STELLENT™

Idoc Script Reference Guide

SDK-EN3-600

Table of Contents

CHAPTER 5: IDOC SCRIPT CONFIGURATION VARIABLES

CHAPTER 6: WEB SERVER VARIABLES

Chapter

1

INTRODUCTION

ABOUT THIS GUIDE

Idoc Script is Stellent's 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.

Note: This reference guide is part of the Stellent Software Developer's Kit (SDK). See the *Customizing Content Server* guide for more information on the SDK.

AUDIENCE

This guide is intended for developers who need to use Idoc Script to customize Stellent software. System administrators might also use Idoc Script to create workflows, define Library pages, and batch load files.

ORGANIZATION

This guide is divided into the following chapters:

- Chapter 2, *Idoc Script Application*, describes how Idoc Script is used in Stellent Content Server.
- Chapter 3, *Idoc Script Global Functions*, describes the global predefined Idoc Script functions.
- Chapter 4, *Idoc Script Predefined Variables*, describes the predefined Idoc Script variables.
- Chapter 5, *Idoc Script Configuration Variables*, describes the predefined configuration variables that are typically set in configuration files.
- Chapter 6, *Web Server Variables*, describes the CGI environment variables that are set when the server executes the gateway program.

An index is provided at the end of this guide.

CONVENTIONS

The following conventions are used throughout this guide:

- The notation ***<install_dir>/*** is used throughout this guide to refer to the location on your system where the Stellent Content Server product is installed.
- Forward slashes (/) are used to separate the directory levels in a path name. A forward slash will always appear after the end of a directory name.

❖ Notes, technical tips, important notices, and cautions use these conventions:

Symbol	Description
	This is a note. It brings special attention to information.
	This is a tech tip. It identifies information that can be used to make your tasks easier.
	This is an important notice. It identifies a required step or critical information.
	This is a caution. It identifies information that might cause loss of data or serious system problems.

STELLENT PRODUCT DISTINCTIONS

In this guide, the term *content server* is used generically to refer to both the Stellent Content Server and the Stellent Collaboration Server products. The following table lists the distinctions of these two Stellent content management solutions:

Table 1-1 Stellent Content Management Product and Feature Distinction

Product	Description
Stellent Content Server	A fully functional content management system providing end-to-end content management and personalized delivery of that content.
Stellent Collaboration Server	A fully functional content management system providing end-to-end content management and personalized delivery of that content. Additionally, a Stellent Collaboration Server license enables project-level security for collaborative authoring environments.

IF YOU NEED ASSISTANCE

The Stellent family of products is backed by a full range of support options to meet every business need. The service philosophy is to keep your Stellent environment fully operational by providing the best information and solutions available. The Stellent product support team consists of highly trained product engineers who excel at resolving complex technical issues. Every customer inquiry is tracked and managed through automated systems.

Important: The support options that are available for specific systems may vary, depending on the applicable service and maintenance agreements. Please refer to your contract for the support details for your Stellent system.

Support Options

You can choose from the following three support programs offered by Stellent:

- **Standard Maintenance and Support Program:** The standard support program is available during standard business hours domestically and internationally (Monday through Friday from 8 am to 5 pm for every time zone). It provides telephone and e-mail support for troubleshooting, bug fixes, call escalation, modifications, enhancements, and updates.
- **SDK Developer Support Program:** The SDK support program is available Monday through Friday from 8 am to 5 pm (Central Time in the USA, which is -6 hours from GMT). It provides telephone and e-mail support for customers who wish to use the Software Developer's Kit (SDK) to customize their Stellent systems.
- **Extended Support Program:** The extended support program provides the standard support services 24 hours a day and 7 days a week.

Before Contacting Support

When you call or send e-mail, please provide the following information:

- Nature and severity of the problem
- Stellent product and version
- Serial number of the registered Stellent product
- Name and telephone number of the person the support engineers should contact if they need to call back
- Operating system and version.

In addition, depending on the situation, it may be helpful to know the following:

- Database type and version
- Web browser type and version
- Web server type and version.

Telephone

From the U.S., technical support is available from the Support Hotline at 1-888-688-TECH (1-888-688-8324).

The Support Hotline is toll-free world-wide. Callers outside the U.S. need to dial a country-specific access code. To find the access code for the country you are dialing from, go to the following Internet address: http://www.att.com/international_business/dialing_guide/country-diallist.cgi

E-Mail

The Stellent support e-mail address is *support@stellent.com*. It is available for all technical support questions.

