow only contributors with certain permissions.

Type and Usage

- "Configuration Variables" on page 3-2
- "JSP" on page 4-13

Location

- System Properties, Server tab, Jsp Enabled Groups
- Admin Server, General Configuration, Jsp Enabled Groups
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
JspEnabledGroups=jsp,group1
```

See Also

- "IsJspServerEnabled" on page 8-173

JvmCommandLine

Defines the location of the executable for the Java Virtual Machine command line.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Inbound Refinery" on page 4-6

Location

- Content Server: *DomainHome*/ucm/cs/bin/intradoc.cfg
- Inbound Refinery: *DomainHome*/ucm/ibr/bin/intradoc.cfg

Example

Used as a configuration entry:

```
JvmCommandLine="c:/jdk1.3.1/bin/java.exe" -classpath $CLASSPATH $STARTUPCLASS
```

lc

Retrieves the value of a localization string based on the current user's locale.

Type and Usage

- "Global Functions" on page 3-11
- "Localization" on page 4-13

Parameters

Takes one required parameter and any number of optional parameters:

- The first parameter is the localization key (such as *apLabelHelp* or *wwMyString*).
- The optional parameters are expressions that are resolved according to arguments inside curly braces in the localized string (for example, `{1}`).

Output

- Returns the value of the localization string for the current user's locale.
- Returns the string ID (such as *wwMyString*) if the value is not found.

Example

Retrieves the options for metadata search operators on a search page:

```
<select name="op" >
    <option value="Contains"><$lc("wwContains")$>
    <option value="Matches"><$lc("wwMatches")$>
    <option value="Starts"><$lc("wwStarts")$>
    <option value="Ends"><$lc("wwEnds")$>
    <option selected value="Substring"><$lc("wwSubstring")$>
</select>
```

Sets the subject line for a *workflow started* notification e-mail. If the name of the workflow (*dWfName*) is *Marketing*, the resulting value in English is *Workflow 'Marketing' has been started*.

```
<@dynamichtml wf_started_subject@>
    <$lc("wwWfStarted", dWfName)$>
<@end@>
```

lcCaption

Function that wraps a string into a caption. This will usually place a colon to the right of the string. For right-to-left reading languages, such as Hebrew and Arabic, the colon is placed on the left of the string.

Type and Usage

- "Global Functions" on page 3-11
- "Localization" on page 4-13

Parameters

Takes one required parameter:

- The string to be wrapped.
- optional parameters are expressions that are resolved according to arguments inside curly braces in the localized string (for example, `{1}`).

loadDocMetaDefinition

Loads the *DocMetaDefinition* database table into the active data as a ResultSet.

After the *DocMetaDefinition* database table is loaded, it can be looped on.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

None.

Output

None.

Example

Loads the *DocMetaDefinition* table into the active data as a ResultSet.

```
<$loadDocMetaDefinition()$>
```

loadDocumentProfile

Loads a content profile as specified by the trigger value for page presentation. This is called during the presentation of Search, Check In, Info, and Update pages. This function is called on request and the page on which it is called determines the action (for example, search, or info).

This information is dependent on context. The *isCheckin*, *isUpdate*, *isQuery*, or *isInfo* variables are set in local data and direct the `loadDocumentProfile` function to the page to be loaded and presented.

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

None.

Output

None.

Example

```
<$loadDocumentProfile()$>
```

See Also

- "utLoadDocumentProfiles" on page 8-355

loadSearchOperatorTables

Loads mappings between the full set of operator names and the search syntax. Used on the Search page.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

None

Output

Returns expressions associated with operator names (search operator map).

Example

```
<$loadSearchOperatorTables()$>
```

loadUserMetaDefinition

This function loads the custom user metadata definition. This exists in the `UserMetaDefinition` table which is placed on the page as a ResultSet. This is used on LDAP administration pages to help administers map user metadata to the Content Server.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

None.

LocaleArchiverDateFormat

Specifies the naming format for Archiver batch file subdirectories. Often used when exporting and importing with different versions of Content Server.

Default is *yy-MMM-dd_HH.mm.ss_SSS*. For example, *03-feb-04_15.04.14_174*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Localization" on page 4-13
- "Archiver" on page 4-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LocaleArchiverDateFormat=dd-mm-yy
```

localPageType

This function returns the page type for a page in the library folders. This may be a directory or a query page. This function is usually used by the Layout Manager API. When `Trays` is the selected layout it is used to construct a tree view of the library's pages.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Parameters

Takes one parameter, the link data used to construct the tree view.

LogMessagePrefix

Adds a prefix to log entries to designate which Content Server is creating the entry.

Typically used in a master/proxy environment.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Debugging" on page 4-9

Location

DomainJHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
LogMessagePrefix=Master
```

LogName

Specifies the file name of the log file created by the Content Server Analyzer utility.

This log file will be created in the directory specified by the IdcAnalyzeLogDir setting.

Default is *IdcAnalyze.log*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9
- "Content Server" on page 4-4

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
LogName=contentserver1_analyze.log
```

See Also

- "IdcAnalyzeLogDir" on page 8-142
- "SafeDir" on page 8-283
- "TempDir" on page 8-322

MacSupportsSignedApplets

Enables the Upload and Download Applets for Macintosh clients.

Used only when Macintosh clients are used.

When set to TRUE, Macintosh clients can use the Upload and Download Applets for checking multiple files in and out.

When set to FALSE, Macintosh clients cannot use the Upload and Download Applets.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Clients" on page 4-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MacSupportsSignedApplets=true
```

MaxArchiveErrorsAllowed

Sets the number of errors after which the Archiver stops importing or exporting. Default is 50.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxArchiveErrorsAllowed=10
```

MaxErrorsAllowed

Sets the number of errors after which the Batch Loader stops processing records from the batch load file.

If you plan to run the Batch Loader with a large number of content items overnight, then increase the default value. If you monitor the Batch Loader with a small amount of content items, then decrease the default value.

Default is 50.

Returns value as integer.

Type and Usage

- "Configuration Variables" on page 3-2
- "Batch Loader" on page 4-2

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
MaxErrorsAllowed=50
```

maxLength

Sets the maximum number of characters allowed in a metadata field.

Type and Usage

- "Common Field Display Variables" on page 3-16
- "Field Display" on page 4-11

Output

Returns the maximum length of the field.

Example

Sets the maximum length of the field to 100 characters:

```
<$maxLength=100$>
```

Specifies a custom field length based on the length of the current field:

```
<$customFieldLength = getValue("#active", fieldName & ":maxLength")$>
```

MSIE

Checks whether the client browser is Microsoft Internet Explorer.

Type and Usage

- "Dynamic Variables" on page 3-10
- "Clients" on page 4-2

Output

- Returns TRUE if the client browser is Internet Explorer.
- Returns FALSE if the client browser is not Internet Explorer.

Example

Redirects to a different URL if the browser is anything other than Internet Explorer:

```
<$if not MSIE$>
    <a href="<$redirect$>">
<$endif$>
```

MultiUpload

Enables the multiple file Upload Applet.

When set to TRUE, the Upload Applet is enabled so that multiple files can be zipped and checked in as a single content item.

When set to FALSE, the Upload Applet is disabled.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Clients" on page 4-2

Location

- System Properties, Options tab, Enable upload applet
- Admin Server, General Configuration, Enable upload applet
- *IntradocDir*/config/config.cfg

Example

As a configuration setting, enables the Upload Applet:

```
MultiUpload=true
```

As script, evaluates the condition of the Upload Applet:

```
<$MultiUpload$>
```

See Also

- "DownloadApplet" on page 8-63
- "UploadApplet" on page 8-334

noInfoColumn

Defines whether to display the Info column on search results pages.

When set to TRUE, a column containing links to Content Information pages is displayed on the standard search results page.

When set to FALSE, Info links are not displayed on the standard search results page.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3
- "Page Display" on page 4-14

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
noInfoColumn=1
```

See Also

- "showDocActions" on page 8-292
- "showSearchResultsCheckoutColumn" on page 8-294
- "noRevisionsColumn" on page 8-219

noMCPrefill

Specifies that the MultiCheckin component should not prefill metadata fields.

This variable can be used on special pages such as custom HCST and Folders component pages.

When set to TRUE, the MultiCheckin component will not prefill metadata fields.

When set to FALSE, the MultiCheckin component will prefill metadata fields.

Type and Usage

- "Page Display Variables" on page 3-15
- "Page Display" on page 4-14

Output

- Returns TRUE if the MultiCheckin component will not prefill metadata fields.
- Returns FALSE if the MultiCheckin component will prefill metadata fields.

Example

Set near the top of the page:

```
noMCPrefill=1
```

Used as script:

```
<$noMCPrefill$>
```

noRevisionsColumn

Defines whether to display the Revisions column on search results pages.

When set to TRUE, a column containing links to Revision History pages is displayed on the standard search results page.

When set to FALSE, Revision History links are not displayed on the standard search results page.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3
- "Page Display" on page 4-14

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
noRevisionsColumn=1
```

See Also

- "showDocActions" on page 8-292
- "showSearchResultsCheckoutColumn" on page 8-294
- "noInfoColumn" on page 8-217

NotationForRequiredFields

Formats how a required metadata field is displayed.

When set, the required field appears with the indicated string in front of the metadata field.

The string is included in a cascading style sheet.

If `StyleForRequiredFields` is defined, `NotationforRequiredFields` will be formatted in the style defined by `StyleForRequiredFields`.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Parameters

The only parameter is the string.

Location

IntradocDir/config/config.cfg

Example

To set an asterisk in front of a required field:

```
NotationForRequiredFields=*
```

See Also

- "StyleForRequiredFields" on page 8-316

OdbcDateFormat

Used in Content Server version 6.x to provide backward compatibility for importation of Content Server 7.x archives.

Content Server 7.x includes a millisecond data format that was not present in earlier versions. When the 6.x Archiver attempts to import the 7.x archives, the millisecond field causes an error.

Type and Usage

- "Configuration Variables" on page 3-2
- "Archiver" on page 4-2

Example

Perform the following steps to add this entry to the 6.x config.cfg file:

1. Copy the 6.x config.cfg file to another name (to save it as a backup).
2. Edit the config.cfg file in a text editor, adding the following setting:

```
OdbcDateFormat='{ts' "yyyy-MM-dd HH:mm:ss{.SSS}"'}'
```

3. Restart the server.

optionListKey

Specifies the name of a ResultSet column that contains option list values.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the values of the option list ResultSet column.

Example

Specifies *dFormat* as the column in the *DocFormats* ResultSet to get option list values from for the standard *Format* field on a checkin page:

```
<@dynamichtml std_override_format_field@>
<$if not isInfo and IsOverrideFormat$>
<$fieldIsOptionList = 1, optionListResultSet = "DocFormats", optionListKey =
"dFormat", optionListValueInclude = "std_override_format_option_value",
addEmptyOption = 1, emptyOptionLabel = lc("wwEmptyFormatOption")$>
<$include std_display_field$>
<$endif$>
<@end@>7
```

See Also

- "optionListResultSet" on page 8-224

optionListName

Specifies the name of an option list.

For standard metadata fields, this is the name of the internal option list (see Internal Option Lists).

For custom metadata fields, this is the name of the field with a suffix of `.options`.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the option list name.

Example

Sets the option list name to *docAuthors* if the field is not restricted to a single user:

```
<$if SingleUser$>
    <$isInfoOnly = 1$>
  <$else$>
    <$fieldIsOptionList = 1, optionListName = "docAuthors"$>
    <$if HasExternalUsers$><$fieldOptionListType= "combo"$>
    <$endif$>
<$endif$>
```

Defines the default option list script:

```
<$if optionsAllowPreselect and fieldValue$>
    <$defaultOptionListScript = "<$optList " & optionListName & ":fieldValue$>"$>
<$else$>
    <$defaultOptionListScript = "<$optList " & optionListName & ":noselected$>"$>
<$endif$>
```

See Also

- "defaultOptionListScript" on page 8-54
- "fieldValue" on page 8-89
- "optionListScript" on page 8-225
- "optList" on page 8-228
- "fieldIsOptionList" on page 8-84
- "fieldOptionListType" on page 8-87

optionListResultSet

Specifies the name of a ResultSet that contains option list values.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the option list ResultSet.

Example

Specifies *DocFormats* as the option list ResultSet for the standard *Format* field on a search page:

```
<@dynamichtml std_format_fields@>
<$if ClientControlled or dFormat or dExtension$>
<$fieldName = "dFormat", fieldCaption = lc("wwNativeFormat"),optionListResultSet =
"DocFormats"$> <$include std_display_field$>
<$fieldName = "dExtension", fieldCaption = lc("wwNativeExtension"), fieldWidth =
10$>
<$include std_display_field$>
<$endif$>
<@end@>
```

Creates an option list by looping over a ResultSet:

```
<@dynamichtml compute_resultset_option_list_script@>
    <$if not optionListKey$>
<$optionListKey = fieldName$>
    <$endif$>
    <$defaultOptionListScript = "<$loop " & optionListResultSet & "$>" &
    "<$inc('std_resultset_option_list_item')$>" & "<$endloop$>"$>
<@end@>
```

See Also

- "optionListKey" on page 8-222
- "defaultOptionListScript" on page 8-54
- "optionListValueInclude" on page 8-226

optionListScript

When this variable is set to a non-empty value, the eval function is used when displaying the option list for the field. This variable allows the standard implementation of option lists (defined by the defaultOptionListScript variable) to be overridden.

Type and Usage

- "Common Field Display Variables" on page 3-16
- "Field Display" on page 4-11

Output

- Returns TRUE if the value is nonempty.
- Returns FALSE if the value is an empty string.

Example

Defines a custom script for creation of an option list:

```
<$customOptionListScript = getValue("#active", fieldName & ":optionListScript")$>
```

See Also

- "defaultOptionListScript" on page 8-54

optionListValueInclude

Specifies an include that defines the values for an option list.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

Returns the include code.

Example

Defines *std_override_format_option_value* as the option list value include:

```
<@dynamichtml std_override_format_field@>
<$if not isInfo and IsOverrideFormat$>
<$fieldIsOptionList = 1, optionListResultSet = "DocFormats", optionListKey =
"dFormat", optionListValueInclude = "std_override_format_option_value",
addEmptyOption = 1, emptyOptionLabel = lc("wwEmptyFormatOption")$>
<$include std_display_field$>
<$endif$>
<@end@>
<@dynamichtml std_override_format_option_value@>
<$dDescription$>
<@end@>
```

Specifies the include to use to display options in an option list from a ResultSet:

```
<@dynamichtml std_resultset_option_list_item@>
<$curValue = getValue("#active", optionListKey)$>
<option value="<$curValue$>" <$if optionsAllowPreselect and strEquals(curValue,
fieldValue)$>selected<$endif$>>
<$if optionListValueInclude$>
<$inc(optionListValueInclude)$>
<$else$>
<$curValue$>
<$endif$>
<@end@>
```

See Also

- "optionListResultSet" on page 8-224

optionsAllowPreselect

Specifies that the metadata field option list can be prefilled with its last value.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Output

- Returns TRUE if the option list can be prefilled.
- Returns FALSE if the option list cannot be prefilled.

Example

Specifies that the option list can be prefilled:

```
<$optionsAllowPreselect=1$>
```

Determines if the option list will be prefilled:

```
<$if optionsAllowPreselect and fieldValue$>
<$defaultOptionListScript = "<$optList " & optionListName & ":fieldValue$>"$>
<$else$>
<$defaultOptionListScript = "<$optList " & optionListName & ":noselected$>"$>
<$endif$>
```

optList

Generates an option list.

This function is used extensively to create option lists on Content Server pages.

This function only produces output when used with a service that calls `loadMetaOptionsList`

Type and Usage

- "Global Functions" on page 3-11
- "Field Display" on page 4-11

Parameters

Takes one parameter and one optional argument to the parameter:

The only parameter is a field name, option list key, or a variable.

- **Field name** syntax is `<$optList fieldName$>`. For custom metadata fields, the field name will resolve to `<$optList xFieldName.options$>`, or you can specify the `.options` suffix directly.
- **Option list key** syntax is `<$optList ListName$>`.
  - For standard metadata fields, the *ListName* is the name of the internal option list (such as *docAuthors*). See Internal Option Lists.
  - For custom metadata fields, the *ListName* is the name of the option list key, such as *FieldNameList*.
- **Variable** syntax is `<$optList variable$>`. The variable must resolve to a field name or an option list key.
- One of the following optional arguments can be added to the parameter:
  - **:noselected**-No values are selected when the option list is displayed.
  - **:fieldValue**-The value specified by the *fieldValue* variable is selected as the default value in the option list.

Output

Returns a list of values.

Example

This script generates a list of possible authors from the internal *docAuthors* list:

```
<$optList docAuthors$>
```

This script generates a list of the options specified in the *xRegion* custom metadata field:

```
<$optList xRegion.options$>
```

This script generates an option list from the variable *optionListName* and specifies the default value:

```
<$if optionsAllowPreselect and fieldValue$>
    <$defaultOptionListScript = "<$optList " & optionListName & ":fieldValue$>"$>
```

```
<$else$>
    <$defaultOptionListScript = "<$optList " & optionListName & ":noselected$>"$>
<$endif$>
```

See Also

- "Option Lists" on page 2-20
- "defaultOptionListScript" on page 8-54
- "optionListName" on page 8-223
- "optionListScript" on page 8-225
- "optionsAllowPreselect" on page 8-227
- "rsMakeFromList" on page 8-267

PageParent

Checks whether a directory page in the Library has a parent page.

Type and Usage

- "Value Variables" on page 3-21
- "Page Display" on page 4-14

Output

- Returns TRUE if the directory page is a child (subfolder) of another directory page.
- Returns FALSE if the directory page is not a child (subfolder).

Example

Checks if the directory page is a subfolder:

```
<$PageParent$>
```

parseDataEntryDate

Parses a date but uses the failover logic for using the alternate parsing formats.

Dates convert to milliseconds when used with standard comparison operators. For example, the expression `(60*60*1000)` equals one hour.

A common usage of this function is to adjust the current time using a multiplication expression that represents a number of seconds, minutes, hours, or days.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is an expression to be parsed.

Output

Returns a Java date object, which is converted to a string for display.

Example

```
<$parseDataEntryDate$>
```

parseDate

Parses a date/time to allow evaluation and arithmetic.

Dates convert to milliseconds when used with standard comparison operators. For example, the expression `(60*60*1000)` equals one hour.

A common usage of this function is to adjust the current time using a multiplication expression that represents a number of seconds, minutes, hours, or days.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

The only parameter is an expression to be parsed.

Output

Returns a Java date object, which is converted to a string for display.

Example

Each of the following expressions returns the date and time one day in the past:

```
<$parseDate(dateCurrent(-1))$>
<$parseDate(dateCurrent()-(24*60*60*1000))$>
<$dateCurrent(-1)$>
<$dateCurrent()-(24*60*60*1000)$>
```

Returns the time one hour in the future. The first line adds one hour using a time multiplication expression, assigns that time and date to a custom variable, and suppresses the output. The second line references the custom variable and defines that only the time is displayed:

```
<$exec my_customParseTime parseDate(dateCurrent()+(1000*60*60))$>
<$formatTimeOnly(my_customParseTime)$>
```

Returns the date one year in the future. The first line adds one year using a time multiplication expression, assigns that time and date to a custom variable, and suppresses the output. The second line references the custom variable and defines that only the date in long format is displayed:

```
<$exec my_customParseTime parseDate(dateCurrent()+(1000*60*60*24*365))$>
<$formatTimeOnly(my_customParseTime)$>
```

This script evaluates whether the date seven days in the future is greater than the expiration date and returns a message to the user if true:

```
<$if dOutDate$>
    <$if dateCurrent(7) > parseDate(dOutDate)$>
Content item expires in one week.
     <$endif$>
<$endif$>
```

This script uses `parseDate` within a conditional statement for customized workflow jumps. The script specifies that if the last time we entered this step was four days ago, go to the first step in workflow *wf_late* and set the return step to be the next step:

```
<$if parseDate(wfCurrentGet("lastEntryTs")) < dateCurrent(-4)$>
    <$wfSet("wfJumpName", "lateJump")$>
    <$wfSet("wfJumpTargetStep", "step_1@wf_late")$>
    <$wfSet("wfJumpReturnStep", wfCurrentStep(1))$>
    <$wfSet("wfJumpEntryNotifyOff", "0")$>
<$endif$>
```

See Also

- "dateCurrent" on page 8-31

parseDateWithPattern

Parses a date/time to a specified date/time pattern.

Dates convert to milliseconds when used with standard comparison operators. For example, the expression `(60*60*1000)` equals one hour.

A common usage of this function is to adjust the current time using a multiplication expression that represents a number of seconds, minutes, hours, or days.

Type and Usage

- "Global Functions" on page 3-11
- "Date and Time" on page 4-8

Parameters

Takes two parameters:

- The first parameter is a date string used by the Content Server, or a date object created with the parseDate or dateCurrent functions.
- The second parameter is the date/time pattern, which is a standard Java *SimpleDateFormat* pattern string, such as MM/dd/yyyy.

Output

Returns a Java date object, which is converted to a string for display.

Example

Displays the current date and time in the format specified by the pattern (for example, Wed, 4 Jul 2001 12:08:56 -0700):

```
<$parseDateWithPattern(<$dateCurrent()$>,"EEE, d MMM yyyy HH:mm:ss Z")$>
```

See Also

- "parseDate" on page 8-232
- "dateCurrent" on page 8-31
- "formatDateWithPattern" on page 8-102

pneNavigation

Enables the left sidebar navigation.

Type and Usage

- "Global Functions" on page 3-11
- "Page Display" on page 4-14

Parameters

Set as a name/value pair:

```
pneNavigation=1
```

Default value is 1 (enabled).

To disable this function, it must be set to a null string.

Output

Evaluates to TRUE or FALSE.

Example

Enables the sidebar navigation:

```
<$pneNavigation=1$>
```

To force the sidebar navigation off, set it to a null string:

```
<$pneNavigation=""$>
```

Setting the definition to other than 1 or a null string value is invalid and will not disable the sidebar navigation:

```
<$pneNavigation=0$>
```

ProviderDir

Defines the path to the /providers directory.

Default is *DomainHome*/ucm/cs/data/providers.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
ProviderDir=c:/domain/data/providers
```

RequestAuditAdditionalVerboseFieldsList

Used with verbose tracing when the Request Audit trace is activated.