Internet

Technical support is also available through the Internet at *http://support.stellent.com*. You will be prompted for a username and password. To obtain a username and password, contact the Support Hotline at 1-888-688-TECH (1-888-688-8324).

Chapter

2

IDOC SCRIPT APPLICATION

Idoc Script is the custom scripting language for Stellent Content Server. It enables you to reference variables, conditionally include content in HTML pages, and loop over results returned from queries. In addition, Idoc Script provides a way to process page elements after the browser has made a request, but before the requested page is returned.

Idoc Script is used primarily for the presentation of HTML templates, not for functionality. The code is defined as queries in the query tables, as execution script defined in the application service scripts, or in the Java class files.

This chapter includes the following sections:

IDOC SCRIPT TAGS

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

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

Tech Tip: If you are using Idoc Script in an HCSP or HCSF page, you must use the syntax <!--$ … --> for Idoc Script tags. For more information, see Chapter 10, Dynamic Server Pages, in the *Customizing Content Server* guide.

IDOC SCRIPT USES

There are five basic uses for Idoc Script:

- *Includes (page 2-3)* enable you to reuse pieces of Idoc Script and HTML code.
- *Variables (page 2-7)* enable you to define and substitute variable values.
- *Conditionals (page 2-9)* enable you to evaluate *if* and *else* clauses to include or exclude code from an assembled page.
- *Looping (page 2-14)* enables you to repeat code for each row in a ResultSet that is returned from a query.
- *Utilities (page 2-17)* allow you to use Idoc Script in the Web Layout Editor, Workflow Admin, and Batch Loader tools.

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 content server leverages includes very heavily; they are used on nearly every page of the content server web site.

Note: Includes make it easier for you to customize your content server instance using component architecture and dynamic server pages. For more information on includes and customization, see the *Customizing Content Server* guide.

- 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 <*install_dir*>/shared/config/resources/std_page.htm file.

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 code is located in the *<install_dir>*/shared/config/resources/std_page.htm file:

```
<@dynamichtml body_def@>
      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@>
```

Figure 2-1 Example of an include definition (from the *<install_dir>*/shared/config/resources/std_page.htm file).

Most of the standard template resource files (for example, *<install_dir>*/shared/config/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.

Super Tag

The *super* tag is used to define exceptions to an existing HTML 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, if you want 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, make sure 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 Stellent Content Server, the super tag ensures that your components are using the most recent version of the include, modifying only the specific code you need to customize your instance.

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@>
```

Variables

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

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

  `<$variable_name$>`

- 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*.

- 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 `<$b=2$>`.

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

  `variable_name=value`

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

- Some variables are predefined in the content server, while other variables can be generated by queries and used by services.

  For example, if the results of the query `SELECT dOption FROM OptionsList WHERE dKey = 'xRegionList'` are assigned to a ResultSet named REGIONS, an entry in the Region option list could be referenced using `<$dOption$>`.

Note: For more information on the DataBinder, see Chapter 4, HDA and HTM File Types, in the *Customizing Content Server* guide.

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

Note: For more information on the DataBinder, see Chapter 4, HDA and HTM File Types, in the *Customizing Content Server* guide.

Regular Variables

A regular variable that does not have special evaluation logic is equivalent to using the *#active (page 2-18)* 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 content server debug output.

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 variable or Idoc Script function.
- *Boolean Operators (page 2-23)* 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 *#active (page 2-18)* prefix. The result is *true* if the value is not null and is either a nonempty string or a nonzero integer.

Conditional Example 1

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

```
<$if xRegion$>
   <td><$xRegion$></td>
<$else$>
   <td>Region is not defined</td>
<$endif$>
```

- If the value of *xRegion* is defined, then the table cell contains the value of *xRegion*.
- If the value of *xRegion* is not defined, a message is written as the content of the table cell.

Conditional Example 2

In this example, the look-and-feel of the content server web pages depends on the value of a variable set in the configuration file.

1. To change the look of the web pages, you set the appropriate variable in the *config.cfg* file:

Look-and-Feel	Example	Variable Setting
Stellent		None (this is the default)
Bellevue		`UseBellevueLook=true`
Xpedio		`UseXpedioLook=true`

2. A standard include named *std_page_variable_definitions* in the *std_page.htm* file evaluates the configuration settings as follows:

   - The code first evaluates if the *UseXpedioLook* variable is defined and is true. If so, the *xpedioLook* variable is set to 1, and the rest of the conditional clause is ignored.
   - If the `<$if$>` statement is not satisfied, the code evaluates if the *UseBellevueLook* variable is defined and is true. If so, the *bellevueLook* variable is set to 1, and the rest of the conditional clause is ignored.
   - If the `<$elseif$>` statement is not satisfied, the *stellentLook* variable is set to 1.