The Request Audit trace appears on the System Audit Information Page. This trace creates two separate summaries calculated over two different time intervals. The trace calculates the number of service requests and length of time to perform the requests. After a summary is reported for the specified interval the values are reset and the cumulative totals and averages are computed again for the next interval period.

This configuration variable is a comma-delimited list of additional fields to display when reporting information about the request.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
RequestAuditAdditionalVerboseFieldsList=xComment,dDocType
```

See Also

- "RequestAuditIntervalSeconds1" on page 8-238
- "RequestAuditIntervalSeconds2" on page 8-239
- "RequestAuditListDepth1" on page 8-240
- "RequestAuditListDepth2" on page 8-241
- "RequestAuditSubClassificationField" on page 8-242

RequestAuditIntervalSeconds1

Used to specify the short audit interval for the Request Audit trace.

The Request Audit trace appears on the System Audit Information Page. This trace creates two separate summaries calculated over two different time intervals. The trace calculates the number of service requests and length of time to perform the requests. After a summary is reported for the specified interval the values are reset and the cumulative totals and averages are computed again for the next interval period.

Specify the value in seconds.

The default value is 120 seconds (2 minutes).

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
RequestAuditIntervalSeconds1=240
```

See Also

- "RequestAuditAdditionalVerboseFieldsList" on page 8-237
- "RequestAuditIntervalSeconds2" on page 8-239
- "RequestAuditListDepth1" on page 8-240
- "RequestAuditListDepth2" on page 8-241
- "RequestAuditSubClassificationField" on page 8-242

RequestAuditIntervalSeconds2

Used to specify the long audit interval for the Request Audit trace.

The Request Audit trace appears on the System Audit Information Page. This trace creates two separate summaries calculated over two different time intervals. The trace calculates the number of service requests and length of time to perform the requests. After a summary is reported for the specified interval period the values are reset and the cumulative totals and averages are computed again for the next interval period.

Specify the value in seconds.

The default value is 3600 seconds (1 hour).

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
RequestAuditIntervalSeconds2=7200
```

See Also

- "RequestAuditAdditionalVerboseFieldsList" on page 8-237
- "RequestAuditIntervalSeconds1" on page 8-238
- "RequestAuditListDepth1" on page 8-240
- "RequestAuditListDepth2" on page 8-241
- "RequestAuditSubClassificationField" on page 8-242

RequestAuditListDepth1

Used to specify the number of services to present in the Request Audit trace.

This configuration value applies only to the short audit period.

The Request Audit trace appears on the System Audit Information Page. This trace creates two separate summaries calculated over two different time intervals. The trace calculates the number of service requests and length of time to perform the requests.

This variable is used to specify the number of services to present from a sorted list of requests. The list is sorted in descending order by elapsed time consumed.

The default value is 5.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
RequestAuditListDepth1=7
```

See Also

- "RequestAuditAdditionalVerboseFieldsList" on page 8-237
- "RequestAuditIntervalSeconds1" on page 8-238
- "RequestAuditIntervalSeconds2" on page 8-239
- "RequestAuditListDepth2" on page 8-241
- "RequestAuditSubClassificationField" on page 8-242

RequestAuditListDepth2

Used to specify the number of services to present in the Request Audit trace.

This configuration value applies only to the long audit period.

The Request Audit trace appears on the System Audit Information Page. This trace creates two separate summaries calculated over two different time intervals. The trace calculates the number of requests and length of time to perform the requests.

This variable is used to specify the number of services to present. The list is sorted in descending order by elapsed time consumed.

The default value is 20.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
RequestAuditListDepth2=25
```

See Also

- "RequestAuditAdditionalVerboseFieldsList" on page 8-237
- "RequestAuditIntervalSeconds1" on page 8-238
- "RequestAuditIntervalSeconds2" on page 8-239
- "RequestAuditListDepth1" on page 8-240
- "RequestAuditSubClassificationField" on page 8-242

RequestAuditSubClassificationField

Used to specify an internal field, usually a metadata field for documents, whose value is appended to the service name in order to differentiate the name in the Request Audit list.

The Request Audit trace appears on the System Audit Information Page. This trace creates two separate summaries calculated over two different time intervals. The trace calculates the number of service requests and length of time to perform the requests.

In some cases, the service name does not create an adequate breakdown of the different types of requests. This entry helps to differentiate the types of entries.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
RequestAuditSubClassificationField=xDepartment
```

See Also

- "RequestAuditAdditionalVerboseFieldsList" on page 8-237
- "RequestAuditIntervalSeconds1" on page 8-238
- "RequestAuditIntervalSeconds2" on page 8-239
- "RequestAuditListDepth1" on page 8-240
- "RequestAuditListDepth2" on page 8-241

requiredMsg

Specifies the error message to be displayed if a required metadata field does not have a value upon checkin.

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the error message as a string.

Example

Defines the required field error message for the *dDocTitle* metadata field as the *wwSpecifyTitle* localized string:

```
<$fieldName = "dDocTitle", fieldCaption = lc("wwTitle"), isRequired = 1, fieldType
= "BigText", requiredMsg = lc("wwSpecifyTitle")$>
```

See Also

- "isRequired" on page 8-184
- *Getting Started with the SDK*

rptDisplayMapValue

This function is meant for internal use only.

This function returns a localized string representation of an internal key code. These key codes are used to store status and state flags for content items. This function is used on the Content Information page and workflow pages to display descriptions of the internal state.

Type and Usage

- "Global Functions" on page 3-11
- "Localization" on page 4-13

Parameters

Takes two parameters:

- The name of the table where the keys are stored.
- The key to be localized.

rs

This function returns a result set given the name of the result set. Because the actual object it returns is a result set, certain actions can be performed on the return value that cannot be performed on the return values of other functions. In particular, the loop syntax can be applied to the result.

Type and Usage

-

Parameters

This function has one parameter, `resultSetName`, which is the name of the result set to return.

Output

A result set object if found, otherwise null.

Example

Suppose MyResultSet is the name of a result set, then you could loop on the result set using the following syntax:

```
<$loop rs("MyResultSet")$>
... Script inside loop ...
<$endloop$>
```

This function also can be used when temporarily assigning variables when calling Idoc script functions. In particular, the following example will temporarily assign the variable 'rsParam' to point to the same result set as pointed to by MyResultSet for the duration of the call to include the resource include *my_include*. If the result of the rs function is assigned to a variable, then that variable will have a shared pointer to the result set creating the same effect as if rsCreateReference were called.

```
<$inc("my_include", rsParam=rs("MyResultSet"))$>
```

rsAddFields

Adds new fields to a ResultSet. This function will only add the field if it is not already present. Note that setValue(...) can also add new fields but it only works on ResultSets that are nonempty and are on a currently valid row.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of the ResultSet to get new fields (columns) added.
- The second parameter is a comma separated list of fields to add.(the column names to be added).

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsAddFields(rsName, fieldsList)$>
```

rsAddFieldsWithDefaults

This function adds new fields to an existing result set. Unlike rsAddFields, this function provides the ability to specify default values for any fields that are added. Note that rsAddFieldsWithDefaults never over-writes any data in fields that already exist; it only adds new fields to a result set.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

The following table lists parameters for the function.

Parameters	Description
`resultSetName`	The name of the result set.
`newFields`	A comma-separated list of fields to append to the result set. Any specified fields that already exist in the result set are ignored.
`defaultValues`	An optional comma-separated list of default values that are set for the new fields in each row. Each value in this list corresponds to the new field in the same spot of the list newFields. If you need to set a default value with a comma in it, you can use '^' as a comma. If the defaultValues list is longer or shorter than newFields list, then the function ignores the extra default values or uses the empty string for unspecified default values, respectively.

Output

Nothing.

Example

Adding fields

```
<$rsCreateResultSet("MyList", "field1,field2,field3")$>
<$rsAppendRowValues("MyList", "A,B,C,D,E,F")$>
<$rsAddFieldsWithDefaults("MyList", "field4,field5,field6")$>
```

In this basic example, we create a result set MyList, fill it with some values, then add three more fields to MyList, without setting default values. The resulting table will look like the following:

field1	field2	field3	field4	field5	field6
A	B	C			
D	E	F			

Adding fields with default

```
<$rsCreateResultSet("MyList", "field1,field2,field3")$>
<$rsAppendRowValues("MyList", "A,B,C,D,E,F")$>
<$rsAddFieldsWithDefaults("MyList", "field4,field5,field6", "X,Y,Z")$>
```

Here we define default values for the new fields. The resulting table will look like the following:

field1	field2	field3	field4	field5	field6
A	B	C	X	Y	Z
D	E	F	X	Y	Z

When a field already exists

```
<$rsCreateResultSet("MyList", "field1,field2,field3")$>
<$rsAppendRowValues("MyList", "A,B,C,D,E,F")$>
<$rsAddFieldsWithDefaults("MyList", "field1,field4,field5", "X,Y,Z")$>
```

In this case, we try and add a field that already exists in MyList. This action is completely ignored, and old field values are preserved. The resulting table will look like the following:

field1	field2	field3	field4	field5
A	B	C	X	Y
D	E	F	X	Y

Fewer default values specified

```
<$rsCreateResultSet("MyList", "field1,field2,field3")$>
<$rsAppendRowValues("MyList", "A,B,C,D,E,F")$>
<$rsAddFieldsWithDefaults("MyList", "field4,field5,field6", "X,Y")$>
```

Notice in this example how there are not enough default values in the default value list for all fields. In this case, it just fills in blanks for the unspecified column.

field1	field2	field3	field4	field5	field6
A	B	C	X	Y	
D	E	F	X	Y	

See Also

- "rsAddFields" on page 8-246

rsAddRowCountColumn

This function adds a new field whose value for each row is the row count for that row. The count starts at 0. This function is useful to use just before using `rsSort(...)` on XML data island ResultSets (inside .hcsp files) so that the original row location can be preserved.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of ResultSet to be modified so that it has an new row count column.
- The second parameter is the name of the field that will hold the row count.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsAddRowCountColumn(rsName, countFieldName)$>
```

rsAppend

This function is similar in nature to rsMerge, except there are no merge test criteria. All rows from the ResultSet rsNameSource are appended to the ResultSet rsNameTarget. In addition, all fields in rsNameSource not in rsNameTarget will be added to rsNameTarget. Useful for doing a simple combining of ResultSets.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the ResultSet that will be modified by the merge
- The second parameter is the ResultSet containing the rows that will be appended to the target.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsAppend("targetRsetName", "sourceRsetName")$>
```

See Also

- "rsMerge" on page 8-271

rsAppendNewRow

Appends a new row to the end of the ResultSet. The current row is not affected.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

The only parameter is the name of the ResultSet to receive the new row.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails or the ResultSet is empty.

Example

```
<$rsAppendNewRow("SearchResults")$>
```

rsAppendRowvalues

A utility function for adding new rows to a ResultSet. The values list is a comma-delimited list of values (using the escape rule of ',' -> '^', '^' -> '#^', '#' -> '##' to encode each value) split up into rows. Assume *n*fields is the number of fields in the ResultSet. The values list is split up into rows by counting *n*fields values to create a new row and then taking the next *n*fields values to create each following row and so on. If the number of values is not an exact multiple of *n*fields then the last row has its fields values padded out with empty strings. This function is useful for compactly hard coding ResultSets using Idoc script.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameters the name of the ResultSet that is to get new rows.
- The second parameter is the new values to add.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rowValues = "a1, b1, a2, b2"$>
<$rsCreateResultSet("ResultSet1", "ColumnA, ColumnB")$>
<$rsAppendRowvalues("ResultSet1", rowValues$>
<table border=2>
    <tr><td>ColumnA</td><td>ColumnB</td></tr>
    <$loop ResultSet1$>
    <tr><td><$ColumnA$></td><td><$ColumnB$></td>
</tr>
<$endloop$>
</table>
```

The resulting HTML would look like the following.

Column A	Column B
Column A	Column B
A1	B1
A2	B2

rsCopyFiltered

This function copies only selected rows from one ResultSet to create another ResultSet. Any prior ResultSet with name `rsNameTarget` will be replaced. The rows are selected by testing each row's value of filterField using a test against the pattern in `filterPattern`. The pattern match is the same as used in the Idoc script `like` operator.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes four parameters:

- The first parameter is the ResultSet that is providing the rows to copy.
- The second parameter is the new ResultSet created by the copy.
- The third parameter is the name of the field being tested.
- The fourth parameter is the pattern match to apply to see if the row should be copied.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsCopyFiltered(rsNameSource, rsNameTarget, filterField, filterPattern)$>
```

rsCreateReference

This function sets an Idoc Script variable with a shared reference to a pre-existing result set in the request data.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

This function has the following parameters:

Parameters	Description
`sourceResultSet`	The name of a pre-existing result set.
`targetResultSet`	The name of the variable that will share a reference to precisely the same result set.

Output

If successful, returns True.

Example

The current value of <$MyResultSet.field2$> and <$MyResultSetAlias.field2$> will both be the letter *E*.

```
<$rsCreateResultSet("MyResultSet", "field1,field2,field3")$>
<$rsAppendRowValues("MyResultSet", "A,B,C,D,E,F")$>
<$rsCreateReference("MyResultSet", "MyResultSetAlias")$>
<$rsNext("MyResultSetAlias")$>
```

rsCreateResultSet

Creates a ResultSet initialized with the list of fields in *fieldList*. All the fields will be set as `string` type.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of the ResultSet to create.
- The second parameter is a comma separated list of fields.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsCreateResultSet(rsName, fieldList)$>
```

rsDeleteRow

Deletes the current row in the ResultSet. The current row is advanced to the next row (or points to after the end of the ResultSet if the end row in the ResultSet is deleted).

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

The only parameter is the name of the ResultSet whose current row is deleted.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails or the ResultSet is empty.

Example

```
<$rsDeleteRow("Folders")$>
```

rsDocInfoRowAllowDisclosure

Checks if the current user is allowed to view the URL of the content item referenced by the current row of a ResultSet.

This function is useful for selectively showing the URLs of a ResultSet generated by a content item query.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15
- "Internal Security" on page 4-18

Parameters

The only parameter is the name of a ResultSet whose current row contains information about a content item.

Output

- Returns TRUE if the user is allowed to view the URL of the content item.
- Returns FALSE if the user is not allowed view the URL or the ResultSet is empty.

Example

Checks if the user can view the URL of the content item in the current row of *resultSet1*.

```
<$rsDocInfoRowAllowDisclosure("resultSet1")$>
```

rsExists

Checks to see if a result set exists.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes one parameter, the name of the result set.

Output

- Returns TRUE if the result set is found.
- Returns FALSE if the result set does not exist or if it is empty.

Example

```
<$if rsExists("resultSet1")$>
    code to execute if TRUE
<$endif$>
```

rsFieldByIndex

Retrieves the name of the field at a specified column index, starting from zero.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The name of the ResultSet.
- The index number.

Example

```
<$rsFieldName = rsFieldByIndex("resultSet1", "2")$>
```

rsFieldExists

Checks to see if the named result set contains the specific field.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The name of the result set to be searched.
- The name of the field to be found.

Output

- Returns TRUE if the field is found.
- Returns FALSE if the field does not exist.

Example

```
<$if rsFieldExists("resultSet1", "FieldName1")$>
    code to execute if TRUE
<$endif$>
```

rsFindRowPrimary

Searches the first column of a ResultSet for a matching value.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of a ResultSet whose first column is to be searched.
- The second parameter is the value to be searched for.

Output

- Returns TRUE if the specified value is found in the first column of the specified ResultSet.
- Returns FALSE if the specified value is not found.

Example

Searches the first column of *resultSet1* until a value matching *value1* is found.

```
<$rsFindRowPrimary("resultSet1","value1")$>
```

rsFirst

Moves to the first row in a ResultSet.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

The only parameter is the name of a ResultSet.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails or the ResultSet is empty.

Example

Advances the *SearchResults* ResultSet to the first row:

```
<$exec rsFirst("SearchResults")$>
```

rsInsertNewRow

Inserts a row just before the current row of the ResultSet being manipulated. The new row then becomes the current row.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

The only parameter is the name of the ResultSet in which to have a new row inserted.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsInsertNewRow("SearchResults")$>
```

rsIsRowPresent

Checks the status of the current row in a result set.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes one parameter, the name of the result set.

Output

- Returns TRUE if the result set is currently on a valid row.
- Returns FALSE if not on a valid row.

Example

```
<$if rsIsRowPresent("resultSet1")$>
    code to execute if TRUE
<$endif$>
```

rsLoopInclude

This function loops on a result set, executing a resource include once for each row. During the execution of this include, the result set is temporarily made active (as is done with the standard `<$loop ...$>` construction).

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

This function has the following parameters:

Parameters	Description
`ResultSetName`	The name of the result set to loop over.
`includeName`	The name of the resource include to execute.

Output

The combined output of the various iterations of the resource include.

Example

```
<$rsLoopInclude(resultSetName, includeName)$>
```

rsLoopSingleRowInclude

This function temporarily promotes a result set to be active while executing a single resource include. It functions much like rsLoopInclude, however the include is only executed once for the current row of the result set.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

This function has the following parameters:

Parameters	Description
`ResultSetName`	The name of the result set to make active.
`includeName`	The name of the resource include to execute.

Output

The output of the resource include.

Example

```
<$rsLoopSingleRowInclude(resultSetName, includeName)$>
```

See Also

- "rsLoopInclude" on page 8-265

rsMakeFromList

Creates a single-column ResultSet from an option list.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two required parameters and one optional parameter:

- The first parameter is the name of the ResultSet to be created.
- The second parameter is the name of an existing option list (for example, *docAccounts* or *xMyCustomField.options*).
- The third optional parameter is the name of the ResultSet column. If no value is specified, the column name defaults to *row*.

Output

Creates a ResultSet with one column, populated by the values in the specified option list.

Example

Creates two ResultSets called *ListA* and *ListB* from the *securityGroups* and *docAuthors* option lists:

```
<$lista = "securityGroups"$>
<$rsMakeFromList("ListA", lista)$>
<$rsMakeFromList("ListB", "docAuthors", "name")$>
<table border=2>
    <tr><td>Security Groups</td><td>Authors</td></tr>
    <tr>
        <td><$loop ListA$><$row$><br>
        <$endloop$></td>
        <td><$loop ListB$><$name$><br>
        <$endloop$></td>
    </tr>
</table>
```

The ResultSets would look like the following.

```
@ResultSet ListA
1
row
Public
Secure
@end
@ResultSet ListB
1
name
hchang
pkelly
sysadmin
user1
```

```
@end
```

The resulting HTML would look like the following.

ListA	ListB
Security Groups	Authors
	hchang
Public	okelly
Secure	sysadmin
	user1

See Also

- "rsMakeFromString" on page 8-269
- "Option Lists" on page 2-20
- "optList" on page 8-228

rsMakeFromString

Creates a single-column ResultSet from a string.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two required parameters and one optional parameter:

- The first parameter is the name of the ResultSet to be created.
- The second parameter is a comma-delimited list of strings to be parsed (such as a,b,c,d), or a variable that has a comma-delimited string as its value.
- The third optional parameter is the name of the ResultSet column. If no value is specified, the column name defaults to `row`.

Output

Creates a ResultSet with one column, populated by the specified string values.

Example

Creates two ResultSets called *StringA* and *StringB*:

```
<$stringa = "a,b,c,d"$>
<$rsMakeFromString("StringA", stringa)$>
<$rsMakeFromString("StringB", "A,B,C,D", "name")$>
<table border=2>
    <tr><td>StringA</td><td>StringB</td></tr>
    <tr>
        <td><$loop StringA$><$row$><br>
        <$endloop$></td>
        <td><$loop StringB$><$name$><br>
        <$endloop$></td>
    </tr>
</table>
```

The ResultSets would look like the following.

```
@ResultSet StringA
1
row
a
b
c
d
@end
@ResultSet StringB
1
name
A
B
C
D
```

```
@end
```

The resulting HTML would look like the following.

Stringa	Stringb
A	A
B	B
C	C
D	D

See Also

- "rsMakeFromList" on page 8-267

rsMerge

Merges the rows of one ResultSet into another. If the value from the `rsNameSource` ResultSet in the `rsCommonField` field matches the value for the same field in a row for `rsNameTarget`, then that row will be replaced. Otherwise, the row from rsNameSource will be appended. If there are multiple rows in `rsNameTarget` that are matched by the same value, then only the first row is replaced and it is replaced by the last row in `rsNameSource` that has a matching row. Any fields in the ResultSet `rsNameSource` that are not in `rsNameTarget` are added as new fields to `rsNameTarget`. This function is best performed on ResultSets that have only unique values in the `rsCommonField` field so that issues with multiple matches are avoided.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes three parameters:

- The first parameter (rsNameTarget) is the ResultSet that will be modified by the merge.
- The second parameter (rsNameSource) is the ResultSet containing the rows that will be merged into the target.
- The third parameter (rsCommonField) is the field that occurs in both results sets that will be used as the basis of the merge.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsMerge(rsNameTarget, rsNameSource, commonField$>
```

See Also

- "rsAppend" on page 8-250

rsMergeDelete

Analogous to `rsMergeReplaceOnly`, except matching rows in the ResultSet `rsNameTarget` are removed instead of replaced. It is useful for removing rows from one ResultSet that already occur in another.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes three parameters:

- The first parameter (rsNameTarget) is the ResultSet that will be modified by the merge.
- The second parameter (rsNameSource) is the ResultSet containing the rows that will be merged into the target.
- The third parameter (rsCommonField) is the field that occurs in both results sets that will be used as the test for which rows to remove from `rsNameTarget`.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsMergeDelete(rsNameTarget, rsNameSource, rsCommonField)$>
```

rsMergeReplaceOnly

Similar to `rsMerge`, but rows are only replaced; none are appended to the ResultSet `rsNameTarget`. In addition, new fields from `rsNameSource` are not added to `rsNameTarget`.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes three parameters:

- The first parameter (rsNameTarget) is the ResultSet that will be modified by the merge.
- The second parameter (rsNameSource) is the ResultSet containing the rows that will be merged into the target.
- The third parameter (rsCommonField) is the field that occurs in both results sets that will be used as the basis of the merge

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$rsMergeReplaceOnly(rsNameTarget, rsNameSource, rsCommonField) $>
```

rsNext

Moves to the next row in a ResultSet.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

The only parameter is the name of a ResultSet.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails or the ResultSet is empty.

Example

Advances the *SearchResults* ResultSet to the next row:

```
<$exec rsNext("SearchResults")$>
```

rsNumFields

Provides a count of the number of fields in a result set. This is useful when combined with rsFieldByIndex.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes one parameter, the name of the result set.

Output

Returns the number of fields.

Example

```
<$numFields = rsNumFields("resultSet1")$>
```

See Also

- "rsFieldByIndex" on page 8-259

rsNumRows

Provides a count of the number of rows in a result set.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes one parameter, the name of the result set.

Output

Returns the number of rows.

Example

```
<$numRows = rsNumRows("resultSet1")$>
```

rsRemove

Removes a result set.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes one parameter, the name of the result set.

Example

```
<$rsRemove("resultSet1")$>
```

rsRename

Renames a ResultSet.

If you use a variable to define the new ResultSet name, you will need to loop over the new ResultSet variable manually using *rsFirst*, *rsNext*, and *#isRowPresent*. For example:

```
<$rsNewName="MySearchResults"$>
<$rsRename("SearchResults", rsNewName)$>
    <$rsFirst(rsNewName)$>
    <$loopwhile getValue(rsNewName, "#isRowPresent")$>
        <!--output code-->
        <$rsNext(rsNewName)$>
    <$endloop$>
```

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of an existing ResultSet.
- The second parameter is the new ResultSet name. The call will fail if the resultset already exists.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

Renames the `SearchResults` ResultSet to `MySearchResults`:

```
<$rsRename("SearchResults", "MySearchResults")$>
```

rsRenameField

Renames a field in the ResultSet.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes three parameters:

- The first parameter is the name of the ResultSet whose field is being renamed.
- The second parameter is the current name of the field.
- The third parameter is the new name of the field.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

Renames the 'invoice' field to 'outlays' in the warehouse ResultSet.

```
<$exec rsRenameField("warehouse", "invoice", "outlays")$>
```

rsSetRow

Moves to a specified row in a ResultSet.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of the ResultSet.
- The second parameter is the number of the row.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails or the ResultSet is empty.

Example

Advances the *SearchResults* ResultSet to the 10th row:

```
<$exec rsSetRow("SearchResults",10)$>
```

rsSort

Sorts a ResultSet by a particular column.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes four parameters. For example:

```
rsSort(rsName,sortCol,sortType,sortOrder)
```

Parameter	Description
rsName	The name of the ResultSet.
sortCol	The name of the column to sort by.
sortType	The type of sort (defaults to *int*): **string**: sort alphabetically **int**: sort numerically **date**: sort by date
sortOrder	The sort order (defaults to *asc*): **asc**: ascending order **desc**: descending order

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails or the ResultSet is empty.

Example

Sorts the *SearchResults* ResultSet by *Content ID* in descending order:

```
<$rsSort("SearchResults","dDocName","string","desc")$>
```

rsSortTree

Sorts a ResultSet that is a representation of an XML data structure (with nodes, parent nodes, and depth attributes).

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes seven parameters. For example:

```
rsSortTree(rsName,itemIdCol,parentIdCol,nestLevelCol,sortCol,sortType,sortOrder)
```

Parameter	Description
rsName	The name of the ResultSet.
itemIdCol	The name of the column that contains the unique ID for each node.
parentIdCol	The name of the column that contains the ID for the parent node, if one exists.
nestLevelCol	The name of the column that contains the nest level (depth) for each node.
sortCol	The name of the column to sort by.
sortType	The type of sort (defaults to *int*): **string**: sort alphabetically **int:** sort numerically **date**: sort by date
sortOrder	The sort order (defaults to *asc*): **asc**: ascending order **desc**: descending order

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails or the ResultSet is empty.

Example

Sorts the *discussionPosts* ResultSet by the *itemNumber* column in ascending order:

```
<$rsSortTree("discussionPosts", "discussionPosts!itemNumber",
"discussionPosts!parentNumber", "dpItemNestLevel", "discussionPosts!itemNumber",
"int", "asc")$>
```

SafeDir

Retrieves the location where extra Content Server files are moved by the Content Server Analyzer utility.

Default is a subdirectory in the log directory specified by the IdcAnalyzeLogDir setting.

This variable is not settable.

Type and Usage

- "Other Read-Only Variables" on page 3-19
- "Debugging" on page 4-9
- "Directories and Paths" on page 4-9

Example

As script, returns the value of the configuration setting:

```
<$SafeDir$>
```

See Also

- "IdcAnalyzeLogDir" on page 8-142

ScriptDebugTrace

Enables a trace of all includes and calls to the Idoc Script eval function.

The contents of the *eval* function and any dynamically assigned includes are also shown as part of the trace. The trace is indented by one + character per nested level of *include* or *eval* call. The trace also shows any error messages (without the nested location information) and the output of any calls to the Idoc Script function trace.

Can be used as a setting in a configuration file or as a parameter to a service call.

Type and Usage

- "Setable Variables" on page 3-19
- "Debugging" on page 4-9

Output

When set to TRUE, performs a trace of all includes and calls to the Idoc Script function *eval*, and displays the results of the trace on the page.

Example

As script, returns the value of the configuration setting:

```
<$ScriptDebugTrace$>
```

When included in a Content Server URL, all debug trace information is added to the bottom of the New Checkin page:

```
http://<home>/idcplg/idc_cgi_isapi-instance.dll?IdcService=CHECKIN_NEW_
FORM&ScriptDebugTrace=true
```

See Also

- "eval" on page 8-71
- "ScriptErrorTrace" on page 8-285
- "setResourceInclude" on page 8-290
- "trace" on page 8-328

ScriptErrorTrace

Enables a trace of script errors.

Can be used as a setting in a configuration file or as a parameter to a service call.

Type and Usage

- "Setable Variables" on page 3-19
- "Debugging" on page 4-9

Output

If set to TRUE, script error information is added to the bottom of the displayed page. If no errors are detected, the message *No Errors* is displayed.

Example

As script, returns the value of the configuration setting:

```
<$ScriptErrorTrace$>
```

When included in a Content Server URL, error trace information is added to the bottom of the New Checkin page:

```
http://<home>/idcplg/idc_cgi_isapi-instance.dll?IdcService=CHECKIN_NEW_
FORM&ScriptErrorTrace=true
```

See Also

- "ScriptDebugTrace" on page 8-284
- "setResourceInclude" on page 8-290
- "trace" on page 8-328

setContentType

Sets the content type for the returned page so the browser renders it properly. For example, a content type (file-format type or MIME type) of text/plain, application/excel, or text/xml.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

The only parameter is the content type.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$setContentType("text/plain")$>
```

setExpires

This function can be used to set an absolute time for an expiration of a page. This is best used for pages with a long life that may be requested frequently. Due to problems in time synchronization between the client and the server, it is not totally reliable, especially for short-lived caches. The data must follow one of the acceptable HTTP date formats.

Refer to the W3 Protocols for more information:

`http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.3`

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is an absolute date, formatted in the manner HTTP requires, after which this page should be refreshed

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

This will cause the page to expire at the given absolute time. Note that all time stamps must be in the GMT time zone.

```
<$setExpires("Sat, 02 Aug 2003 24:00:00 GMT")$>
```

This is a way to use other IdocScript functions to set the expiration date to 14 days from the current date.

```
<$date=formatDateWithPattern(dateCurrent(14), "EEE, dd MMM yyyy")$>
<$setExpires(date & " 24:00:00 GMT"))$> >
```

See Also

- "setHttpHeader" on page 8-288

setHttpHeader

This function can be used to set any HTTP header. Values include, but are not limited to, 'Cache-Control', 'Content-Type', 'Last-Modified', or any of the other acceptable HTTP headers.

Refer to the W3 Protocols for more information:

`http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.3`

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

Takes two parameters. For example:

```
setHttpHeader(headerName, headerValue)
```

Parameter	Description
headerName	The name of a valid HTTP header.
headerValue	The value for the header.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

This sets the content-type to XML, to enable your browser to render it with the passed XSLT.

```
<$setHttpHeader("Content-Type", "text/xml")$>
```

See Also

- "setMaxAge" on page 8-289

setMaxAge

This function can be used to set an absolute time for an expiration of a page. This is best used for pages with a long life that may be requested frequently. Due to problems in time synchronization between the client and the server, it is not totally reliable, especially for short-lived caches. The data must follow one of the acceptable HTTP date formats.

Refer to the W3 Protocols for more information:

`http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.3`

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

The only parameter is the lifetime of this page in seconds.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

This will expire the content of the page immediately:

```
<$setMaxAge(0)$>
```

This will set the cache lifetime for this page to 10 minutes, and tell the browser that it absolutely must try to refresh the page after that time is up:

```
<$setMaxAge(600)$>
<$setHttpHeader("Cache-Control", "must-revalidate")$>
```

See Also

- "setHttpHeader" on page 8-288

setResourceInclude

Defines an include.

This function allows dynamically constructed script to be assigned to an include (much as the eval function enables such script to be parsed and evaluated).

If the specified include already exists, the *super* keyword can be used in the new script to refer to the existing include.

Type and Usage

- "Global Functions" on page 3-11
- "Idoc Script" on page 4-12

Parameters

Takes two parameters:

- The first parameter is the name of the include.
- The second parameter is a string that defines the contents of the include. This string can contain Idoc Script.

Output

Returns 0 if the include could not be parsed, 1 if it can be parsed.

Use `exec` to suppress this behavior.

Example

Uses the string "My name is *resource include*" to dynamically construct script:

```
<$setResourceInclude("my_name","My name is <$my_name$>")$>
```

Uses the *super* keyword to modify the *std_display_field* include:

```
<$myInclude="<span class='field'><$include super.std_display_field$></span>"
<$setResourceInclude("std_display_field",myInclude)$>
```

The following example suppresses the returned result:

```
<$exec setResourceInclude("std_display_field",myInclude)$>
```

setValue

The target can either be `#local` or the name of a ResultSet. If the target is `#local`, then the parameter *name* identifies the name of a local data variable whose value is to be set by the parameter *value*. If the target area is nonempty and is not `#local`, then a field in a ResultSet named by the parameter *target* is being set. If the ResultSet exists and is currently on a valid current row, then that particular column's value (column named by the parameter *name*) on that row will be set with the value in the parameter *value*. If the field is currently not a part of that ResultSet, then the field will be added as a new column to the ResultSet and the value will be set. If the target ResultSet is not on a valid row, then this function will have no effect (but an internal exception will be reported on the server output). Note that this function should be contrasted with getValue.

Type and Usage

- "Global Functions" on page 3-11
- "ResultSets" on page 4-15

Parameters

Takes three parameters:

- The first parameter is the target area to be set (either #local or the name of a ResultSet)
- The second parameter is the name of key that holds the value to be set (either a local variable or a ResultSet field)
- The third parameter is the value to be set.

Output

- Returns TRUE if the function is successful.
- Returns FALSE if the function fails.

Example

```
<$setValue(target, name, value)$>
```

showDocActions

Defines whether to display checkout links on the search results page.

This variable is also used to evaluate additional code for ODMA applications, such as Oracle Desktop. To display checkout links for a Content Server that is not accessed by ODMA clients, use the showSearchResultsCheckoutColumn variable.

- When set to TRUE, a column containing checkout links is displayed on the search results page.
- When set to FALSE, checkout links are not displayed on the search results page.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3
- "Page Display" on page 4-14
- "Clients" on page 4-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
showDocActions=1
```

See Also

- "showSearchResultsCheckoutColumn" on page 8-294
- "noInfoColumn" on page 8-217
- "noRevisionsColumn" on page 8-219

ShowRevisionsColumn

In later versions of the Content Server, the revision number of a document is not displayed on the Search Results page while using Classic view. This configuration variable is used to display the revision number.

- When set to TRUE, the revision number of a document is displayed on the Search Results page while using Classic View.
- This setting works in Classic search mode only.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Field Display" on page 4-11

Location

- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
ShowRevisionsColumn=false
```

showSearchResultsCheckoutColumn

Defines whether to display checkout links on the standard search results page.

- When set to TRUE, a column containing checkout links is displayed on the standard search results page.
- When set to FALSE, checkout links are not displayed on the standard search results page.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3
- "Page Display" on page 4-14

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
showSearchResultsCheckoutColumn=1
```

See Also

- "showDocActions" on page 8-292
- "noInfoColumn" on page 8-217
- "noRevisionsColumn" on page 8-219

StdPageWidth

Sets the page width in pixels.

Default is 550.

Type and Usage

- "Configuration Variables" on page 3-2
- "Page Display" on page 4-14

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
StdPageWidth=1000
```

Returns the page width as a string:

```
<$StdPageWidth$>
```

strCenterPad

Pads equal space on both sides of a string. An equal number of spaces is added to each side of the string to make it at least the specified length. A character will be added to the length of the string if required.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is the string.
- The second parameter is the string length.

Output

Returns a string with spaces on both sides.

Example

Pads equal space on each side and creates a string seven characters long, using the form `<space><space>inf<space><space>`:

```
<$strCenterPad("inf", 7)$>
```

Pads equal space on each side and creates a string nine characters long, using the form `<space><space><space>inf<space><space><space>`:

```
<$strCenterPad("inf", 8)$>
```

Pads equal space on each side and creates a string nine characters long, using the form `<space><space><space>inf<space><space><space>`:

```
<$strCenterPad("inf", 9)$>
```

strCommaAppendNoDuplicates

This function appends a new token to an existing string. It returns the string plus a comma plus the new token. If the token already exists in the string, it is not added.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is the string.
- The second parameter is the token.

Example

```
<$myString = strCommaAppendNoDuplicates(myString, "A")$>
<$myString = strCommaAppendNoDuplicates(myString, "B")$>
<$myString = strCommaAppendNoDuplicates(myString, "C")$>
<$myString = strCommaAppendNoDuplicates(myString, "A")$>
<$myString = strCommaAppendNoDuplicates(myString, "B")$>
```

strConfine

Confines a string to a maximum length and appends padding if necessary.

If the string equals or is shorter than the specified length, it is unaffected.

If the string is longer than the specified length, it is shortened and three padding characters are appended to equal the specified length. The character used for padding can be specified by changing the StrConfineOverflowChars variable.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is the string.
- The second parameter is the maximum string length.

Output

Returns a string.

Example

Confines the string and appends three padding characters (dots) to make it a string five characters long, using the form `in<dot><dot><dot>`:

```
<$strConfine("inform", 5)$>
```

Confines the string and appends padding characters (dots) to make it a string five characters long, using the form `i<dot><dot><dot><dot>`:

```
<$strConfine("i", 5)$>
```

Confines the string to a maximum of ten characters in length, so this six-character string is unaffected:

```
<$strConfine("inform", 10)$>
```

Confines the string to a maximum of six characters in length, so this six-character string is unaffected:

```
<$strConfine("inform", 6)$>
```

See Also

- "StrConfineOverflowChars" on page 8-299

StrConfineOverflowChars

Defines a string padding character.

Defines the character used for padding by the strConfine function.

Default is a dot.

Type and Usage

- "Configuration Variables" on page 3-2
- "Strings" on page 4-19

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
StrConfineOverflowChars=.
```

See Also

- "strConfine" on page 8-298

strEquals

Checks if two strings are equal, including case.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is a string.
- The second parameter is the string to be compared to the first parameter.

Output

- Returns TRUE if the strings are equal.
- Returns FALSE if strings are not equal.

Example

Evaluates whether the strings are equal and returns TRUE (1):

```
<$strEquals("Home","Home")$>
```

Evaluates whether the strings are equal and returns FALSE (0):

```
<$strEquals("home","Home")$>
```

See Also

- "regexMatches" on page 6-201
- "regexReplaceAll" on page 6-202
- "regexReplaceFirst" on page 6-203
- "strEqualsIgnoreCase" on page 8-301

strEqualsIgnoreCase

Checks if two strings are equal, not including case.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is a string.
- The second parameter is the string to be compared to the first parameter.

Output

- Returns TRUE if the strings are equal.
- Returns FALSE if strings are not equal.

Example

Evaluates whether the strings are equal and returns TRUE (1):

```
<$strEqualsIgnoreCase("home","Home")$>
```

Evaluates whether the strings are equal and returns FALSE (0):

```
<$strEqualsIgnoreCase("home","page")$>
```

See Also

- "strEquals" on page 8-300

strGenerateRandom

This function generates a random string of hexadecimal characters. It outputs as many characters as specified by the `length` parameter. If length is not specified, it defaults to 16 characters.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes the length of the string as a parameter.

strIndexOf

Determines if one string is a substring of another.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is a string.
- The second parameter is the possible substring.

Output

- If the second string is a substring of the first, returns an index value that indicates where in the first string the substring begins. The first character has an index value of 0.
- If the second string is not a substring of the first, returns a value of -1.

Example

Evaluates whether `xy` is a substring of `xyz` and returns the index value 0:

```
<$if strIndexOf("xyz","xy") >=0$> check for substring <$endif$>
```

Evaluates whether `yz` is a substring of `xyz` and returns the index value 1:

```
<$if strIndexOf("xyz","yz") >=0$> check for substring <$endif$>
```

Evaluates whether `ab` is a substring of `xyz` and returns the index value –1 to indicate that this is not a substring:

```
<$if strIndexOf("xyz","ab") >=0$> check for substring <$endif$>
```

strLeftFill

Fills the left side of a string with characters to make it a specified length.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes three parameters:

- The first parameter is the string.
- The second parameter is the fill character.
- The third parameter is the length.