```
<$if UseXpedioLook and isTrue(UseXpedioLook)$>
   <$xpedioLook=1$>
   <$mailColumnHeaderColor="#000000",
   highlightFieldColor="#600060",errorHighlightColor="#A
   00060", strongHighlightFieldColor="#A00060",
   revLabelItemColor="#F000F0",
   reportColumnHeaderColor="#000000",
   configLabelColor="#600060", pageTitleColor="#800080",
   requiredFieldColor="#800080",
   navHeaderTextColor="#ffffff",
   pneCellBackgroundColor="#705B92", pne_nav_width=208,
   ProductID="Xpedio"$>
<$elseif UseBellevueLook and isTrue(UseBellevueLook)$>
   <$bellevueLook=1, ProductID="Xpedio"$>
<$else$>
   <$stellentLook=1, resizableTopBanner=1,
   ProductID="Stellent"$>
<$endif$>
```

Figure 2-2 The code that evaluates the configuration settings for the look-and-feel of the content server web site.

3. A standard include named *define_image_files* in the *std_page.htm* resource file defines three different sets of graphics that create an overall look-and-feel for the web site.

   - The code first evaluates if the *xpedioLook* variable is defined. If so, the first set of images (the Xpedio look) is used to assemble the web pages, and the rest of the conditional clause is ignored.
   - If the `<$if$>` statement is not satisfied, the code evaluates if the *bellevueLook* variable is defined. If so, the second set of images (the Bellevue look) is used to assemble the web pages, and the rest of the conditional clause is ignored.
   - If the `<$elseif$>` statement is not satisfied, the third set of images (the Stellent look) is used to assemble the web pages.

```
<$if xpedioLook$>
   <$button_enthome_color_image="xpedio/enthome2.gif",
   button_enthome_grey_ish_image="xpedio/enthome.gif",
   button_home_color_image="xpedio/home2.gif",
   ...
<$elseif bellevueLook$>
   <$pneCellBackgroundImage="bellevue/pne_cellbkg.gif"$>
   <$button_enthome_color_image="bellevue/enthome2.gif",
   button_enthome_grey_ish_image="bellevue/enthome.gif",
   ...
<$else$>
<!-- stellent look -->
   <$pneCellBackgroundColor="#000000",
   pneCellBackgroundImage="stellent/stepbkg.gif"$>
   <$button_enthome_color_image="stellent/enthome2.gif",
   button_enthome_grey_ish_image="stellent/enthome.gif",
   ...
<$endif$>
```

Figure 2-3 The code that defines the graphics for the content server web site.

Looping

Loop structures allow you to repeat code for each row in a ResultSet that is returned from a query. Looping can be accomplished in three ways with Idoc Script:

ResultSet Looping

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

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

- When inside a ResultSet loop, you can reference any column of the ResultSet.
- Substitution of values depends on which row is currently being accessed in the loop.
- The code between Idoc Script tags is repeated once for each row in the ResultSet.
- When inside a ResultSet loop, that ResultSet becomes active and has priority over other ResultSets when evaluating variables and conditional statements.

ResultSet Looping Example

In this example, an option list entry is created for each row returned as part of the ResultSet named INFOTYPES:

```
<SELECT NAME="ufotype" onChange="isAddTextToField(this,
this.form.xINFO_Type)">
   <$loop INFOTYPES$>
      <OPTION><$xINFO_Type$>
   <$endloop$>
</SELECT>
```

Figure 2-4 Example of a ResultSet loop.

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.

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.

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

Figure 2-5 Example of a while loop.

Option List Looping

Option list looping enables you to loop over the contents of an option list. The name of the option list to be looped is specified as a variable using the special `<$optList$>` Idoc Script function:

```
<$optList variable$>
```

For example, you can:

- Generate a list of possible authors using `<$optList docAuthors$>`
- Generate a list from a custom metadata field set up as an option list, using `<$optList xFieldName.options$>`

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.

Utilities

You can use Idoc Script in the following content server utilities:

- *Web Layout Editor (page 2-17)*
- *Workflow Admin (page 2-17)*
- *Batch Loader (page 2-17)*

Web Layout Editor

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

For example, to set the search results to return all content items up to 30 days, define the search query to be:

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

Workflow Admin

In the Workflow Admin tool, you can use Idoc Script to define step events, jumps, and user tokens.

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

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

Batch Loader

In the Batch Loader, you can use Idoc Script in a mapping file, which tells the Spider utility how to determine the metadata for file records.

SPECIAL KEYWORDS

The following keywords have special meaning in Idoc Script:

Keyword	Example	Description
#active	<$#active.*variable*$>	Retrieves the value from the DataBinder, searching in the following default order: local data, active ResultSets, all other ResultSets, environment. Does not send an error report to the content server debug output if the variable is not found.
#local	<$#local.*variable*$>	Retrieves the value from the local data only. Sends an error report to the content server debug output if the variable is not found.
#val	<$#val(*variable*)$>	Used in applets to recursively evaluate the values of a variable.

Keyword	Example	Description
exec	<$exec *expression*$>	Executes an expression and suppresses the output (does not display the expression in the output file). **Note:** 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 (page 2-3)* for more information.
super	<$include super.<*include*>$>	Starts with the existing version of the include code. See *Super Tag (page 2-5)* for more information.

OPERATORS AND WILDCARDS

Idoc Script supports a number of operators and wildcards. This section describes the following:

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:

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

Important: 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 (page 2-21)*.
- To perform advanced string operations, use *strEquals (page 3-38)*, *strReplace (page 3-46)*, or other string-related global functions.

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 string on the right of the operator can use *Wildcards (page 2-24)*. ❖ This operator is not case sensitive.	`<$if b like "car\|truck"$>`
|	The **string inclusion operator** separates multiple options, performing a logical “OR” function.	`<$if a like "cart"$>`

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 (page 3-38)*, *strReplace (page 3-46)*, or other string-related global functions.

Binary Operators

Use the following binary 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.	`<$a=(b+2)$>`
-	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.	`<$a=(b%2)$>`

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.
- Boolean operators can be used to evaluate strings. If the string is defined, it will be evaluated as true. If the string is NULL it will be evaluated as false.

```
<$MyString="false"$>
<$if MyString$>
... this conditional will always be true...
<$endif$>
```

Wildcards

Idoc Script 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*

PRE-DEFINED VARIABLES

Special variables can be used to gather information about the current template or the user currently logged in. These variables are read-only and cannot be assigned a value.

Template-Related Variables

Template-related 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.

Table 2-1 Template-Related Variables

Variable	Description
<$TemplateClass$>	The classification of the template. ❖ For standard templates, this variable is defined in the *class* column of the *IntradocTemplates* table in the *templates.hda* file. ❖ For search result templates, this variable evaluates to `Results`. ❖ For report templates, this variable evaluates to `Reports`.
<$TemplateFilePath$>	File location from which the template was actually loaded. For example, `C:/stellent/shared/config/templates`.
<$TemplateName$>	The internal name of the template (for example, `DOC_INFO` or `CHECKIN_NEW_FORM`).
<$TemplateType$>	The type of the template. ❖ For standard and search results templates, this variable is defined in the *formtype* column of the *IntradocTemplates* 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.

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, the pre-defined *TemplateName* variable (shown in bold below) was added to the *pne_nav_admin_links* include that defines the Administration links:

```
<$if IsSubAdmin$>
<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>
<$endif$>
```

Figure 2-1 Example of using the *TemplateName* pre-defined variable to display the internal template name on a web page.

User-Related Variables

User-related variables make it possible to gather information about the user currently logged in.

Variable	Description
<$DelimitedUserRoles$>	A comma-separated, colon-delimited list of roles the current user belongs to. For example: `PublicContributor,ClassifiedConsumer`
<$UserAccounts$>	A comma-separated list of accounts the current user has access to. For example: `London,ParisSales,RomeSales`
<$UserAddress$>	The e-mail address of the current user.
<$UserDefaultAccount$>	The default account of the current user.
<$UserFullName$>	The full name of the current user.
<$UserName$>	The user name of the current user. If no user is logged in, evaluates to `anonymous`.
<$UserRoles$>	A comma-separated list of roles the current user belongs to. For example: `PublicContributor,ClassifiedConsumer`

IDOC SCRIPT PAGE FLAGS

Flags Affecting Page Display

Page related flags are set to enable specific page attributes or functionality. These are flags set on the page and are not defined as global variables:

defaultFieldInclude	isExcluded	isInfoOnly
defaultOptionListScript	isDocPage	isQuery
DocURL	isEditMode	isRelocated
fieldCaptionInclude	isFieldExcluded	isStrictList
fieldCaptionStyle	isFieldHidden	isUpdate
fieldEntryInclude	isFieldInfoOnly	isUploadFieldScript
fieldValueStyle	isFieldMemo	optionListName
fileURL	isFormSubmit	optionsAllowPreselect
hasOptionList	isHidden	useAllVisibleFields
isCheckin	isInfo	

These field flags can be set at the top of any page as local variables.

- The flags *isExcluded*, *isHidden* and *isInfoOnly*, must be placed in the template file before the <HTML> tag. Placing them after the <HEAD></HEAD> section can result in the field indeed not showing up, but the validation code for that field will still be in the header. On check-in you then typically get a "… is not an object" error.

- If you wish to use the *include* tag to insert you own custom html code for a special field, make sure you place the statement after the </HEAD> tag. If you place it before the </HEAD> tag, the content server will insert your custom html code into the header between the form validation code and attempt to read it as JavaScript. This will cause the Internet Explorer browser to fail at check-in and cause the validation on check-in to generate syntax errors.

Flag	**Usage**
isHidden	The field is incorporated on the checkin and query pages but is not visible to user. Setting the local variable 'dDocTitle' to a value will set the hidden title to a particular value.
isInfoOnly	The field presented on the page will be display only, not an input field.
isExcluded	The field is completely excluded from the form.
isRelocated	The field is excluded unless the local variable isRelocated is set to 1 while the resource include is evaluated. This allows the field to be on the page a second time with a custom location using with the variable isRelocated set to 1

Flag	Usage
optionListScript	If set to a non empty value then “eval(…)” function is used when displaying the option list for the field. This allows the standard implementation of the option list to be overridden. Note: the option list is displayed on both the checkin/update and query pages. The default implementation can be referenced using the variable defaultOptionListScript.

Flag	Usage
include	If the previous options are not sufficient to provide all the flexibility needed, the entire implementation of the field can be replaced by setting this variable to name of a resource include that should be used instead. The implementation being replaced can be referred to by the variable defaultFieldInclude whose value will be different depending on whether the field is being shown on a checkin/update, query, or info page. It will also vary considerably based on the type of field being displayed. If the field include is overridden, then the new implementation must deal with all the issues of the different pages including dealing with 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 the custom functionality executed (in particular look at the resource include 'std_field_entry').

Variables Affecting Page Display

These Idoc Script variables affect specific page display and should not be edited unless absolutely needed.

Important: Altering these variables in any include will change the way all metadata is displayed on the page. .

Flag	Usage
isCheckin	Set if the current page is a checkin page.
isQuery	Set if the current page is a search page.
isInfo	Set if the current page is a doc info page.
isDocPage	Set for checkin, search, search results, query, and update pages.
isUpdate	Set if the current page is a doc info update page.
isFormSubmit	Set if the metadata field is part of a form.

Flag	Usage
isEditMode	Set if the metadata field is editable on a form (checkin and update pages only).
isUploadFieldScript	A flag set when the metadata include is being used inside JavaScript to determine how the fields are uploaded.
useAllVisibleFields	Set if all fields that are not “isExcluded” are to be displayed.

Flags and Variables Affecting Metadata Display

These flags and variables alter the display of metadata. They can be set without affecting other metadata fields on the page: .

Flag	Usage
defaultFieldInclude	The name of the include to use, with the above variables, to determine how to display the metadata (for example, std_namevalue_field, std_file_entry).
fieldCaptionInclude	The name of the include to use, with the above variables, to determine how to display the caption for the metadata (for example, std_field_caption).
fieldEntryInclude	The name of an Idoc Script include to be used when displaying the value of the metadata field on the html page (for example, std_field_entry, std_memo_entry).
isFieldMemo	Set if the field is a memo field.
isFieldInfoOnly	Set if the metadata field value is only for display, not editing (see isInfoOnly).
isFieldHidden	Set if the metadata field is not displayed on the page (see isHidden).

Flag	Usage
isFieldExcluded	Set if the metadata field is completely excluded from the page, or is relocated and not displayed in the normal place (see isExcluded and isRelocated).
hasOptionList	Set if the metadata field is an option list.
isStrictList	Set if an option list is strict, and not a multiselect.
optionsAllowPreselect	Set if you want a metadata option list to be prefilled with its last value (for example, update and new revision checkin pages).
optionListName	The name of the include to use for the option list (default to the value fieldName.options).
defaultOptionListScript	A piece of Idoc Script used to display the option list.
fieldCaptionStyle	The style on a checkin or search page of the caption for the metadata.
fieldValueStyle	The style on an update page of the value for the metadata.

Local Page Variables

The following variables affect only the local page that they are defined on:

Variable	Usage
DocUrl	Used on the Content Information page to display the URL for the content.
fileUrl	Can be used within a dynamic server page (.hcsp or .hcst) to refer to the current page.

COMMON METADATA AND CONTENT INFORMATION FIELD NAMES

The content information field names used within the content server are descriptive captions not actual metadata terms. The actual metadata name should be used when batch loading files or scripting .hcst, .hcsp, and .hcsf pages. The content information field names should be used when displaying content to the user.

Referencing Standard Metadata

This is a cross reference of the standard metadata field names used in the system:.

Caption	Internal Usage
Account	dDocAccount
Author	dDocAuthor
Content ID	dDocName
Title	dDocTitle
Type	dDocType
Release Date	dInDate
Revision	dRevLabel
Security Group	dSecurityGroup
Comments	xComments

Note: Do not confuse the Content ID (dDocName) with the internal content item revision identifier (dID). The *dID* is a generated reference to a specific rendition of a content item.

Creating Custom Metadata

When adding custom metadata append a lowercase 'x' to the field label created within the Configuration Manager.

For example, a custom metadata field named Department created inside the database (and when accessing it via Idoc Script or for batch loading) will be known as *xDepartment*.

Important: In all cases field names are case sensitive.

Chapter

3

IDOC SCRIPT GLOBAL FUNCTIONS

OVERVIEW

Idoc Script has a number of built-in functions. Functions perform actions and also can return results. Sometimes these are the results of calculations or comparisons. Idoc Script functions perform various string comparison and manipulation routines, date formatting and ResultSet manipulation.

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 or arguments. 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.

Some of the most commonly used functions are those relating to string manipulation and retrieving the current system date.

FUNCTION DESCRIPTIONS

abortToErrorPage

Description

Aborts the current page and displays an error message.

- Takes one parameter as String. This parameter is the error message to display.
- Some functions set a status code parameter to a negative numeric value if they fail. This status code is placed in the Idoc Script variable *StatusCode*. The function evaluates the status code and if a negative numeric value is returned the display of the current page is aborted and is substituted with an error page.
- The variable *StatusMessage* holds the currently generated error message from the most recent function call that enables the setting of the status code.

See Also: executeService (3-9)

getUserValue (3-17)

hasAppRights (3-20)

IsRequestError (4-67)

Example

Aborts the current page and displays *Access Denied* as an error message:

```
<$abortToErrorPage("Access Denied")$>
```

break

Description

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.

Example

N/A

computeRenditionUrl

Description

Returns the URL of the rendition.

- Takes three arguments.
- This function takes as arguments the URL of the file, the dRevLabel value, and the dRendition1 value.
- This function, given the URL of the file, returns the URL of the rendition as String.

Example

Returns the URL of the rendition as a String.

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

dateCurrent

Description

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.
- Optional parameter can adjust the date by the number of days.
- Returns a formatted date with the output:

  ```
  m/d/yy h:mm XM
  ```

- Returns a two digit year and month/day without leading zero.

Example

Returns the current date and the current time (for example, 9/12/01 1:55 PM):

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

Returns the date ten days in the future and the current time (for example, 9/22/01 1:55 PM):

```
<$dateCurrent(10)$>
```

Returns the date ten days in the past and the current time (for example, 9/2/01 1:55 PM):

```
<$dateCurrent(-10)$>
```

docLoadResourceIncludes

Description

Loads *resource includes* for display on the current page.

- This function loads all the *resource includes* in the specified content item for use in the display of the current page. The content item specified must have the file extension *idoc*.
- Takes a CGI encoded parameter list referring to a content item controlled by the content server.
- 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.

See Also: abortToErrorPage (3-2)

Optional Parameters

The optional parameters, in the following table, may be defined:

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*.

Optional Parameters	Description
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.
RevisionSelection Method	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 rendition. ❖ Latest: The latest revision of the content item 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.

Example

Loads the *resource includes* in the primary vault rendition of the latest revision of *mydoc*.

```
<$docLoadResourceIncludes("dDocName=mydoc&RevisionSelection
Method=Latest")$>
```

Note: When used in .hcsp pages, the ampersand (&) character in the CGI encoded parameter must be changed to the & characters.

docUrlAllowDisclosure

Description

- ❖ Evaluates whether a URL can be disclosed to the user.
- ❖ Relates to the currently logged on user. Takes one parameter. Takes as a parameter an absolute path or a full relative path (see example). Determines if the passed in URL can be disclosed to the user.
- ❖ Returns a Boolean value.
  - Returns TRUE if URL can be disclosed.
  - Returns FALSE if URL is restricted.

Example

Searches the first column of *resultSet1* until a value matching *value1* is found.