Output

Returns a string, left-filled with the specified character if necessary,.

Example

Formats the string *sleep* by left-filling with the character *Z* to ten spaces. This returns the string *ZZZZZsleep*:

```
<$strLeftFill("sleep",'Z',10)$>
```

Returns the string *sleep*:

```
<$strLeftFill("sleep",'Z',4)$>
```

strLeftPad

Pads extra space to the left of a string to make it a specified length.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is the string.
- The second parameter is the length.

Output

Returns a string, left-filled with spaces if necessary.

Example

Pads extra space on the left to make it a string five characters long, using the form `<space><space>inf`:

```
<$strLeftPad("inf", 5)$>
```

Returns the string *information*:

```
<$strLeftPad("information", 5)$>
```

strLength

Evaluates the length of a string.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns an integer value.

Example

Evaluates the length of the string home and returns the integer 4:

```
<$strLength("home")$>
```

strLower

Formats a string in all lowercase letters.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string in all lowercase letters.

Example

Evaluates the string `Home` and returns *home*.

```
<$strLower("Home")$>
```

strRemoveWs

Removes empty spaces from a string.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string with no spaces.

Example

Formats the string `h o m e` as the string *home*:

```
<$strRemoveWs("h o m e")$>
```

strReplace

Replaces an existing substring with another substring.

If there are multiple occurrences of the substring to be replaced, they will all be replaced by the new substring.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes three parameters:

- The first parameter is the string on which the substitution will be performed.
- The second parameter is the substring to be replaced.
- The third parameter is the substring that will replace the existing substring.

Output

Returns a string.

Example

Replaces the word *classified* with *restricted* and results in the string *This document is restricted*.

```
<$strReplace("This document is classified.","classified","restricted")$>
```

Replaces the slashes in the date with periods, giving a date in the form `6.20.2001`:

```
<$strReplace(formatDateOnly(dateCurrent()),"/",".")$>
```

strReplaceIgnoreCase

Replaces an existing substring with another substring, performing the search without using case sensitivity.

If there are multiple occurrences of the substring to be replaced, they will all be replaced by the new substring.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes three parameters:

- The first parameter is the string on which the substitution will be performed.
- The second parameter is the substring to be replaced.
- The third parameter is the substring that will replace the existing substring.

Output

Returns a string.

Example

```
<$hello = "Hello world! I love the world!"$>
<$strReplaceIgnoreCase(hello, "WORLD", "universe")$>
```

strRightFill

Fills the right side of a string with characters to make it a specified length.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes three parameters:

- The first parameter is the string.
- The second parameter is the fill character.
- The third parameter is the length.

Output

Returns a string, right-filled with the specified character if necessary,.

Example

Formats the string *sleep* by right filling with the character Z to ten spaces. This returns the string *sleep*ZZZZZ:

```
<$strRightFill("sleep",'Z',10)$>
```

Returns the string *sleep*:

```
<$strRightFill("sleep",'Z',4)$>
```

strRightPad

Pads extra space to the right of a string to make it a specified length.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes two parameters:

- The first parameter is the string.
- The second parameter is the length.

Output

Returns a string, right-filled with spaces if necessary.

Example

Pads extra space on the right to make it a string five characters long, using the form `inf<space><space>`:

```
<$strRightPad("inf", 5)$>
```

Returns the string *information*:

```
<$strRightPad("information", 5)$>
```

strSubstring

Retrieves a substring from a string.

The first character has an index value of 0.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

Takes either two or three parameters:

- The first parameter is the string.
- The second parameter is a number representing the start index.
- The third optional parameter is a number representing the stop index.

Output

Returns a substring.

- If the stop index parameter is specified, the substring starting with the character at the start index and ending one character before the stop index is returned.
- If the stop index parameter is not specified, the substring from the start index to the end of the string is returned.

Example

Retrieves the first two characters of the string *my*:

```
<$strSubstring("mystring",0,2)$>
```

Retrieves the string after the second character *string*:

```
<$strSubstring("mystring",2)$>
```

See Also

- "regexReplaceAll" on page 6-202
- "regexReplaceFirst" on page 6-203

strTrimWs

Removes spaces from the beginning and end of a string.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string with no leading or trailing spaces.

Example

Formats the string " *homepage* " as the string *homepage*:

```
<$strTrimWs(" homepage ")$>
```

strUpper

Formats a string in all uppercase letters.

> **Note:** Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string in all uppercase letters.

Example

Evaluates the string *Home* and returns *HOME*.

```
<$strUpper("Home")$>
```

StyleForRequiredFields

Formats how a required metadata field is displayed.

When set, the required field is displayed in the style as defined in a cascading style sheet.

Values for the style can be any of the field caption styles defined for the Content Server.

Type and Usage

- "Other Field Display Variables" on page 3-17
- "Field Display" on page 4-11

Parameters

The only parameter is the specified style.

Location

IntradocDir/config/config.cfg

Example

The following example displays a required field in red:

```
StyleForRequiredFields=requiredField
```

See Also

- "fieldCaptionStyle" on page 8-78
- "NotationForRequiredFields" on page 8-220

SubjectsDir

Defines the path to the /subjects directory.

Subjects are subsystems that are notified when a Content Server service makes a change in the system, such as modifying an option list or adding a new user.

Default is *DomainHome*/ucm/cs/data/subjects/.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
SubjectsDir=c:/domain_temp/data/subjects
```

SysAdminAddress

Defines the system administrator e-mail address.

This is the address used in workflow and subscription notification e-mails that come from the Content Server.

Returns a string.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Users" on page 4-20
- "Content Server" on page 4-4

Location

- System Properties, Internet tab, Administrator Mail Address
- Admin Server, Internet Configuration, Administrator Mail Address
- *IntradocDir*/config/config.cfg

Example

As a configuration setting:

```
SysAdminAddress=admin@companyname.com
```

As script, returns the value of the configuration setting:

```
<$SysAdminAddress$>
```

See Also

- "MailServer" on page 6-162
- "SmtpPort" on page 6-244

SystemDateFormat

Sets the date/time format for the Content Server.

Note that there are two types of date formats: one that stores seconds and one that truncates seconds and stores the date up to the minute. If seconds are used, dates can only be stored up to the year 2037.

If SystemLocale is not specified as a configuration setting, the *SystemDateFormat* will be deduced directly from the OS settings in the Java VM instead of using the Content Server configuration table settings for the *SystemLocale*. If *SystemLocale* is explicitly defined, then the date/time format from the Content Server configuration tables for that locale will be used instead.

The syntax for this format is an extension of the date format functionality provided in the standard Java class libraries.

Type and Usage

- "Configuration Variables" on page 3-2
- "Date and Time" on page 4-8
- "Localization" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry to set the standard US date time using the four digit year format:

```
SystemDateFormat=M/d{/yyyy} {h:mm[:ss] {aa}[zzz]}
```

See Also

- "SystemLocale" on page 8-320
- "SystemTimeZone" on page 8-321

SystemLocale

Sets the system locale for the Content Server.

This setting controls the following:

- localized strings used in the Content Server interface
- character encoding for Web pages
- date/time format
- time zone
- parameters used for full-text indexing

Any of these individual features can be controlled by separate configuration entries or by changing the settings for the *SystemLocale*.

This setting can be changed on the Server tab of the System Properties utility. The defaults for a specific locale can be changed on the Localization tab of the System Properties utility.

If *SystemLocale* is not specified as a configuration setting, the SystemDateFormat will be deduced directly from the OS settings in the Java VM instead of using the Content Server configuration table settings for the *SystemLocale*. If *SystemLocale* is explicitly defined, then the date/time format from the Content Server configuration tables for that locale will be used instead.

Type and Usage

- "Configuration Variables" on page 3-2
- "Localization" on page 4-13

Location

- System Properties, Server tab, System Locale
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
SystemLocale=Deutsch
```

See Also

- "SystemDateFormat" on page 8-319
- "SystemTimeZone" on page 8-321

SystemTimeZone

Sets the default time zone for the Content Server.

The *SystemTimeZones* table in the *ucm_home_dir*/shared/config/resources/std_locale.htm file lists the time zones that can be specified.

Default is the time zone set for the local computer's operating system. If the default is used, the variable does not appear in the configuration file.

Type and Usage

- "Configuration Variables" on page 3-2
- "Date and Time" on page 4-8
- "Localization" on page 4-13

Location

- System Properties, Server tab, System Timezone
- *IntradocDir*/config/config.cfg

Example

Used as a configuration entry:

```
SystemTimeZone=Europe/Berlin
```

See Also

- "SystemDateFormat" on page 8-319
- "SystemLocale" on page 8-320

TempDir

Specifies the location where the Content Server Analyzer utility duplicates and tests files.

Default is a temp subdirectory in the log directory specified by the IdcAnalyzeLogDir setting.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9
- "Directories and Paths" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
TempDir=f:/domain/analyzer_logs/temp/
```

See Also

- "IdcAnalyzeLogDir" on page 8-142

TemplateClass

Retrieves the classification of the template for the current page.

This variable makes it possible to create conditional content in a template based on the identity of the template. Use this variable within a template page to determine the source of the pages delivered by the server.

For standard templates, this variable is defined in the *class* column of the *IntradocTemplates* table in the *templates.hda* file.

For search results templates, this variable evaluates to `Results`.

For report templates, this variable evaluates to `Reports`.

For dynamic server pages, this variable is typically `IdcDynamicFile`.

Type and Usage

- "Template Read-Only Variables" on page 3-18
- "Page Display" on page 4-14
- "Templates" on page 4-20

Output

Returns a string.

Example

This markup displays a table of template information on the page:

```
<TABLE>
    <TR><TD>Template Name</TD>
    <TD><$TemplateName$></TD></TR>
    <TR><TD>Template Class</TD>
    <TD><$TemplateClass$></TD></TR>
    <TR><TD>Template Type</TD>
    <TD><$TemplateType$></TD></TR>
    <TR><TD>Template File Path</TD>
    <TD><$TemplateFilePath$></TD></TR>
</TABLE>
```

See Also

- "TemplateFilePath" on page 8-324
- "TemplateName" on page 8-325
- "TemplateType" on page 8-326

TemplateFilePath

Retrieves the path from where the template was actually loaded.

This variable makes it possible to create conditional content in a template based on the identity of the template. Use this variable within a template page to determine the source of the pages delivered by the server.

Type and Usage

- "Template Read-Only Variables" on page 3-18
- "Page Display" on page 4-14
- "Templates" on page 4-20

Output

Returns the path as a string.

Example

This markup displays a table of template information on the page:

```
<TABLE>
    <TR><TD>Template Name</TD>
    <TD><$TemplateName$></TD></TR>
    <TR><TD>Template Class</TD>
    <TD><$TemplateClass$></TD></TR>
    <TR><TD>Template Type</TD>
    <TD><$TemplateType$></TD></TR>
    <TR><TD>Template File Path</TD>
    <TD><$TemplateFilePath$></TD></TR>
</TABLE>
```

See Also

- "TemplateClass" on page 8-323
- "TemplateName" on page 8-325
- "TemplateType" on page 8-326

TemplateName

Retrieves the internal name of the template for the current page. For example, DOC_INFO or CHECKIN_NEW_FORM.

This variable makes it possible to create conditional content in a template based on the identity of the template. Use this variable within a template page to determine the source of the pages delivered by the server.

Type and Usage

- "Template Read-Only Variables" on page 3-18
- "Page Display" on page 4-14
- "Templates" on page 4-20

Output

Returns the template name as a string.

Example

This markup displays a table of template information on the page:

```
<TABLE>
    <TR><TD>Template Name</TD>
    <TD><$TemplateName$></TD></TR>
    <TR><TD>Template Class</TD>
    <TD><$TemplateClass$></TD></TR>
    <TR><TD>Template Type</TD>
    <TD><$TemplateType$></TD></TR>
    <TR><TD>Template File Path</TD>
    <TD><$TemplateFilePath$></TD></TR>
</TABLE>
```

See Also

- "TemplateClass" on page 8-323
- "TemplateFilePath" on page 8-324
- "TemplateType" on page 8-326

TemplateType

Provides the template type for the current page.

This variable makes it possible to create conditional content in a template based on the identity of the template. Use this variable within a template page to determine the source of the pages delivered by the server.

For standard templates, this variable is defined in the *formtype* column of the *IntradocTemplates* table in the *templates.hda* file.

For search results templates, this variable is defined in the *formtype* column of the *SearchResultTemplates* table in the *templates.hda* file.

For report templates, this variable is defined in the *datasource* column of the *IntradocReports* table in the *reports.hda* file.

For dynamic server pages, this variable is either `hcsp` or `hcsf`.

Type and Usage

- "Template Read-Only Variables" on page 3-18
- "Page Display" on page 4-14
- "Templates" on page 4-20

Output

Returns the template type as a string.

Example

This markup displays a table of template information on the page:

```
<TABLE>
    <TR><TD>Template Name</TD>
    <TD><$TemplateName$></TD></TR>
    <TR><TD>Template Class</TD>
    <TD><$TemplateClass$></TD></TR>
    <TR><TD>Template Type</TD>
    <TD><$TemplateType$></TD></TR>
    <TR><TD>Template File Path</TD>
    <TD><$TemplateFilePath$></TD></TR>
</TABLE>
```

See Also

- "TemplateClass" on page 8-323
- "TemplateFilePath" on page 8-324
- "TemplateName" on page 8-325

toInteger

Converts a string to an integer.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns an integer value. If the string does not evaluate to a number, or evaluates to a non-integer number, an error is thrown.

Example

Converts the string 4 to an integer and returns the value 4:

```
<$toInteger("4")$>
```

trace

Enables logging a debug or trace message to the ScriptDebugTrace output. A message can also be output to the console or to the system logs.

Type and Usage

- "Global Functions" on page 3-11
- "Debugging" on page 4-9

Parameters

Takes one required and two optional parameters:

- The first parameter is the message which is output. The value can be one of the following:
  - A string. If the string is passed as the only parameter, the ScriptDebugTrace variable must be set for this function to execute. The string is added to the accumulated debug trace.
  - `#local`, which prepares a dump of all current local variables.
  - `#all`, which prepares a full HDA dump of all local data, result sets, and environment data.
- The second (optional) parameter is the location where the message will be relayed. The values can be the following:
  - `#console`, to display to a console
  - `#log` to log a message in the HTML log files
  - the name of a variable (such as StatusMessage). In that case, the message is appended to the current value.
- The third (optional) parameter is for the appropriate tracing section and is only relevant if the location parameter is set to `#console`.

Example

The following example logs the string `message` to the system console, which is always logged:

```
<$trace("message", "#console")$>
```

The following example logs the string `message` to the system console in the `pagecreation` tracing section.

```
<$trace("message", "#console", "pagecreation")$>
```

The following example logs the string `message` to the HTML Content Server log file.

```
<$trace("message", "#log")$>
```

The following example dumps all local variables and their values to the system console.

```
<$trace("#local", "#console")$>
```

The following example dumps all local variables, result sets, and environment variables to the system console.

```
<$trace("#all", "#console")$>
```

The following example dumps all data to the variable `MyTraceDump`, which can then be displayed on the page. This is useful for HCSP developers who may not have the appropriate access rights to view the console logs.

```
<$trace("#all", "MyTraceDump")$>
<$MyTraceDump$>
```

See Also

- "ScriptDebugTrace" on page 8-284

TraceResourceConflict

When the Content Server is started from a command line, each system resource that is overridden twice by component resources is output to the command line.

When set to TRUE, system resources that are overridden by two or more component resources are output.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
TraceResourceConflict=true
```

See Also

- "TraceResourceLoad" on page 8-331
- "TraceResourceOverride" on page 8-332

TraceResourceLoad

When the Content Server is started from a command line, all resource loading activities are output to the command line.

When set to TRUE, all resources loaded, resource overrides, resource conflicts, and resource merges are output.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
TraceResourceLoad=true
```

See Also

- "TraceResourceConflict" on page 8-330
- "TraceResourceOverride" on page 8-332

TraceResourceOverride

When the Content Server is started from a command line, system resources and non-system component resources that are overridden by a component resource are output to the command line.

When set to TRUE, system resources and non-system resources that are overridden by one or more component resources are output.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
TraceResourceOverride=true
```

See Also

- "TraceResourceConflict" on page 8-330
- "TraceResourceLoad" on page 8-331

UieHideSearchCheckboxes

Records Manager only.

Controls if the metadata field check boxes on the search page are shown.

If set to TRUE, the metadata field check boxes are not shown on the page. If set to FALSE, the metadata field check boxes are shown.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

DomainHome/ucm/urm/custom/RecordsManagement/records_management_environments.cfg

DomainHome/ucm/urm/custom/RetentionManager/retentionmanager_environments.cfg

Example

Used as a configuration entry:

```
UieHideSearchCheckboxes=false
```

UploadApplet

Enables the multiple file Upload Applet.

> **Important:** This setting is only for use by custom legacy versions of the Upload Applet. It should not be enabled when the standard MultiUpload variable is enabled.

When set to TRUE, the Upload Applet is enabled so that multiple files can be zipped and checked in as a single content item.

When set to FALSE, the Upload Applet is disabled.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Clients" on page 4-2

Location

IntradocDir/config/config.cfg

Example

As a configuration setting, enables the Upload Applet:

```
UploadApplet=true
```

As script, evaluates the condition of the Upload Applet:

```
<$UploadApplet$>
```

See Also

- "DownloadApplet" on page 8-63
- "MultiUpload" on page 8-216

url

Formats a string for use in a URL.

This function converts blank spaces and reserved characters to an escape sequence.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string formatted for use in a URL.

Example

Formats the string *home page* as *home%20page*:

```
<$url("home page")$>
```

Formats the string *home/page* as *home%2fpage*:

```
<$url("home/page")$>
```

Formats the string *home?page* as *home%3fpage*:

```
<$url("home?page")$>
```

urlEscape7Bit

This function returns a URL-encoded version of the string parameter. This is similar to the `url` function but it only encodes characters that are not 7-bit clean (ASCII). Therefore this function can be called repeatedly on the same string.

If the `url` function is used to double encode a string, every % character is encoded to %25.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string formatted for use in a URL.

Example

```
<$path = "mÿ fïlë.txt"$>
20fïlë.txt = <$url(path)$>
mÿ%2520fïlë.txt = <$url(url(path))$>
m%ff%20f%efl%eb.txt = <$urlEscape7Bit(url(path))$>
m%ff%20f%efl%eb.txt= <$urlEscape7Bit(urlEscape7Bit(url(path)))$>
```

See Also

- "url" on page 8-335

UseBellevueLook

Enables an alternate look-and-feel for the Content Server interface.

When set to TRUE, the Bellevue look-and-feel (green and red color scheme) is used.

If UseXpedioLook is TRUE, it overrides this setting.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Page Display" on page 4-14

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UseBellevueLook=true
```

See Also

- "UseXpedioLook" on page 8-352

UseFourDigitYear

This string is obsolete. Use SystemDateFormat.

Specifies a four-digit or two-digit year display format.

- When set to TRUE, the Content Server displays years with four digits.
- When set to FALSE, the Content Server displays years with two digits.

The Content Server supports both two-digit and four-digit years for data entry.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Date and Time" on page 4-8

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UseFourDigitYear=false
```

See Also

- "SystemDateFormat" on page 8-319

UserAccounts

Retrieves a comma-delimited list of accounts the current user is assigned to.

- The `#none` entry indicates privileges to content items that have no account assigned.
- The `#all` entry indicates privileges to all accounts.

Type and Usage

- "User Read-Only Variables" on page 3-19
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

Returns the account list as a string.

Example

For example:

```
UserAccounts=BOS,SEA,MSP/Gen
```

This markup displays user variables on a page:

```
<$if UserName$>
    Logon Name: <$UserName$><BR>
    User Name: <$UserFullName$><BR>
    E-mail Address: <$UserAddress$><BR>
    Accounts: <$UserAccounts$><BR>
<$endif$>
```

UserAddress

Retrieves the e-mail address of the current user.

Type and Usage

- "User Read-Only Variables" on page 3-19
- "Users" on page 4-20

Output

Returns the e-mail address as a string.

Example

This markup displays user variables on a page:

```
<$if UserName$>
    Logon Name: <$UserName$><BR>
    User Name: <$UserFullName$><BR>
    E-mail Address: <$UserAddress$><BR>
    Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserAppRights

Checks the application rights of the current user.

Type and Usage

- "User Read-Only Variables" on page 3-19
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

Returns a bit flag number specifying the user's rights:

Admin Application	Bit Flag
User Admin	1
Web Layout Editor	2
Repository Manager	4
Workflow Admin	8
Configuration Manager	16
Archiver	32

Example

Displays application rights of the current user:

```
<$UserAppRights$>
```

For example, the following combinations would result in the following numbers:

- User Admin + Web Layout Editor = 3
- Repository Manager + Workflow Admin + Archiver = 44
- All six applications = 63

See Also

- "AdminAtLeastOneGroup" on page 6-5
- "UserIsAdmin" on page 6-269

UserCache

Defines the path to the *users* directory, which is the location where user personalization information is stored.

Default is *DomainHome*/ucm/cs/data/users/.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UserCache=c:/myuserdata/
```

UserCacheTimeout

Sets the timeout in milliseconds for the temporary cache of global and external user information.

Default is 60000 milliseconds (1 minute).

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Server" on page 4-4
- "Users" on page 4-20

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UserCacheTimeout=120000
```

UserDefaultAccount

Retrieves the default account for the current user.

Type and Usage

- "User Read-Only Variables" on page 3-19
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

Returns the account name as a string.

Example

This markup displays user variables on a page:

```
<$if UserName$>
    Logon Name: <$UserName$><BR>
    User Name: <$UserFullName$><BR>
    E-mail Address: <$UserAddress$><BR>
    Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UseRedirectedOutput

Enables the capture of output from the Java Virtual Machine. This includes logging output from any enabled tracing facilities and stack dumps resulting from the VM crash.

Used only on Windows systems and if the Content Server is running as a service.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Location

IntradocDir/config/config.cfg

Example

```
UseRedirectedOutput=true
```

UserFullName

The full name of the current user.

Type and Usage

- "User Read-Only Variables" on page 3-19
- "Users" on page 4-20

Output

- Returns the user's full name as a string.
- If the user is not logged in, returns the string anonymous.

Example

This markup displays user variables on a page:

```
<$if UserName$>
    Logon Name: <$UserName$><BR>
    User Name: <$UserFullName$><BR>
    E-mail Address: <$UserAddress$><BR>
    Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserLanguageID

Returns the two-letter code that represents the user's preferred language, as in `en` for English, `fr` for French, or `ja` for Japanese.

This is useful when constructing URLs to localized content.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Localization" on page 4-13

Output

Returns the code for the preferred language.

UserLocaleId

Returns the full name for a user's locale, as in `English-US`, `English-UK` or `Japanese`.

The locale contains information about language, date, and number formatting.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Localization" on page 4-13

Output

Returns the name of the preferred language.

UserName

Retrieves the username of the current user.

Type and Usage

- "User Read-Only Variables" on page 3-19
- "Users" on page 4-20

Output

- Returns the username as a string.
- If the user is not logged in, evaluates to the string anonymous.

Example

This markup displays user variables on a page:

```
<$if UserName$> Logon Name: <$UserName$><BR>
    User Name: <$UserFullName$><BR>
    E-mail Address: <$UserAddress$><BR>
    Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserPublishCacheDir

Allows the Web server plug-in to refer to data in a local directory (stub directory) rather than the file system where the Content Server is installed.

This setting is used for clustered Windows Content Server configurations. See the clustering documentation for more information.

Use the format `UserPublishCacheDir=stub_dir/data/users`, where *stub_dir* is the path of the stub directory (such as `f:/`*`domain`*`/`).

Type and Usage

- "Configuration Variables" on page 3-2

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UserPublishCacheDir=f:/domain/data/users
```

UserRoles

Retrieves a comma-delimited list of roles the current user belongs to.

Type and Usage

- "User Read-Only Variables" on page 3-19
- "Internal Security" on page 4-18
- "Users" on page 4-20

Output

Returns the user role list as a string.

Example

For example:

```
PublicContributor,ClassifiedConsumer
```

References the list of user roles:

```
<$if UserRoles$>
    <$include optional_field$>
<$endif$>
```

UseXpedioLook

Enables an alternate look-and-feel for the Content Server interface.

- When set to TRUE, the Xpedio look-and-feel (purple and blue color scheme) is used.
- If this setting is TRUE, it overrides the UseBellevueLook setting.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Page Display" on page 4-14

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UseXpedioLook=true
```

See Also

- "UseBellevueLook" on page 8-337

utGetValue

Returns the value of a user property from a specified personalization (user topic) file.

User topic files are HDA files that are located in the *IntradocDir*/data/users/profiles/*us*/*username*/ directories.

Type and Usage

- "Global Functions" on page 3-11
- "Personalization" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of a user topic HDA file.
- The second parameter is the name of a property defined in the user topic file.

Output

Returns the value of the property in the specified user topic file.

Example

Returns the value of the *emailFormat* property in the current user's *pne_portal.hda* file:

```
<$utGetValue("pne_portal","emailFormat")$>
```

See Also

- "utLoad" on page 8-354
- "utLoadResultSet" on page 8-356

utLoad

Loads a personalization (user topic) file so it is available for use by the utGetValue and utLoadResultSet functions.

User topic files are HDA files that are located in the *IntradocDir*/data/users/profiles/*us*/*username*/ directories.

Type and Usage

- "Global Functions" on page 3-11
- "Personalization" on page 4-15

Parameters

The only parameter is the name of a user topic HDA file.

Example

Loads the current user's *wf_in_queue.hda* file and makes it available to other user topic functions:

```
<$utLoad("wf_in_queue")$>
```

See Also

- "utGetValue" on page 8-353
- "utLoadResultSet" on page 8-356

utLoadDocumentProfiles

Used to retrieve information about a user's current My Check In and My Search links for content profiles.

Type and Usage

- "Global Functions" on page 3-11
- "Content Profiles" on page 4-3

Parameters

Takes two parameters:

- The user topic name.
- The name of the result set containing the content profiles available for the user on the personalization links.

Example

```
<$utLoadDocumentProfiles("pne_portal", "PneDocumentProfiles")
```

See Also

- "loadDocumentProfile" on page 8-204

utLoadResultSet

Loads a ResultSet specified in a personalization (user topic) file into the DataBinder.

User topic files are HDA files that are located in the *IntradocDir*/data/users/profiles/*us*/*username*/ directories.

Type and Usage

- "Global Functions" on page 3-11
- "Personalization" on page 4-15
- "ResultSets" on page 4-15

Parameters

Takes two parameters:

- The first parameter is the name of a user topic HDA file.
- The second parameter is the name of a ResultSet defined in the user topic file.

Output

- Returns TRUE if the ResultSet is successfully loaded into the DataBinder.
- Returns FALSE if the function fails.

Example

Loads the *SavedQueries* ResultSet from the current user's *pne_portal.hda* file into the DataBinder:

```
<$utLoadResultSet("pne_portal","SavedQueries")$>
```

See Also

- "utGetValue" on page 8-353
- "utLoad" on page 8-354

valueStyle

Specifies the style of the value for the metadata field.

The following SPAN styles are typically used for values. They are defined in the *std_style_declaration* include in the *ucm_home_dir*/shared/config/resources/std_page.htm resource file:

- tableEntry
- xxsmall
- strongHighlight

Type and Usage

- "Field Information Variables" on page 3-15
- "Field Display" on page 4-11

Output

Returns the name of the value style.

Example

Sets the value style:

```
<$if isFieldInfoOnly$>
    <$if valueStyle$>
        <$fieldValueStyle = valueStyle$>
    <$else$>
        <$fieldValueStyle = "tableEntry"$>
    <$endif$>
<$endif$>
```

Uses the *xxsmall* style for the field value when the Download applet is used:

```
<$if useDownloadApplet$>
    <$valueStyle="xxsmall", fieldValue = strTrimWs(inc("download_file_by_
    applet_form_content"))$>
<$else$>
    <$fieldValue = strTrimWs(inc("doc_file_get_copy"))$>
<$endif$>
```

See Also

- "fieldValueStyle" on page 8-90

VaultDir

Defines the path to the *vault* directory.

- This is the root directory of the Content Server native file repository.
- This setting is required only if the *vault* directory is not located in the Content Server root directory.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9
- "Inbound Refinery" on page 4-6

Location

- Content Server: *DomainHome*/ucm/cs/bin/intradoc.cfg
- Inbound Refinery: *IntradocDir*/ucm/ibr/bin/intradoc.cfg

Example

Used as a configuration entry:

```
VaultDir=f:/vault
```

WarnAboutTimeZone

Specifies whether the Content Server installer displays a warning if it cannot determine the computer's time zone.

- When set to TRUE, a default time zone is used and a warning is displayed.
- When set to FALSE, a default time zone is used, but no warning is displayed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Date and Time" on page 4-8
- "Localization" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
WarnAboutTimeZone=true
```

WebBrowserPath

Defines the path to the Web browser that displays the Content Server Help files from stand-alone applications.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

- System Properties, Paths tab, Browser Executable Path
- *DomainHome*/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
WebBrowserPath=c:/Program Files/Internet Explorer/Iexplore.exe
```

WeblayoutDir

Defines the path to the *weblayout* directory.

- This is the root directory of the Content Server Web site.
- This setting is required only if the *weblayout* directory is not located in the Content Server root directory.

There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9
- "Inbound Refinery" on page 4-6

Location

- Content Server: *DomainHome*/ucm/cs/bin/intradoc.cfg
- Inbound Refinery: *DomainHome*/ucm/ibr/bin/intradoc.cfg

Example

Used as a configuration entry:

```
WeblayoutDir=f:/weblayout
```

xml

Formats a string for use in XML.

This function replaces non-alphanumeric characters with the correct XML sequence that represents the character. For example, the ampersand "&" character is replaced with the "&amp;" sequence.

When using a double-quote character within a string, a backslash "\" must precede the double-quote to display it as a character. If the backslash is not used as an escape flag, the double-quote is interpreted as ending the string.

Type and Usage

- "Global Functions" on page 3-11
- "Strings" on page 4-19

Parameters

The only parameter is the string.

Output

Returns a string formatted for use in a URL.

Example

Escapes the ampersand and returns the XML formatted string, "Me &amp; you."

```
<$xml("Me & you.")$>
```

Escapes the non-alphanumeric characters and returns the XML-formatted string, "Test the &quot;, &gt;, and &lt; characters."

```
<$xml("Test the \", >, and < characters.")$>
```

9

Feature and Product Variables and Functions by Type and Usage

This chapter lists configuration variables used with added Content Server features and products, and also groups the variables by *usage*.

9.1 Configuration Variables

Configuration variables are predefined Idoc Script variables that are generally used as settings in configuration files. For a list of configuration variables and details about usage, see "Configuration Variables" on page 3-2.

"AllowForceDelete" on page 10-2
"AllowRetentionPeriodWithoutCutoff" on page 10-3
"AllowSecondaryConversion" on page 10-4
"AllowSkippedHyperlinktoCauseIncomplete" on page 10-5
"DefaultGraphicTimeout" on page 10-29
"DefaultNativeTimeout" on page 10-31
"DefaultPostscriptTimeout" on page 10-32
"DistillerPath" on page 10-36
"FileEncoding" on page 8-92
"FrameMakerCyclesAllowed" on page 10-42
"FrameMakerTimeout" on page 10-43
"HasSiteStudio" on page 10-44
"HideVitalReview" on page 10-45
"JvmCommandLine" on page 8-200
"LkDisableOnRebuild" on page 10-57
"LkDisableReferenceInfo" on page 10-58
"LkDisallowConversionFormats" on page 10-59
"LkIsSecureSearch" on page 10-60
"LkReExtractOnRebuild" on page 10-61
"LkRefreshBatchSize" on page 10-62
"LkRefreshErrorsPercent" on page 10-63
"LkRefreshErrorsAllowed" on page 10-64

"LkRefreshErrorThreshold" on page 10-65
"MaxNumRecursiveStepDefinitions" on page 10-67
"MSPubexePath" on page 10-68
"OpenOfficeListenerPort" on page 10-71
"OpenOfficeListenerURL" on page 10-72
"PageMakerExePath" on page 10-74
"PostConversionTimeOut" on page 10-75
"PostprocessPDFPath" on page 10-76
"PostscriptPrinterName" on page 10-77
"PowerPointPrintOutput" on page 10-78
"PreconversionPath" on page 10-79
"PreconversionWaitTime" on page 10-80
"PrinterPortPath" on page 10-83
"ProcessExcelUrlHyperlinks" on page 10-84
"ProcessInDesignBookMarks" on page 10-85
"ProcessInDesignHyperlinks" on page 10-86
"ProcessInDesignInternalPDFThumbnail" on page 10-87
"ProcessInDesignLinkUpdate" on page 10-88
"ProcessInDesignPDFCompatibility50" on page 10-89
"ProcessPowerpointHyperlinks" on page 10-90
"ProcessVisioInternalHyperlinks" on page 10-91
"ProcessVisioUrlHyperlinks" on page 10-92
"ProcessWordBookMarks" on page 10-93
"ProcessWordComments" on page 10-94
"ProcessWordEmailLinks" on page 10-95
"ProcessWordFootEndNote" on page 10-96
"ProcessWordHeadings" on page 10-97
"ProcessWordHyperlinks" on page 10-98
"ProcessWordToc" on page 10-99
"ProcessWordUrlHyperlinks" on page 10-100
"pucDefaultTime" on page 10-101
"pucEnableOnSearch" on page 10-102
"pucHideTimeInDateFields" on page 10-103
"pucUseMonthLocale" on page 10-104
"pucUseWeekdayLocale" on page 10-105
"pucXOffset" on page 10-106
"pucYOffset" on page 10-107
"pucZIndex" on page 10-108
"RecordsManagementDenyAuthorFreePassOnRMSecurity" on page 10-110
"RecordsManagementNumberOverwriteOnDelete" on page 10-111
"RmaAddDocWhereClauseForScreening" on page 10-112
"RMAHideExternalFieldsFromCheckInUpdate" on page 10-113
"RMAHideExternalFieldsFromSearchInfo" on page 10-114
"RmaNotifyDispReviewerAndCatAuthor" on page 10-115
"RmaNotifyReviewerAndAlternateReviewer" on page 10-116
"SecondaryConversionStepName" on page 10-118
"ShowHyperlinkBox" on page 10-120
"VerboseMode" on page 10-125
"WeblayoutDir" on page 8-361

9.2 Conversion

This section describes the Idoc Script variables and functions that are related to various conversion products.

- "Inbound Refinery" on page 9-3
- "PDF Converter" on page 9-3
- "Dynamic Converter" on page 9-4

9.2.1 Inbound Refinery

The following Idoc Script variables are related to Inbound Refinery.

"AllowSecondaryConversion" on page 10-4
"DefaultFilterInputFormat" on page 10-27
"DefaultFilterOutputFormat" on page 10-28
"DefaultGraphicTimeout" on page 10-29
"DefaultNativeTimeout" on page 10-31
"FileEncoding" on page 8-92
"IsAutoQueue" on page 10-52
"IsContentRefineryPresent" on page 10-53
"IsFailedConversion" on page 10-55
"JvmCommandLine" on page 8-200
"MaxNumRecursiveStepDefinitions" on page 10-67
"QueueDir" on page 10-109
"PostConversionTimeOut" on page 10-75
"PreconversionPath" on page 10-79
"PreconversionWaitTime" on page 10-80
"SecondaryConversionStepName" on page 10-118
"VaultDir" on page 8-358
"VerboseMode" on page 10-125
"WeblayoutDir" on page 8-361

9.2.2 PDF Converter

This section lists the Idoc Script variables and functions that are related to PDF Converter.

- "General Configuration Variables" on page 9-3
- "Microsoft Office Configuration Variables" on page 9-3
- "InDesign Configuration Variables" on page 9-4
- "Native Applications Configuration Variables" on page 9-4

9.2.2.1 General Configuration Variables

"DefaultPostscriptTimeout" on page 10-32
"DistillerPath" on page 10-36
"NumAdditionalRenditions" on page 10-70
"PostprocessPDFPath" on page 10-76
"PostscriptPrinterName" on page 10-77
"PrinterPortPath" on page 10-83
"ShowHyperlinkBox" on page 10-120

9.2.2.2 Microsoft Office Configuration Variables

"AllowSkippedHyperlinktoCauseIncomplete" on page 10-5
"PowerPointPrintOutput" on page 10-78
"ProcessExcelUrlHyperlinks" on page 10-84
"ProcessPowerpointHyperlinks" on page 10-90
"ProcessVisioInternalHyperlinks" on page 10-91

"ProcessVisioUrlHyperlinks" on page 10-92
"ProcessWordBookMarks" on page 10-93
"ProcessWordComments" on page 10-94
"ProcessWordEmailLinks" on page 10-95
"ProcessWordFootEndNote" on page 10-96
"ProcessWordHeadings" on page 10-97
"ProcessWordHyperlinks" on page 10-98
"ProcessWordToc" on page 10-99
"ProcessWordUrlHyperlinks" on page 10-100

9.2.2.3 InDesign Configuration Variables

"ProcessInDesignBookMarks" on page 10-85
"ProcessInDesignHyperlinks" on page 10-86
"ProcessInDesignInternalPDFThumbnail" on page 10-87
"ProcessInDesignLinkUpdate" on page 10-88
"ProcessInDesignPDFCompatibility50" on page 10-89

9.2.2.4 Native Applications Configuration Variables

"FrameMakerCyclesAllowed" on page 10-42
"FrameMakerTimeout" on page 10-43
"MSPubexePath" on page 10-68
"OpenOfficeListenerPort" on page 10-71
"OpenOfficeListenerURL" on page 10-72
"OpenOfficeListenerPort" on page 10-71
"PageMakerExePath" on page 10-74

9.2.3 Dynamic Converter

The following Idoc Script variables are related to Dynamic Converter.

"ConversionCacheExpirationPeriodInDays" on page 10-20
"DatedCacheIntervalDays" on page 10-21
"DCMaxFileSize" on page 10-22
"dcShowExportLink" on page 10-23
"DCTimeOut" on page 10-24
"DCViewFormat" on page 10-25
"DefaultFilterInputFormat" on page 10-27
"DefaultFilterOutputFormat" on page 10-28
"DefaultHtmlConversion" on page 10-30
"DisableForcedConversions" on page 10-33
"DisableKeepingForcedConversions" on page 10-34
"DisableOptimizedLatestReleaseQuerying" on page 10-35
"DynamicConverterHtmlFlavor" on page 10-38
"EnableOptimizedLatestReleaseQuerying" on page 10-39
"EnableRebuildForcesReconversion" on page 10-40
"ForcedConversionRules" on page 10-41
"HighlightBegin" on page 10-46
"HighlightEnd" on page 10-47
"incDynamicConversionByRule" on page 10-50
"IsDynamicConverterEnabled" on page 10-54
"rule:IsFragmentOnly" on page 10-117
"LatestReleasedTimestampTimeout" on page 10-56
"MaxConversionCacheSizeInMegs" on page 10-66
"NeverTimeoutForcedConversionsTemplateChange" on page 10-69
"TimeoutChangedTemplateForcedConversionsInMins" on page 10-121

"TimeoutMismatchedForcedConversionsInMins" on page 10-122
"TraceConversionCacheCleanup" on page 10-123
"UseCascadingStyleSheets" on page 10-124

9.3 Folders and WebDAV

The following Idoc Script variables are related to Folders and WebDAV.

"CollectionDeleteEnabled" on page 10-6
"CollectionHiddenMeta" on page 10-7
"CollectionInhibitUpdateMeta" on page 10-8
"CollectionMeta" on page 10-9
"CollectionMoveEnabled" on page 10-10
"CollectionReadOnlyMarkedFolders" on page 10-11
"CollectionReadOnlyMeta" on page 10-12
"CollectionReleasedOnly" on page 10-13
"CollectionSecurityReadOnly" on page 10-14
"CollectionTrashDeleteDate" on page 10-15
"CollectionTrashDeleteLocation" on page 10-16
"CollectionTrashDeleteOldName" on page 10-17
"CollectionTrashDeleter" on page 10-18
"CollectionWebDAVServer" on page 10-19
"InitialColID" on page 10-51

9.4 HTML Preview

The following Idoc Script variables are related to HTML Preview.

"DtmPort" on page 10-37
"PreviewOutputExtension" on page 10-81
"PreviewPath" on page 10-82

9.5 Pop-Up Calendar

The following configuration variables are used with Pop-Up Calendar.

"pucDefaultTime" on page 10-101
"pucEnableOnSearch" on page 10-102
"pucHideTimeInDateFields" on page 10-103
"pucUseMonthLocale" on page 10-104
"pucUseWeekdayLocale" on page 10-105
"pucXOffset" on page 10-106
"pucYOffset" on page 10-107
"pucZIndex" on page 10-108

9.6 Records Manager

The following Idoc Script variables are related to Records Manager.