```
<$docUrlAllowDisclosure("/<home>/groups/documents/mydocumen
t.pdf")$>
```

eval

Description

Evaluates the parameter as if it were Idoc Script.

- Recursively evaluates a literal string.
- Returns Idoc Script enclosed within a string as a variable.

Example

Variable *one* is assigned the string *CompanyName*, 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>
<$eval(two)$>
```

executeService

Description

This function executes a content server service.

- This function enables a specified service to be executed while the page is being constructed. Only a small subset of “read only” services can be called using this function. Generally, services are executed using a tool such as IdcCommand or the CGI URL on the browser to execute the services.
- 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.
- Takes one argument. This function has no output. All output of the service is suppressed but any result sets and values loaded are available. The current live data in the DataBinder being displayed is used as parameters to the service.
- Returns TRUE if the service was executed.
- Returns FALSE if an error in executions occurred.
- Returns the status code "-1" if the value is unspecified or unknown.

Example

Executes a service when given a service name:

```
<$executeService("servicename")$>
```

formatDate

Description

Formats and displays date and time.

- Given a date and time in an alternate format, displays the date and time in the format used by `dateCurrent` (for example, 7/12/00 1:55 PM).
- Returns null if date cannot be evaluated.
- Database formatted dates cannot be evaluated (for example, 2000-02-02).
- Long formatted dates cannot be evaluated (for example, June 12, 2001).
- If a time is not provided, returns default time of 12:00 AM.

Example

Formats the date and time and displays as 12/14/99 2:00 PM:

```
<$formatDate("12/14/1999 02:00 PM")$>
```

Formats the date, assigns the default time, and displays as 2/2/00 12:00 AM:

```
<$formatDate("02/02/2000")$>
```

This 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 (for example, Final Approval: 1/4/00 3:05 PM):

```
<$my_customDateTime = formatDate("01/04/2000
15:05:34")$>
Final Approval: <$my_customDateTime$>
```

formatDateDatabase

Description

Formats the date and time for an SQL query.

- Given a date, formats the date and time in preparation for an SQL query.
- Returns an ODBC formatted date with the output `[ts 'yyyy-mm-dd hh:mm:ss']`.
- Returns null if date cannot be evaluated.
- Long formatted dates cannot be evaluated (for example, May 22, 2000).
- If a time is not provided, returns 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:

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

formatDateOnly

Description

Displays the date.

- Given a date and time, formats it to show only the date. Displays the date in the format used by `dateCurrent` (for example, 7/12/00).
- Returns null if date cannot be evaluated.
- Database formatted dates cannot be evaluated (for example, 2000-02-02).
- Long formatted dates cannot be evaluated (for example, May 22, 2000).
- Returns a two digit year and month/day without leading zero.

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):

```
<$my_startDate = formatDateOnly(currentDate())$>
<$my_endDate = formatDateOnly(currentDate(100))$>
Start Date: <$my_startDate$> and End Date:
<$my_endDate$>
```

formatDateOnlyFull

Description

Displays the date in long format.

- Given a date and time, formats to show only the date. Displays the date in long format.
- Returns a formatted date with the output:

  ```
  month d, yyyy.
  ```

- Returns null if date cannot be evaluated.
- Database formatted dates cannot be evaluated (for example, 2000-02-02).
- Returns a four digit year and a date without leading zero.

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:

```
<$formatDateOnlyFull("6/12/01 3:00 PM")$>
```

Formats the date only and displays as February 2, 1999:

```
<$formatDateOnlyFull("02/02/99")$>
```

formatTimeOnly

Description

Displays the time.

- Given a date and time, formats it to show only the date. Displays the time in the format used by `dateCurrent` (for example, 1:55 PM).
- Returns a formatted time with the output:

  `h:mm XM`
- Returns null if time 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:

```
<$formatTimeOnly("04/21/2001 18:14:00")$>
```

getDebugTrace

Description

Retrieves the output of the debug trace.

- This function takes no parameters.
- The function returns an empty string if *ScriptDebugTrace* has not been set.
- Returns the output of the accumulated debug trace for the page being constructed.

See Also: ScriptDebugTrace (4-74)

Example

Retrieves the output of the debug trace and outputs the information to a page:

```
<$getDebugTrace()$>
```

getErrorTrace

Description

Retrieves the output of the error trace.