"AllowRetentionPeriodWithoutCutoff" on page 10-3
"HideVitalReview" on page 10-45
"RecordsManagementDenyAuthorFreePassOnRMSecurity" on page 10-110
"RecordsManagementNumberOverwriteOnDelete" on page 10-111
"RmaAddDocWhereClauseForScreening" on page 10-112
"RMAHideExternalFieldsFromCheckInUpdate" on page 10-113
"RMAHideExternalFieldsFromSearchInfo" on page 10-114

"RmaNotifyDispReviewerAndCatAuthor" on page 10-115
"RmaNotifyReviewerAndAlternateReviewer" on page 10-116
"ShowContentForStorageBrowse" on page 10-119

10

Feature and Product Variables and Functions

This chapter lists Idoc Script variables and functions in alphabetical order, and provides details about each item.

AllowForceDelete

Used to delete content items that are referenced or used by other content items.

The default setting is `AllowForceDelete=FALSE`. This configures the Content Server to refuse a request to delete a content item that is referenced as a link.

If `AllowForceDelete` is set to true, a content item that is referenced as a link can be deleted. This invalidates the link.

Type and Usage

- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
AllowForceDelete=TRUE
```

See Also

- See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about Link Manager use.

AllowRetentionPeriodWithoutCutoff

Specifies retention periods for non-record content items. If set to TRUE, retention periods for triggers for non-record content items are enabled. If set to FALSE, retention periods are disabled.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

DomainHome/ucm/urm/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
AllowRetentionPeriodWithoutCutoff=false
```

AllowSecondaryConversion

Defines if the Inbound Refinery will use a backup conversion step if the native conversion step fails.

- When set to TRUE, the backup conversion step will be used upon conversion failure.
- When set to FALSE, no backup conversion step will be used upon conversion failure.
- Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

- Local Inbound Refinery Configuration screen,General tab-Allow secondary conversion step
- *IntradocDir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
AllowSecondaryConversion=false
```

See Also

- "SecondaryConversionStepName" on page 10-118

AllowSkippedHyperlinktoCauseIncomplete

Marks the conversion as incomplete when a link cannot be converted to PDF.

- When set to TRUE, the conversion is marked as incomplete when a link cannot be converted.
- When set to FALSE, hyperlinks that cannot be converted are skipped, and the conversion is marked as converted.
- Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

As a configuration entry:

```
AllowSkippedHyperlinktoCauseIncomplete=false
```

CollectionDeleteEnabled

Specifies if the Delete icon is displayed so users can delete folders and content items.

- When set to TRUE, displays the Delete icon so users can delete folders and content items from the Exploring page.
- When set to FALSE, hides the Delete icon.
- This variable is set automatically by the Folders installation program according to options specified at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

Example

Used as a configuration entry:

```
CollectionDeleteEnabled=true
```

CollectionHiddenMeta

Enables users to specify if a virtual folder or content item is hidden or visible.

- This setting adds Hide/Unhide icons to the Exploring page.
- This setting adds a **Show hidden when browsing** check box to the Folder Configuration page.
- This is set automatically by the Folders installer program according to options specified at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/data/components/folders/install.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionInhibitUpdateMeta

Defines an inhibit field for the metadata propagation fuction. This variable enables contributors to copy metadata values from a folder to its subfolders and content items.

This is set automatically by the Folders installer program according to the field name specified at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionMeta

Specifies the metadata field that is used to store the unique Collection ID for each virtual folder.

- If this setting is not specified in the configuration file, the default value is `xCollectionID`.
- This is set automatically by the Folders installer program according to the field name specified at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionMoveEnabled

Specifies if the Move icon is displayed on the Exploring pages.

- When set to TRUE, displays the Move icon so users can move folders and content items to other folders.
- When set to FALSE, hides the Move icon.
- This variable is set automatically by the Folders installation program according to options specified at install tme.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

Example

Used as a configuration entry:

```
CollectionMoveEnabled=true
```

CollectionReadOnlyMarkedFolders

Specifies if system-level folders (such as Contribution Server folders and Trash) can be modified (that is, set to Read Only status).

- When set to TRUE, prevents modification of system level folders.
- When set to FALSE, enables modifications.
- This variable is set to TRUE by default.
- If this variable is set to FALSE, you should assign an appropriate security group to the system-level folders so only authorized personnel can change them. System-level folders should be changed only if there is a particular reason to do so.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

Example

Used as a configuration entry:

```
CollectionReadOnlyMarkedFolders=true
```

CollectionReadOnlyMeta

Enables users to specify if a virtual folder and its contents are read-only.

- Read-only is subject to the rules of the operating environment where the instance is run.
- This adds Read-only/Editable icons to the Exploring pages.
- This setting applies only to the current folder level. A subfolder does not inherit the status from the parent.
- This variable is set automatically by the Folders installation program according to options specified at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/data/components/folders/install.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionReleasedOnly

Specifies if a revised document is visible to the author and users with Read permission to the content item.

- If set to TRUE and if the item has not released and the user accessing the item is not the author, no part of the content item will be visible.
- When set to FALSE, allows the author and all users with Read permission to the content item to see the latest version.
- This variable is set to FALSE by default.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

Example

Used as a configuration entry:

```
CollectionReleasedOnly=false
```

CollectionSecurityReadOnly

Specifies if users can see content items for which they have only Read permission.

- When set to TRUE, allows users to see content items for which they have only Read permissions.
- When set to FALSE, requires that users have at least Write permission to see a content item.
- This variable is set automatically by the Folders installation program according to options specified at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

Example

Used as a configuration entry:

```
CollectionSecurityReadOnly=false
```

CollectionTrashDeleteDate

Defines a date metadata field for the Trash function.

- When set, this defines a date metadata field for the Trash function, which enables the deletion date and time to be recorded as metadata for items moved to the Trash folder.
- If set, the deletion date and time are displayed in the Trash folder.
- This variable is set automatically by the Folders installation program if the Trash Bin function is enabled at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionTrashDeleteLocation

Defines an integer metadata field designating the parent folder for the Trash function.

- When set, this enables the original parent folder to be recorded as metadata for items moved to the Trash folder.
- If set, users can restore deleted items from the Trash folder.
- If set, the deletion date and time are displayed in the Trash folder.
- This variable is set automatically by the Folders installation program if the Trash Bin function is enabled at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionTrashDeleteOldName

Defines a metadata field for the Trash function to track file names of deletions.

- When set, files that are renamed due to a naming conflict in the Trash folder can be restored with their original file names.
- This defines a file name metadata field for the Trash function. This enables the original file name to be recorded as metadata for items that are moved to the Trash folder.
- This variable is set automatically by the Folders installation program if the Trash Bin function is enabled at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionTrashDeleter

Defines a metadata field for the Trash function to track user deletions.

- When set, this defines a metadata field for the Trash function which records the user's login information as metadata for items that are moved to the Trash folder.
- If set, users can choose to view only the items they have moved to the Trash folder or view all items that everyone has moved to the Trash folder.
- This variable is set automatically by the Folders installation program if the Trash Bin function is enabled at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

See Also

This feature can be enabled manually. See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details.

CollectionWebDAVServer

Enables switching views in Windows Explorer.

- If set to a URL, this enables users to switch the view from a Content Server to a WebDAV folder in Windows Explorer.
- WebDAV functionality on the Content Server must be installed to use this function.
- In Microsoft Internet Explorer 5.0 or later, this adds a web folder icon to the virtual Exploring folder (this icon is not displayed in any other browser).
- This variable is set automatically by the Folders installation program if the WebDAV function is enabled at install time.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

Example

```
CollectionWebDAVServer=http://my_computer/root/idcplg/webdav
```

ConversionCacheExpirationPeriodInDays

Defines the minimum number of days that Dynamic Converter content will remain cached.

- Content converted by Dynamic Converter will not be deleted from the cache before it is this many days old (based on last access date). Content may stay in the cache longer, depending on when the next cleanup scan is run. See DatedCacheIntervalDays.
- This setting applies only to files that are no longer in the Content Server or were not generated by a forced or up front conversion. See MaxConversionCacheSizeInMegs information on cache cleanup rules.
- Default is 7.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ConversionCacheExpirationPeriodInDays=14
```

See Also

- "DatedCacheIntervalDays" on page 10-21
- "MaxConversionCacheSizeInMegs" on page 10-66

DatedCacheIntervalDays

Defines the number of days between scans for cached Dynamic Converter content to delete.

- The Content Server scans at this interval to find and delete old cached content.
- Default is 7.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DatedCacheIntervalDays=30
```

See Also

- "ConversionCacheExpirationPeriodInDays" on page 10-20
- "MaxConversionCacheSizeInMegs" on page 10-66

DCMaxFileSize

Defines the maximum size in bytes of files that Oracle Dynamic Converter will process.

- If a source file is larger than the specified value, the conversion will not occur.
- Default is 20000000 (20 megabytes).
- Used in Dynamic Converter version 5.0 and higher.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DCMaxFileSize=10000000
```

dcShowExportLink

This function verifies if the Dynamic Converter has been configured to convert a content item. The returned value is based on the value for `dFormat` for the item.

This function is typically used on Search Result pages to conditionally display a Dynamic Converter link.

Type and Usage

- "Conditional Dynamic Variables" on page 3-1
- "Dynamic Converter" on page 4-7

Output

- Returns TRUE if the Dynamic Converter is configured to convert the content item.
- Returns FALSE if the Dynamic Converter is not configured.

Example

```
<$QueryText = "dDocTitle <substring> `test`"$>
<$executeService("GET_SEARCH_RESULTS")$>
<$loop SearchResults$>
    <$if dcShowExportLink()$>
        <a href="<$HttpCgiPath$>?IdcService=
        GET_DYNAMIC_CONVERSION&dID=<$dID$>">
        HTML Conversion of <$dDocTitle$></a>
    <$endif$>
<$endloop$>
```

DCTimeOut

Sets the timeout value in minutes for the Oracle Dynamic Converter conversion process.

- Defines the number of minutes to wait for the dynamic conversion of a document into HTML.
- Default is 3 minutes.
- Used in Dynamic Converter version 5.0 and higher.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DCTimeOut=5
```

DCViewFormat

Defines which rendition of the source file will be converted by Oracle Dynamic Converter.

- When set to *native*, the native file is converted.
- When set to *webviewable*, the web-viewable file is converted.
- When set to *alternate*, the alternate file is converted.
- Default is *native*.
- Used in Dynamic Converter version 5.0 and higher.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DCViewFormat=webviewable
```

DebugStdConversion

Configuration setting that overrides the conversion/step definitions published by the Content Server.

Type and Usage

- "Configuration Variables" on page 3-2
- "Debugging" on page 4-9

Parameters

This variable takes one argument, the path to the new definitions.

Location

IntradocDir/IdcRefinery/shared/idcrefinery.cfg

Example

```
DebugStdConversion=/temp/idcconversions
```

DefaultFilterInputFormat

Defines the default input character set for conversion filters, such as Dynamic Converter or the Text Indexer Filter component.

- Conversion products use the character encoding of the system to determine the encoding of the input to the conversion filter. This should be sufficient in most cases; however, there may be cases when you want to override this value.
- For a complete list of encoding values, see the `lcIsoEncoding` values in the *std_encoding.htm* file, which is located in the *IdcHomeDir*/resources directory.
- Default is *Cp1252*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Localization" on page 4-13
- "Inbound Refinery" on page 4-6
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry for Japanese text:

```
DefaultFilterInputFormat=shift_jis
```

Used as a configuration entry for Korean text:

```
DefaultFilterInputFormat=euc-kr
```

See Also

- "DefaultFilterOutputFormat" on page 10-28

DefaultFilterOutputFormat

Defines the default output character set for conversion filters, such as Dynamic Converter or the Text Indexer Filter component.

- Conversion products use the character encoding of the system to determine the encoding of the output to the conversion filter. This should be sufficient in most cases; however, there may be cases when you want to override this value.
- For a complete list of encoding values, see the `lcIsoEncoding` values in the *std_encoding.htm* file, which is located in the *IdcHomeDir*/resources directory.
- Default is *Cp1252*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Localization" on page 4-13
- "Inbound Refinery" on page 4-6
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry for Japanese text:

```
DefaultFilterOutputFormat=shift_jis
```

Used as a configuration entry for Korean text:

```
DefaultFilterOutputFormat=euc-kr
```

See Also

- "DefaultFilterInputFormat" on page 10-27

DefaultGraphicTimeout

Defines the default thumbnail timeout.

- Returns the thumbnail timeout values.
- This variable must be followed by one of these flags:

Flag	Description
#factor	This value is multiplied by the file size to determine the amount of time allowed for thumbnail conversion. (The resulting time is limited by the #max and #min flags.) The default value is 3. Increase this value for busy or slow systems.
#max	Specifies the maximum time in seconds that the Inbound Refinery waits for the thumbnail conversion process to complete. The default time is 60 seconds.
#min	Specifies the minimum time in seconds that the Inbound Refinery waits for the thumbnail conversion process to complete. The default time is 5 seconds.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Timeout Values tab, Default Timeout for Graphics
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
DefaultGraphicTimeout#factor=3
```

DefaultHtmlConversion

Defines the default template for Dynamic Converter conversions.

- This value must be the Content ID of a checked-in template file to specify as the default.
- There is no default value.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DefaultHtmlConversion=PLAIN
```

DefaultNativeTimeout

Defines the default native file timeout.

- Returns the native file timeout values.
- This variable must be followed by one of these flags:

Flag	Description
#factor	This value is multiplied by the file size to determine the amount of time allowed for native-to-PostScript conversion. (The resulting time is limited by the #max and #min flags.) The default value is 3. Increase this value for busy or slow systems.
#max	Specifies the maximum time in seconds that the Inbound Refinery waits for the native-to-PostScript conversion process to complete. The default time is 10 seconds.
#min	Specifies the minimum time in seconds that the Inbound Refinery waits for the native-to-PostScript conversion process to complete. The default time is 2 seconds.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Timeout Values tab, Default Native to Postscript Timeout
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
DefaultNativeTimeout#factor=3
```

DefaultPostscriptTimeout

Defines the default PostScript timeout.

- Returns the default PostScript timeout values.
- This variable must be followed by one of these flags:

Flag	Description
#factor	This value is multiplied by the file size to determine the amount of time allowed for PostScript-to-PDF conversion. (The resulting time is limited by the #max and #min flags.) The default value is 4. Increase this value for busy or slow systems.
#max	Specifies the maximum time in seconds that the Inbound Refinery waits for the PostScript-to-PDF conversion process to complete. The default time is 2 seconds.
#min	Specifies the minimum time in seconds that the Inbound Refinery waits for the PostScript-to-PDF conversion process to complete (native or PostScript). The default time is 20 seconds.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Timeout Values tab, Default Postscript to PDF Timeout
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
DefaultPostscriptTimeout#factor=4
```

DisableForcedConversions

Defines whether forced and up front conversions in Dynamic Converter are disabled.

- When set to TRUE, files will only be converted by Dynamic Converter when a user views the dynamically converted rendition. Conversions at checkin (forced and up front) will not occur.
- This setting applies to any new files checked in after the setting is set to TRUE. Any existing forced and up front conversions will still be used, unless the DisableKeepingForcedConversions setting is enabled.
- This setting might be useful in the following situations:
  - Server space for caching or processing capacity for conversion is limited.
  - Most files are rarely viewed as web pages, so a relatively small number of files must be converted dynamically.
  - The delay time for on-the-fly conversions is acceptable.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DisableForcedConversions=true
```

See Also

- "DisableKeepingForcedConversions" on page 10-34
- "ForcedConversionRules" on page 10-41

DisableKeepingForcedConversions

Defines whether Dynamic Converter treats converted files as forced conversions for updating purposes.

- When set to TRUE, files that were converted by forced or up front conversion are not considered forced conversions by Dynamic Converter. This means that even if Dynamic Converter would normally use the existing conversion rather than reconverting the file, the existing conversion will not be recognized as a forced conversion and the file will be reconverted.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DisableKeepingForcedConversions=true
```

See Also

- "DisableForcedConversions" on page 10-33

DisableOptimizedLatestReleaseQuerying

Defines whether cached timestamps or Content Server database information is used by Dynamic Converter to determine which files must be reconverted.

- When set to TRUE, Dynamic Converter queries the Content Server database rather than the cached timestamps to determine which files must be reconverted.
- Enabling this setting reverts to pre-6.2 Dynamic Converter behavior. This setting can be used if you are experiencing problems with the optimized query feature (EnableOptimizedLatestReleaseQuerying) or you need to troubleshoot a related issue.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DisableOptimizedLatestReleaseQuerying=true
```

See Also

- "EnableOptimizedLatestReleaseQuerying" on page 10-39

DistillerPath

Sets the path to the executable used by PDF Converter to convert files to PDF.

- Returns the path as a string.
- If Adobe Acrobat Distiller is not detected during installation of PDF Converter, the default is the path to the Jaws PDF Library.
- If Adobe Acrobat Distiller is detected during installation of PDF Converter, the default is the path to the Distiller executable.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Paths tab, Adobe Distiller Executable File Path
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
DistillerPath=C:/Program Files/IdcRefinery/shared/jaws_ps/ps2pdf.exe
```

DtmPort

Defines the port number on which the DtmServer (required for HTML Preview) listens to connections from Oracle Content Publisher clients.

Default is 4441.

Type and Usage

- "Configuration Variables" on page 3-2
- "HTML Preview" on page 4-12

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DtmPort=4442
```

DynamicConverterHtmlFlavor

Defines the HTML flavor used by Dynamic Converter.

- Possible settings are:
  - Internet Explorer 4.0
  - Netscape 4.0
  - HTML 3.0
  - HTML 4.0
- Default is *null*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
DynamicConverterHtmlFlavor=HTML 4.0
```

EnableOptimizedLatestReleaseQuerying

Defines whether the timestamp of a content item is updated when its metadata is updated.

- Timestamps are cached and used by Dynamic Converter to determine which files must be reconverted.
- When set to TRUE, the timestamps for all dynamically converted, web-viewable renditions are updated by one second when the metadata is updated for the source content item. This ensures that the change will be recognized by Dynamic Converter, and the file will be reconverted.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
EnableOptimizedLatestReleaseQuerying=true
```

See Also

- "DisableOptimizedLatestReleaseQuerying" on page 10-35
- "LatestReleasedTimestampTimeout" on page 10-56

EnableRebuildForcesReconversion

Defines whether up front and forced Dynamic Converter conversions are reevaluated during a rebuild of the search index.

- When set to TRUE, content items being indexed during an index rebuild will be reevaluated for conversion by Dynamic Converter based on the conversion rules for forced and up front conversions.
- This setting can be used to dynamically convert files during an index rebuild after you have created a new conversion rule.
- Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
EnableRebuildForcesReconversion=true
```

See Also

- "ForcedConversionRules" on page 10-41

ForcedConversionRules

Defines the Dynamic Converter rules that will apply forced conversions upon file checkin.

- This is a comma-delimited list.
- If a content item matches any of the specified conversion rules upon checkin, the file will be converted according to that rule. Each file can be converted into multiple renditions.
- There is no default.

> **Note:** The forced conversion process does not apply the same Dynamic Converter template twice. For example, if you set `ForcedConversionRules=RuleA,RuleB`, but RuleA and RuleB specify the same template with a different layout, the conversion according to RuleB will not occur. (Best practice in this case would be to merge the layouts and use Idoc Script to dynamically select the appropriate layout elements.)

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
ForcedConversionRules=Rule1,Rule2,Rule3
```

See Also

- "DisableForcedConversions" on page 10-33
- "incDynamicConversionByRule" on page 10-50
- "rule:IsFragmentOnly" on page 10-117

FrameMakerCyclesAllowed

The maximum number of 30-second cycles that the Inbound Refinery allows before FrameMaker is considered to be not responding.

The Inbound Refinery watches the *consfile.txt* file for FrameMaker activity.

Default is 1.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
FrameMakerCyclesAllowed=1
```

FrameMakerTimeout

Defines the FrameMaker conversion timeout.

Returns the FrameMaker timeout values.

This variable must be followed by one of these flags:

Flag	Description
#factor	This value is multiplied by the file size to determine the amount of time allowed for FrameMaker conversion. (The resulting time is limited by the #max and #min flags.) The default value is 1. Increase this value for busy or slow systems.
#max	Specifies the maximum time in seconds that the Inbound Refinery waits for the FrameMaker conversion process to complete. The default time is 10 seconds.
#min	Specifies the minimum time in seconds that the Inbound Refinery waits for the FrameMaker conversion process to complete. The default time is 1 second.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Timeout Values tab, FrameMaker to PostScript Timeout
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
FrameMakerTimeout#factor=2
```

HasSiteStudio

Used with the Link Manager functionality.

This variable is used to enable Site Studio-specific link patterns for parsing friendly URLs for the pattern engine. If set to TRUE, this variable enables the Site Studio-specific link patterns.

The default value is FALSE, disabling the Site Studio-specific link patterns.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
HasSiteStudio=TRUE
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about Link Manager use.

HideVitalReview

If set to TRUE, hides Subject to Review related fields on Universal Records Manager search and checkin pages. If set to FALSE, fields are revealed. The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

IntradocDir/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
HideVitalReview=false
```

HighlightBegin

Sets the opening HTML tag of the text highlighting style for files converted by Dynamic Converter.

Default is `<strong>`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
HighlightBegin=<b><font color=blue>
```

See Also

- "HighlightEnd" on page 10-47

HighlightEnd

Sets the closing HTML tag of the text highlighting style for files converted by Dynamic Converter.

Default is `</strong>`.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
HighlightEnd=</font></b>
```

See Also

- "HighlightBegin" on page 10-46

IdcNativeDir

Defines the path to the resource data subdirectory of the runtime media home for Content Server.

For information on local copies and load balancing see the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Enterprise Content Management Suite*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

- IdcNativeDir/resources

Example

Used as a configuration entry:

```
$IdcNativeDir=/ucm/cs/resources/
```

IdcResourcesDir

Defines the path to the resource data subdirectory of the runtime media home for Universal Content Management.

For information on local copies and load balancing see the *Oracle Fusion Middleware Enterprise Deployment Guide for Oracle Enterprise Content Management Suite*.

Type and Usage

- "Configuration Variables" on page 3-2
- "Directories and Paths" on page 4-9

Location

- IdcHomeDir/resources

Example

Used as a configuration entry:

```
$IdcHomeDir=/ucm/resources/
```

incDynamicConversionByRule

Returns the results of converting the LatestReleased revision of a document using the template and layout associated with a particular Dynamic Converter conversion rule.

Fragments created through forced conversions can be referenced directly using this function.

Type and Usage

- "Global Functions" on page 3-11
- "Dynamic Converter" on page 4-7

Parameters

Takes two parameters:

- The first parameter is the Content ID of a document.
- The second parameter is the name of a conversion rule.

Output

Retrieves the specified document converted according to the template and layout defined for the specified conversion rule.

Example

Retrieves the converted rendition of the content item with a Content ID of *PhoneList*, converted according to conversion rule *RuleA*:

```
<$incDynamicConversionByRule("PhoneList","RuleA")$>
```

See Also

- "ForcedConversionRules" on page 10-41

InitialColID

Defines the counter that initializes the first collection ID.

This defines the initial folder identifier for the server (in millions).

This variable is set automatically by the Folders installation program if the WebDAV function is enabled at install.

Default is 0.

Type and Usage

- "Configuration Variables" on page 3-2
- "Folders and WebDAV" on page 4-12

Location

IntradocDir/components/folders/folders_environment.cfg

Example

```
InitialColID=0
```

IsAutoQueue

Enables the processing of content items after conversion by the Inbound Refinery.

If set to TRUE, the Content Server will process content items after conversion by the Inbound Refinery.

If set to FALSE, the Content Server will not process content items after conversion by the Inbound Refinery.

The NoAutomation setting overrides this setting.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IsAutoQueue=false
```

See Also

- "NoAutomation" on page 6-179

IsContentRefineryPresent

Specifies if the Inbound Refinery is installed.

This value is set to FALSE during initial Content Server installation.

This value is set to TRUE during Inbound Refinery installation.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
IsContentRefineryPresent=true
```

IsDynamicConverterEnabled

Enables Dynamic Converter.

This value is set to FALSE during initial Content Server installation.

This value is set to TRUE during Dynamic Converter installation.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

- System Properties, Options tab, Enable Dynamic Converter
- Admin Server, General Configuration, Enable Dynamic Converter
- *IntradocDir*/config/config.cfg

Example

As a configuration setting, enables Dynamic Converter functionality:

```
IsDynamicConverterEnabled=true
```

As script, used to evaluate whether dynamic converter functionality is enabled:

```
<$if IsDynamicConverterEnabled and isTrue(IsDynamicConverterEnabled)$>
    <tr>
      <td align="center" width="75">
        <a href="<$HttpCgiPath$>?IdcService=GET_TEMPLATE_CONVERSIONS">
          <img src="<$HttpImagesRoot$><$admin_conversion_image$>" border="0"
        width="<$adminImageWidth$>" height="<$adminImageHeight$>"></a>
      </td>
      <td width="10">&nbsp;
      </td>
      <td width="245"><span class=largeTableEntry>
        <a href="<$HttpCgiPath$>?IdcService=GET_TEMPLATE_CONVERSIONS">
        <$lc("wwTemplateConversions")$></span></a>
      </td>
    </tr>
<$endif$>
```

IsFailedConversion

Checks if the Inbound Refinery has failed to convert the content item.

Type and Usage

- "Value Variables" on page 3-21
- "Content Items" on page 4-3
- "Inbound Refinery" on page 4-6

Output

- Returns TRUE if the conversion process failed.
- Returns FALSE if no conversion failure was detected.

Example

Displays text if the conversion process was not complete:

```
<$if IsFailedConversion$>
    <p><font face="arial" size="2">
    The Refinery was unable to complete the conversion process.</p>
<$endif$>
```

LatestReleasedTimestampTimeout

Defines the minimum interval in milliseconds at which the timestamp of a converted content item will be checked.

For a particular cached web-viewable file, Dynamic Converter will consult the timestamp again only after the time specified by this setting has passed.

If the timestamp of a cached content item has changed, Dynamic Converter reconverts the file.

Default is 1500 milliseconds (1.5 seconds).

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LatestReleasedTimestampTimeout=60000
```

See Also

- "EnableOptimizedLatestReleaseQuerying" on page 10-39

LkDisableOnRebuild

Used with the Link Manager functionality.

This variable controls link extraction during the indexing rebuild cycle.

If set to TRUE, this variable prevents Link Manager from extracting links during the index rebuild cycle. If set to FALSE, links are extracted.

The default is FALSE.

Because Link Manager does all of its work during the indexing cycle, it will increase the amount of time required to index documents and to rebuild collections.

However, the time taken may not be noticeable because most of the time is spent indexing the document into the collection. The amount of time required depends on the type and size of the documents involved. That is, if the file needs to be converted, this requires more time than text-based (HTML) files.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Indexing" on page 4-13

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkDisableOnRebuild=TRUE
```

See Also

- See the *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.

LkDisableReferenceInfo

Used with the Link Manager functionality.

This variable enables or disables the Resources section on the Content Information page. If set to TRUE, it configures the Content Server to suppress the References section. Link Manager then does not compute the link references for presentation.

If set to FALSE, Content Server displays the Show and Hide toggle switches on the Content Information page. By clicking on the toggle switch, link references are displayed or concealed.

The default is FALSE.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkDisableReferenceInfo=TRUE
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.

LkDisallowConversionFormats

Used with the Link Manager functionality.

This variable provides a list of file formats that Link Manager disregards during the link extraction process. By default, this is empty.

There are some file formats (such as Word) that must be converted by HtmlExport before links can be extracted. However, links in text-based file formats (HTML) can be extracted by Link Manager without requiring conversion by HtmlExport.

For example, you may have PHP files (or files in some other custom format) that are actually text-based. For such files, conversion by HtmlExport may not be necessary before Link Manager processes them for link extraction. Such formats can be listed in this configuration variable.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Parameters

The only parameter(s) are the formats.

Location

IntradocDir/config/config.cfg

Examples

In the following example, specifying the full MIME type format limits the variations of the listed types that are excluded. For example, listing `application/msword` will not exclude `application/vnd.msword` or other variations. In this case, you must include each specific MIME type variation in the list that you want to exclude.

```
LkDisallowConversionFormats=application/msword,audio/wav,video/avi
```

In the following example, all variations of each MIME type in the list are excluded. Using the shortened format of the MIME type is more advantageous in that it allows greater flexibility.

```
LkDisallowConversionFormats=msword,wav,avi
```

> **Important:** Link Manager does not use HtmlExport to convert files that contain any of the following strings in the file format: hcs, htm, image, text, xml, jsp, and asp. These files are handled by Link Manager without need for conversion. To check the current file formats and extension mappings of your system, use the File Formats window in the Configuration Manager

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.

LkIsSecureSearch

Used with the Link Manager functionality.

This variable is used to manage security restriction checks for non-admin users during a link search.

If set to TRUE, this configures Link Manager to perform a security check on non-admin users while implementing a managed link search. This is the default setting.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkIsSecureSearch=TRUE
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.

LkReExtractOnRebuild

Used with the Link Manager functionality.

This variable controls link extraction from previously indexed documents during a rebuild. If set to TRUE, it configures Link Manager to extract links from documents as they are indexed into the system during a rebuild. This is the default setting.

If set to FALSE, this prevents Link Manager from extracting links from documents as they are indexed into the system during a rebuild.

Because Link Manager does all of its work during the indexing cycle, it will increase the amount of time required to index documents and to rebuild collections.

However, the time taken may not be noticeable since most of the time is spent indexing the document into the collection. Although, the amount of time required does depend on the type and size of the documents involved. That is, if the file needs to be converted, this requires more time than text-based (HTML) files.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkReExtractOnRebuild=FALSE
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.

LkRefreshBatchSize

Used to control the number of links submitted to Site Studio during a refresh process. By default, the batch size value is set to 100.

Depending on the value, this configuration variable can make the refresh processes more (or less) receptive to an abort request. The calculation for `LkRefreshErrorsAllowed` is completed after, not during, each batch process. Therefore, the lower the value for `LkRefreshBatchSize`, the more likely it is that the refresh activity will promptly terminate because the number of allowed errors is more quickly exceeded.

If you are using Site Studio, LkRefreshBatchSize enables you to have better control over the refresh abort option than using the combined LkRefreshErrorPercent and LkRefreshErrorThreshold settings. For example, if you set the percent value to 5 and the threshold value to 20, you would expect the refresh activity to abort after the first error. However, Link Manager might actually process more than one error before aborting.

The reason for this is that during refresh activities, all the links recognized as Site Studio links (or those requiring Site Studio processing) are grouped and sent to Site Studio as a batch. Consequently, refreshes are more efficient, but abort requests are unresponsive during this time because Site Studio is unaware of aborts and total error counts.

However, Site Studio is aware of the number of errors it has encountered in the current batch. For this reason, the Link Manager's abort calculation can not take place in all situations and the error configuration values (percent and threshold) are simply suggestions to Link Manager for when an abort should occur. Using `LkRefreshBatchSize`, however, enables you to more accurately control the abort receptivity during refresh activities that involve Site Studio link batches.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshBatchSize=200
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.

LkRefreshErrorsPercent

Used with Link Manager functionality.

This variable is used with `LkRefreshErrorThreshold` to compute if the refresh activity should be aborted. When the refresh activity processes more links than the threshold value, Link Manager calculates the proportion of errors. If the number of errors exceeds the percent value, the refresh activity is aborted.

The percent value is set to 10 by default.

> **Note:** If you choose to set a value for `LkRefreshErrorsAllowed`, it could override the combination of threshold and percent values. For example, if the value is less than the calculated threshold/percent value for links/errors, then the refresh activity could abort even if the number of links processed does not exceed the threshold limit. Therefore, it is recommended that you use either the `LkRefreshErrorsAllowed` configuration setting or the combination of the `LkRefreshErrorThreshold` and `LkRefreshErrorsPercent` configuration settings.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshErrorsPercent=20
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.
- "LkRefreshErrorsAllowed" on page 10-64
- "LkRefreshErrorThreshold" on page 10-65

LkRefreshErrorsAllowed

Used with Link Manager functionality.

This variable sets an absolute error count for refresh processes. When the set number of errors is encountered, the refresh activity aborts. This configuration setting is not used by default.

> **Note:** If you choose to set a value for `LkRefreshErrorsAllowed`, it could override the combination of threshold and percent values. For example, if the value is less than the calculated threshold/percent value for links/errors, then the refresh activity could abort even if the number of links processed does not exceed the threshold limit. Therefore, it is recommended that you use either the `LkRefreshErrorsAllowed` configuration setting or the combination of the `LkRefreshErrorThreshold` and `LkRefreshErrorPercent` configuration settings.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshErrorsAllowed=10
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.
- "LkRefreshErrorsPercent" on page 10-63
- "LkRefreshErrorThreshold" on page 10-65

LkRefreshErrorThreshold

Used with Link Manager functionality.

This variable is used with `LkRefreshErrorPercent` to compute if the refresh activity should be aborted. When the refresh activity processes more links than the threshold value, Link Manager calculates the proportion of errors. If the number of errors exceeds the percent value, the refresh activity is aborted.

The threshold value is set to 100 by default.

> **Note:** If you choose to set a value for `LkRefreshErrorsAllowed`, it could override the combination of threshold and percent values. For example, if the value is less than the calculated threshold/percent value for links/errors, then the refresh activity could abort even if the number of links processed does not exceed the threshold limit. Therefore, it is recommended that you use either the `LkRefreshErrorsAllowed` configuration setting or the combination of the `LkRefreshErrorThreshold` and `LkRefreshErrorPercent` configuration settings.

Type and Usage

- "Link Manager Configuration Variables" on page 3-10
- "Content Items" on page 4-3

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
LkRefreshErrorThreshold=200
```

See Also

- *Oracle Fusion Middleware Application Administrator's Guide for Content Server* for details about using Link Manager.
- "LkRefreshErrorsAllowed" on page 10-64
- "LkRefreshErrorsPercent" on page 10-63

MaxConversionCacheSizeInMegs

Defines the maximum size of the Dynamic Converter cache, in megabytes.

When a cleanup scan is run (see DatedCacheIntervalDays), the cache is cleaned up according to this setting as follows:

- If the Dynamic Converter cache is larger than this size, the oldest cached files (determined by last access date) are deleted until the cache is smaller than this size.
- If the Dynamic Converter cache is smaller than this size, only files that are older than the ConversionCacheExpirationPeriodInDays and were not converted using conversion rules (forced or up front conversion) are deleted.

Default is 10,000 (10 gigabytes).

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
MaxConversionCacheSizeInMegs=5,000
```

See Also

- "ConversionCacheExpirationPeriodInDays" on page 10-20
- "DatedCacheIntervalDays" on page 10-21

MaxNumRecursiveStepDefinitions

Specifies the maximum number of levels that recursion can be used when defining a custom conversion.

A custom conversion engine can call another conversion, but to prevent loops, this setting limits the number of levels of such calls.

Default is 6.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

IntradocDir/ucm/ibr/bin/config.cfg

Example

Used as a configuration entry:

```
MaxNumRecursiveStepDefinitions=3
```

MSPubexePath

Defines the path for the MS Publisher executable file.

Returns the path as a string.

Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Paths tab, MS Publisher Executable Path
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
MSPubexePath=c:/Program Files/Microsoft Publisher/mspub.exe
```

NeverTimeoutForcedConversionsTemplateChange

Defines whether Dynamic Converter will reconvert a file when the template or layout has changed.

When set to TRUE, files that were converted by forced or up front conversion will not be reconverted when the template or layout is changed.

If a conversion rule is changed to use a different template or layout, the normal evaluation logic is used.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
NeverTimeoutForcedConversionsTemplateChange=true
```

See Also

- "TimeoutChangedTemplateForcedConversionsInMins" on page 10-121
- "TimeoutMismatchedForcedConversionsInMins" on page 10-122

NumAdditionalRenditions

Used by the Thumbnails program to set the number of renditions of a content item.

When set to 0, thumbnails are not created on search results pages.

When set to 1, thumbnails are created on search results pages.

Default is 0.

Type and Usage

- "Configuration Variables" on page 3-2
- "Content Items" on page 4-3
- "Inbound Refinery" on page 4-6

Location

IntradocDir/config/config.cfg

Example

Used to determine if useThumbnails should be set:

```
<@dynamichtml searchapi_result_definitions@>
    <$if NumAdditionalRenditions and NumAdditionalRenditions > 0$>
        <$useThumbnails=1$>
    <$endif$>
<@end@>
```

OpenOfficeListenerPort

Defines the socket port for the OpenOffice listener.

This setting is used by PDF Converter's OpenOffice connection string:

```
uno:socket,host=<OpenOfficeListenerURL>,
port=<OpenOfficeListenerPort>;urp;StarOffice.ServiceManager
```

Default is 8100.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
OpenOfficeListenerPort=8110
```

See Also

- "OpenOfficeListenerURL" on page 10-72

OpenOfficeListenerURL

Defines the URL (socket hostname) for the OpenOffice listener.

This setting is used by PDF Converter's OpenOffice connection string:

```
uno:socket,host=<OpenOfficeListenerURL>,port=<OpenOfficeListenerPort>;urp;
StarOffice.ServiceManager
```

Default is *localhost*.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
OpenOfficeListenerURL=/mycomputer/localhost
```

See Also

- "OpenOfficeListenerPort" on page 10-71

OpenOfficePath

Defines the path for the OpenOffice listener.

If PDF Converter cannot connect to the OpenOffice connection, it attempts to open the listener defined by this path.

Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
OpenOfficePath=C:\Program Files\Open Office.org.1.0.1\program\quickstart.exe
```

PageMakerExePath

Defines the path for the Adobe PageMaker executable file.

Returns the path as a string.

Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Paths tab, PageMaker Executable File Path
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
PageMakerExePath=c:/Program Files/Adobe/PageMaker 6.5/pagemaker.exe
```

PostConversionTimeOut

Defines the default post-conversion timeout.

Returns the post-conversion timeout values.

This variable must be followed by one of these flags:

Flag	Description
#factor	This value is multiplied by the file size to determine the amount of time allowed for post-conversion processing. (The resulting time is limited by the #max and #min flags.) The default value is 1. Increase this value for busy or slow systems.
#max	Specifies the maximum time in seconds that the Inbound Refinery waits for any post-conversion processes to complete. The default time is 10 seconds.
#min	Specifies the minimum time in seconds that the Inbound Refinery waits for any post-conversion processes to complete. The default time is 1 second.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

- Shared Inbound Refinery Configuration screen, Timeout Values tab, Post PDF Production Timeout
- *IntradocDir*/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
PostConversionTimeOut#factor=1
```

PostprocessPDFPath

Defines the path to an executable that is used to process PDF files after conversion.

The PDF file path is the only command line parameter.

The specified file must be a Win32 executable.

This process is invoked after the conversion to PDF and before the optimization of the PDF. Since this is invoked once for each file that is processed by the Inbound Refinery, it could also be used for other events that might need to be synchronized to the conversion of a file. If the value is not empty, the Inbound Refinery will attempt to run the process.

Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, General tab, Post PDF Production Path
- *IntradocDir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
PostprocessPDFPath=c:/domain/IdcRefinery/ConversionProcess/IDCWaterPost.exe
```

PostscriptPrinterName

Specifies the name of the printer that is used to convert files to PostScript.

This must be the name of a valid PostScript printer.

Default is *IDC PDF Converter*.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
PostscriptPrinterName=Acrobat Distiller
```

PowerPointPrintOutput

Defines how Microsoft PowerPoint content items are converted.

When set to *slide*, each page of the PDF file contains one slide.

When set to *notes*, each page of the PDF file contains a slide with its notes.

When set to *handout*, each page of the PDF file contains six slides.

Default is *slide*.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Convert in PowerPoint format options
- *IntradocDir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
PowerPointPrintOutput=notes
```

See Also

- "ProcessPowerpointHyperlinks" on page 10-90

PreconversionPath

Defines the path to an executable that is used to process files before conversion.

The specified file must be a Win32 executable.

Default is an empty string.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Legacy Conversions tab, Preconversion Path
- *DomainHome*/urm/ibr/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
PreconversionPath=c:/out/preconversion.bat
```

PreconversionWaitTime

Defines the preconversion timeout.

Returns the preconversion timeout value.

This variable must be followed by one of these flags:

Flag	Description
#factor	This value is multiplied by the file size to determine the amount of time allowed for preconversion processes. (The resulting time is limited by the #max and #min flags.) The default value is 1. Increase this value for busy or slow systems.
#max	Specifies the maximum time in seconds that the Inbound Refinery waits for the preconversion process to complete. The default time is 10 seconds.
#min	Specifies the minimum time in seconds that the Inbound Refinery waits for the preconversion process to complete. The default time is 1 second.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Legacy Conversions tab, Pre-Converter Timeout
- *IntradocDir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
PreconversionWaitTime#max=60
```

PreviewOutputExtension

Defines the file extension output from Content Publisher that is to be used in HTML Preview.

Default is *HCSP*.

Type and Usage

- "Configuration Variables" on page 3-2
- "HTML Preview" on page 4-12

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
PreviewOutputExtension=jsp
```

PreviewPath

Defines the path for HTML Preview executable.

The *tcpreview.exe* file resides in the Content Publisher installation directory.

Type and Usage

- "Configuration Variables" on page 3-2
- "HTML Preview" on page 4-12

Location

DomainHome/ucm/cs/bin/intradoc.cfg

Example

Used as a configuration entry:

```
PreviewPath=C:/Program Files/Oracle Content Publisher/tcpreview.exe
```

PrinterPortPath

Defines the location of the PostScript file created by the Inbound Refinery.

This setting must match the port setting of the printer used by the Inbound Refinery.

Default is *c:/temp/idcoutput.ps*.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
PrinterPortPath=c:/temp/idcrefinery/idcoutput.ps
```

See Also

- "PostscriptPrinterName" on page 10-77

ProcessExcelUrlHyperlinks

Enables the processing of URL hyperlinks in Microsoft Excel content items.

When set to TRUE, URL hyperlinks in Excel files are processed.

When set to FALSE, URL hyperlinks in Excel files are not processed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process Excel URL links
- *IntradocDir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessExcelUrlHyperlinks=true
```

ProcessInDesignBookMarks

Enables the conversion of internal document links in Adobe InDesign content items.

- When set to TRUE, internal document links in Adobe InDesign files are processed.
- When set to FALSE, internal document links in Adobe InDesign files are not processed.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessInDesignBookmarks=false
```

See Also

- "ProcessInDesignHyperlinks" on page 10-86
- "ProcessInDesignInternalPDFThumbnail" on page 10-87
- "ProcessInDesignLinkUpdate" on page 10-88
- "ProcessInDesignPDFCompatibility50" on page 10-89

ProcessInDesignHyperlinks

Enables the conversion of URL links in Adobe InDesign content items.

- When set to TRUE, URL links in Adobe InDesign files are processed.
- When set to FALSE, URL links in Adobe InDesign files are not processed.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessInDesignHyperlinks=false
```

See Also

- "ProcessInDesignBookMarks" on page 10-85
- "ProcessInDesignInternalPDFThumbnail" on page 10-87
- "ProcessInDesignLinkUpdate" on page 10-88
- "ProcessInDesignPDFCompatibility50" on page 10-89

ProcessInDesignInternalPDFThumbnail

Enables the creation of PDF page thumbnails when converting Adobe InDesign content items.

- When set to TRUE, the PDF file will include page thumbnails.
- When set to FALSE, the PDF file will not include page thumbnails.

This setting does not affect the creation of thumbnails on search results pages.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessInDesignInternalPDFThumbnail=false
```

See Also

- "ProcessInDesignBookMarks" on page 10-85
- "ProcessInDesignHyperlinks" on page 10-86
- "ProcessInDesignLinkUpdate" on page 10-88
- "ProcessInDesignPDFCompatibility50" on page 10-89

ProcessInDesignLinkUpdate

Enables an update of all links in Adobe InDesign content items before conversion.

- When set to TRUE, links in Adobe InDesign files are updated before conversion. If linked objects are not available, the conversion will fail.
- When set to FALSE, links in Adobe InDesign files are not updated before conversion.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessInDesignLinkUpdate=true
```

See Also

- "ProcessInDesignBookMarks" on page 10-85
- "ProcessInDesignHyperlinks" on page 10-86
- "ProcessInDesignInternalPDFThumbnail" on page 10-87
- "ProcessInDesignPDFCompatibility50" on page 10-89

ProcessInDesignPDFCompatibility50

Sets the version of the PDF files that PDF Converter produces from Adobe InDesign content items.

- When set to TRUE, PDF Converter produces PDF version 1.4 (Acrobat 5.0 compatible) files from Adobe InDesign files.
- When set to FALSE, PDF Converter produces PDF version 1.3 (Acrobat 4.0 compatible) files from Adobe InDesign files.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

IntradocDir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessInDesignPDFCompatibility50=true
```

ProcessPowerpointHyperlinks

Enables the processing of hyperlinks in Microsoft PowerPoint content items.

- When set to TRUE *and* the PowerPointPrintOutput setting is set to SLIDE, the following hyperlinks in PowerPoint files are processed:
  - URL links
  - Next slide
  - Previous slide
- When set to FALSE, hyperlinks in PowerPoint files are not processed.

If the *PowerPointPrintOutput* setting is set to NOTES or HANDOUT, hyperlinks in PowerPoint files are not processed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process PowerPoint Hyperlinks
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessPowerpointHyperlinks=true
```

See Also

- "PowerPointPrintOutput" on page 10-78

ProcessVisioInternalHyperlinks

Enables the conversion of internal hyperlinks in Microsoft Visio content items.

- When set to TRUE, internal hyperlinks in Visio files are processed.
- When set to FALSE, internal hyperlinks in Visio files are not processed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process internal Visio links
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessVisioInternalHyperlinks=true
```

See Also

- "ProcessVisioUrlHyperlinks" on page 10-92

ProcessVisioUrlHyperlinks

Enables the conversion of URL links in Microsoft Visio content items.

- When set to TRUE, URL links in Visio files are processed.
- When set to FALSE, URL links in Visio files are not processed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process Visio URL links
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessVisioUrlHyperlinks=true
```

See Also

- "ProcessVisioInternalHyperlinks" on page 10-91

ProcessWordBookMarks

Enables the conversion of page reference bookmarks in Microsoft Word content items.

- When set to TRUE, page reference bookmarks in Microsoft Word files are converted to PDF bookmarks. (All other types of bookmarks are ignored.)
- When set to FALSE, bookmarks in Microsoft Word files are not processed.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process document bookmarks
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordBookMarks=false
```

See Also

- "ProcessWordHyperlinks" on page 10-98

ProcessWordComments

Enables the conversion of comments in Microsoft Word content items to PDF notes.

- When set to TRUE, comments in Microsoft Word files are converted to PDF notes.
- When set to FALSE, comments in Microsoft Word files are not processed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process document comments to PDF notes
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordComments=true
```

See Also

- "ProcessWordHyperlinks" on page 10-98

ProcessWordEmailLinks

Enables the conversion of e-mail hyperlinks in Microsoft Word content items.

- The ProcessWordUrlHyperlinks setting must be TRUE for this setting to be valid.
- When set to TRUE, e-mail hyperlinks in Microsoft Word files are processed.
- When set to FALSE, e-mail hyperlinks in Microsoft Word files are not processed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process mailto links
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordEmailLinks=true
```

See Also

- "ProcessWordHyperlinks" on page 10-98
- "ProcessWordUrlHyperlinks" on page 10-100

ProcessWordFootEndNote

Enables the conversion of footnotes and endnotes in Microsoft Word content items to hyperlinks.

- When set to TRUE, footnotes and endnotes in Microsoft Word files are converted to hyperlinks.
- When set to FALSE, footnotes and endnotes in Microsoft Word files are not processed.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process document links to footnotes and endnotes
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordFootEndNote=true
```

See Also

- "ProcessWordHyperlinks" on page 10-98

ProcessWordHeadings

Enables the conversion of standard headings in Microsoft Word content items to PDF bookmarks.

- When set to TRUE, standard heading paragraph styles (Heading 1 through Heading 9 and any custom styles based on these standard styles) in Microsoft Word files are converted to PDF bookmarks.
- If a table of contents has already been processed using the ProcessWordToc setting, then this setting is ignored and headings will not be converted to bookmarks.
- When set to FALSE, heading styles in Microsoft Word files are not processed.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process heading styles to PDF bookmarks
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordHeadings=false
```

See Also

- "ProcessWordHyperlinks" on page 10-98
- "ProcessWordToc" on page 10-99

ProcessWordHyperlinks

Enables the conversion of hyperlinks in Microsoft Word content items.

- When set to TRUE, all types of hyperlinks in Microsoft Word files are processed.
- When set to FALSE, hyperlinks in Microsoft Word files are not processed.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

instance_dir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordHyperlinks=true
```

See Also

- "ProcessWordBookMarks" on page 10-93
- "ProcessWordComments" on page 10-94
- "ProcessWordEmailLinks" on page 10-95
- "ProcessWordFootEndNote" on page 10-96
- "ProcessWordHeadings" on page 10-97
- "ProcessWordToc" on page 10-99
- "ProcessWordUrlHyperlinks" on page 10-100

ProcessWordToc

Enables the conversion of tables of contents in Microsoft Word content items to PDF bookmarks.

- When set to TRUE, tables of contents entries that are automatically generated in Microsoft Word files are converted to PDF bookmarks.
- When set to FALSE, tables of contents in Microsoft Word files are not processed. (However, if the table of contents entries have hyperlinks, they will be processed according to the ProcessWordUrlHyperlinks setting.)

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process Table of Contents to PDF bookmarks
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordToc=true
```

See Also

- "ProcessWordHyperlinks" on page 10-98
- "ProcessWordUrlHyperlinks" on page 10-100
- "ProcessWordHeadings" on page 10-97

ProcessWordUrlHyperlinks

Enables the conversion of URL hyperlinks in Microsoft Word content items.

- When set to TRUE, URL hyperlinks in Microsoft Word files are processed.
- When set to FALSE, URL hyperlinks in Microsoft Word files are not processed.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

- Local Inbound Refinery Configuration screen, Native Options tab, Process URL links
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
ProcessWordUrlHyperlinks=false
```

See Also

- "ProcessWordHyperlinks" on page 10-98

pucDefaultTime

Specifies how the time is entered into the date/time fields when using the pop-up calendar. If this option is not set, the time defaults to midnight of the selected date.

This is a string value which can be set in the following ways:

- current: sets the time equivalent to the client computer's clock.
- any time, expressed in the system locale format. Missing values are interpreted as a zero. Missing AM/PM designations are interpreted as using 24-hour format. For example, 1:15 PM, 01:15 PM, and 13:15 are interpreted the same within a system locale that designates AM and PM.

Type and Usage

- "Pop-Up Calendar Configuration Variables" on page 3-10
- "Pop-Up Calendar" on page 4-15

Location

IntradocDir/config/config.cfg

Example

```
pucDefaultTime=current
```

pucEnableOnSearch

If set to TRUE, specifies if the pop-up calendar is available on a search page. If set to FALSE, disables the pop-up calendar on the search page.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Pop-Up Calendar" on page 4-15

Location

IntradocDir/config/config.cfg

Example

```
pucEnableOnSearch=false
```

pucHideTimeInDateFields

Specifies if date fields display only the date or the date and time.

- If set to TRUE, disables the display of the time in a date field and forces the Content Server to parse the check-in time as midnight of the selected date, regardless of how pucDefaultTime is set.
- If set to false, displays the time in a date field.

The default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Pop-Up Calendar" on page 4-15

Location

IntradocDir/config/config.cfg

Example

```
pucHideTimeInDateFields=true
```

See Also

- "pucDefaultTime" on page 10-101

pucUseMonthLocale

Determines how the month names are displayed in the pop-up calendar.

This is a string value which can be set in the following ways:

- short: truncates the month name based on user locale. For example, January is truncated to Jan, February to Feb, and so on.
- long: displays the full month name based on user locale.

Default is long.

Type and Usage

- "Pop-Up Calendar Configuration Variables" on page 3-10
- "Pop-Up Calendar" on page 4-15

Example

```
pucUseMonthLocale=short
```

See Also

- "pucUseWeekdayLocale" on page 10-105

pucUseWeekdayLocale

Determines how the day names are displayed in the pop-up calendar.

This is a string value which can be set in the following ways:

- 1char: uses the first letter of the day. Note that this starts with the numeral '1', not a lower-case 'L'.
- short: uses two characters to represent the day name.
- medium: uses the standard three-character day names
- long: displays the full name based on user locale.

Default is long.

Type and Usage

- "Pop-Up Calendar Configuration Variables" on page 3-10
- "Pop-Up Calendar" on page 4-15

Example

```
pucUseWeekdayLocale=short
```

See Also

- "pucUseMonthLocale" on page 10-104

pucXOffset

Specifies the horizontal location of the pop-up calendar on the page in pixels, relative to the default position. This is a value set as an integer. The default is 0.

Type and Usage

- "Pop-Up Calendar Configuration Variables" on page 3-10
- "Pop-Up Calendar" on page 4-15

Example

```
pucXOffset=10
```

See Also

- "pucYOffset" on page 10-107

pucYOffset

Specifies the vertical location of the pop-up calendar on the page in pixels, relative to the default position. This is a value set as an integer. The default is 0.

Type and Usage

- "Pop-Up Calendar Configuration Variables" on page 3-10
- "Pop-Up Calendar" on page 4-15

Example

```
pucYOffset=10
```

See Also

- "pucXOffset" on page 10-106

pucZIndex

Specifies the z-ordering object position of the pop-up calendar layer. The larger the number, the higher the placement within the page layer positions. Use a positive number greater than any other z-index style attribute specified on the page to ensure the calendar is displayed over other elements.

This is specified as an integer value. Default is 0.

Type and Usage

- "Pop-Up Calendar Configuration Variables" on page 3-10
- "Pop-Up Calendar" on page 4-15

Example

```
pucZIndex=10
```

QueueDir

Defines the path to the working directory for the Inbound Refinery queue.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

IntradocDir/config/config.cfg

Example

As a configuration setting:

```
QueueDir=x:/domain_refinery/
```

RecordsManagementDenyAuthorFreePassOnRMSecurity

When a content item is placed in an acl-restricted folder (RW recordsmanager alias), users have only RW accesss, except for the author who can delete the item. Setting this configuration variable to true dissallows deletion of the item by the author, restricting the author to only RW access.

The default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
RecordsManagementDenyAuthorFreePassOnRMSecurity=true
```

RecordsManagementNumberOverwriteOnDelete

Universal Records Manager only: Sets the number of disk scrubbing passes used for a destroy action.

The default is 2.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
RecordsManagementNumberOverwriteOnDelete=4
```

RmaAddDocWhereClauseForScreening

Retention Manager only.

Allows users with the retadmin role to screen for frozen items to which they do not have access (through ACLs) on the screening page or the Freeze Information Page.

The default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/components/RetentionManager/retentionmanager_environments.cfg

Example

Used as a configuration entry:

```
RmaAddDocWhereClauseForScreening=true
```

RMAHideExternalFieldsFromCheckInUpdate

If set to TRUE, hides external fields on the Checkin and Update pages.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
RmaHideExternalFieldsFromCheckInUpdate=false
```

RMAHideExternalFieldsFromSearchInfo

If set to TRUE, hides external fields on the Search and Content Information pages. The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
RmaHideExternalFieldsFromSearchInfo=false
```

RmaNotifyDispReviewerAndCatAuthor

Controls who is notified when events are triggered by a disposition rule. If set to TRUE, both the specified notification reviewer and the original category author receive e-mail notifications of the event. If set to FALSE, only the category author receives e-mail.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
RmaNotifyDispReviewerAndCatAuthor=false
```

RmaNotifyReviewerAndAlternateReviewer

Controls if an alternate user can perform review actions and process assigned disposition events. If set to TRUE, both the original user and the alternative reviewer receive e-mail notifications about the action. If set to FALSE, only the alternative reviewer receives e-mail.

The default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Record Manager and Retention Manager" on page 4-15

Location

instance_dir/components/RecordsManagement/records_management_environments.cfg

Example

Used as a configuration entry:

```
RmaNotifyReviewerAndAlternateReviewer=false
```

rule:IsFragmentOnly

Specifies that a particular forced conversion rule creates a fragment in Dynamic Converter.

Fragments are pieces of content that will be included in another content item. Fragments generally do not include <HTML> or <BODY> tags, and are not intended to be viewed by themselves.

When this variable is set to 1, the specified forced conversion rule converts files to fragments.

The specified forced conversion rule must be included in the ForcedConversionRules setting.

There is no default.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Specifies that the *RuleA* forced conversion rule converts files to fragments:

```
RuleA:IsFragmentOnly=1
ForcedConversionRules=RuleA
```

See Also

- "ForcedConversionRules" on page 10-41

SecondaryConversionStepName

Defines the name of the backup conversion step that the Inbound Refinery will use if the native step conversion step fails.

Default is *OIXConverter* (Outside In).

AllowSecondaryConversion must be set to TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6

Location

instance_dir/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
SecondaryConversionStepName=OIXConverter
```

See Also

- "AllowSecondaryConversion" on page 10-4

ShowContentForStorageBrowse

Defines whether content is displayed when browing storage locations for Physical Content Manager (PCM).

- If set to TRUE, permits content to be displayed. If storage locations contain large numbers of items (that is, more than 100 per location), display time can be slowed.
- If set to FALSE, content is hidden in storage locations.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2

Location

- instance_dir/components/PhysicalContentManager/physicalcontentmanager_environment.cfg

Example

Used as a configuration entry:

```
ShowContentForStorageBrowse=TRUE
```

ShowHyperlinkBox

Defines whether a box is placed around hyperlinks in PDF files.

- When set to TRUE, a box is placed around hyperlinks in PDF files.
- When set to FALSE, a box is not placed around hyperlinks in PDF files.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "PDF Converter" on page 4-6

Location

instance_dir/IdcRefinery/shared/idcrefinery.cfg

Example

Used as a configuration entry:

```
ShowHyperlinkBox=true
```

TimeoutChangedTemplateForcedConversionsInMins

Defines the minimum interval in minutes at which Dynamic Converter will reconvert a file when the template or layout has changed.

- When set to a positive value, forced or up front conversions will be reconverted because of a change to the template or layout only after the time specified by this setting has passed.
- If a conversion rule is changed to use a different template or layout, the normal evaluation logic is used.
- This setting is ignored if the NeverTimeoutForcedConversionsTemplateChange setting is TRUE.
- When set to 0, the TimeoutMismatchedForcedConversionsInMins setting is used.

Default is 0.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
TimeoutChangedTemplateForcedConversionsInMins=1
```

See Also

- "NeverTimeoutForcedConversionsTemplateChange" on page 10-69
- "TimeoutMismatchedForcedConversionsInMins" on page 10-122

TimeoutMismatchedForcedConversionsInMins

Defines the minimum interval in minutes at which Dynamic Converter will reconvert a file when the source document, template, or layout has changed.

- When set to a positive value, forced or up front conversions will be reconverted because of a change to the source document, template, or layout only after the time specified by this setting has passed.
- If a conversion rule is changed to use a different template or layout, the normal evaluation logic is used.
- If the TimeoutChangedTemplateForcedConversionsInMins is set to a positive value, this setting applies only to changes in the source document.
- When set to 0, forced or up front conversions will not be reconverted because of a change to the source document.

Default is 15 minutes.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
TimeoutMismatchedForcedConversionsInMins=10
```

See Also

- "NeverTimeoutForcedConversionsTemplateChange" on page 10-69
- "TimeoutChangedTemplateForcedConversionsInMins" on page 10-121

TraceConversionCacheCleanup

Outputs a report to the debug trace during the Dynamic Converter cache cleanup cycle.

- When set to TRUE, the debug trace includes a report of which documents were deleted, the conversion type of deleted documents (forced or not), and whether documents were deleted because of cache size constraints.
- See ConversionCacheExpirationPeriodInDays, DatedCacheIntervalDays, and MaxConversionCacheSizeInMegs for more information on the Dynamic Converter cache cleanup cycle.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
TraceConversionCacheCleanup=true
```

See Also

- "ConversionCacheExpirationPeriodInDays" on page 10-20
- "DatedCacheIntervalDays" on page 10-21
- "MaxConversionCacheSizeInMegs" on page 10-66

UseCascadingStyleSheets

Enables the use of cascading style sheets (CSS) for Dynamic Converter.

When set to TRUE, cascading style sheets are enabled.

When set to FALSE, cascading style sheets are disabled. This option is useful only for older browsers that do not support CSS.

Default is TRUE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Dynamic Converter" on page 4-7

Location

IntradocDir/config/config.cfg

Example

Used as a configuration entry:

```
UseCascadingStyleSheets=false
```

VerboseMode

Specifies the level of logging in the Inbound Refinery log.

- When set to TRUE, the Inbound Refinery log records all messages and configuration information (such as whether the PDF is optimized, where the custom converter path is located, and how much time is allowed for custom conversions).
- When set to FALSE, the Inbound Refinery log records only error messages.

Default is FALSE.

Type and Usage

- "Configuration Variables" on page 3-2
- "Inbound Refinery" on page 4-6
- "Debugging" on page 4-9

Location

- Local Inbound Refinery Configuration screen, General tab, Verbose logging
- *instance_dir*/IdcRefinery/connections/*main*/intradoc.cfg

Example

Used as a configuration entry:

```
VerboseMode=true
```

Index