- This function takes no parameters.
- Returns the output of the accumulated error trace for the page being constructed.
- Output has been encoded for display in HTML pages. For example, the < and > delimiters are HTML escaped and carriage returns converted to <*br*> tags. The function returns an empty string if *ScriptErrorTrace* has not been set.
- Returns the output of the accumulated error trace for the page being constructed.

See Also: ScriptErrorTrace (4-75)

Example

Retrieves the output of the error trace and outputs the information to a page:

```
<$getErrorTrace()$>
```

getUserValue

Description

Retrieves the user-related value.

- Relates to the currently logged on user. This function returns the value of a user metadata field for the user currently logged in.
- The parameter must refer to a column in the Users 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 a group of content servers.
- Takes one argument. Any of the user-related variables can be used as an argument.
- Returns TRUE if the user value was retrieved.
- Returns FALSE if an error in retrieval occurred.
- Returns the status code "-1" if the value is unspecified or unknown.

See Also: User Related Variables (2-28)

Example

Returns the user entry of the currently logged on user:

```
<$getUserValue('dUserType')$>
```

getValue

Description

The getValue function can be used to either do a general search for a particular field, or request specific information about a given ResultSet.

- Takes two arguments.
- If the value is not found an empty string is returned.

The function getValue has the following variations:

Variation	Description
getValue ("#local",fieldName)	The value is retrieved from the local data.
getValue ("#active",fieldName)	Attempts to retrieve the value from the data in the following order: active ResultSets, local data, all other ResultSets and finally the environment.
getValue (ResultSetName,fieldName)	Searches the current row of the named ResultSet for the fieldName.

A shorthand variable format that returns identical results can be used. The format uses the form <$arg1.arg2$>, where arg1 and arg2 are the literal string arguments "*arg1*" and "*arg2*" to getValue.

Additional information about a resultset can be retrieved using getValue.

Current Row:

```
<$SearchResults.#row$>
```

Number of total rows:

```
<$SearchResults.#numRows$>
```

Test to see if a row is present, useful if looping manually with *rsNext*.

```
<$SearchResults.#isRowPresent$>
```

Test to see if a search results has any rows present:

```
<$SearchResults.#isEmpty$>
```

Retrieve data specifically out of the environment:

```
<$#env.fieldName$>
```

Example

Get the content item name from the ResultSet named “DOC_INFO”:

```
<$name = getValue("DOC_INFO","dDocName")$>
```

Check to see if the passed parameter dDocType (that is in the local data) is the same as the value in the active result set:

```
<$loop DocTypes$>
<$if strEquals(#active.dDocType,
getValue(#local,dDocType))$>
<!--do special HTML for selected document type-->
<$endif$>
   <!-- additional statement-- <$endloop$>
```

hasAppRights

Description

Retrieves the user rights of the specified application.

- Relates to the currently logged on user.
- Takes the name of an application as argument such as *UserAdmin, WebLayout, RepoMan, Workflow, ConfigMan,* or *Archiver*.
- Returns TRUE if the user has rights to the application.
- Returns FALSE if the user does not have rights to use the application.
- Returns the status code "-1" if the value is unspecified or unknown.

See Also: *Includes (page 2-3)*

Example

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

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

inc

Description

Displays the resource include name.

- Given a name of a resource, includes it into the page for a presentation.
- Returns the resource include as string.

Example

Displays the *std_header* resource include.

```
<$inc("std_header")$>
```

incGlobal

Description

This function adds the specified global include file to the page.

- Takes one argument. This function takes as an argument the name of a global include file without the file extension.
- Global includes files have a .inc extension. The argument must reference a file with that extension. However, do not include the file extension along with the global include name within the argument.
- See the directory <*home*>/data/pages/ for examples several existing Global include files.

Example

Retrieves the global include file *portal_message.inc* and adds it to the page.

```
<$incGlobal("portal_message")$>
```

incTemplate

Description

This function returns the contents of a template after evaluating any Idoc Script.

- Takes one parameter. This function takes as a parameter the name of a content server template without the file extension.
- A page can use this function to include the content of an entire template. This usage is normally discouraged because resource includes are usually sufficiently flexible to support all needs for the sharing of Idoc Script between pages.

Example

Retrieves the *new_look* template file.

```
<$incTemplate("new_look")$>
```

isFalse

Description

The function returns FALSE with specified parameters.

If the function parameter is a string:

- Returns TRUE if the string begins with a F, f, N, n (or 0).

If the function parameter is not a string:

- Returns FALSE if the value is not 0.
- Returns TRUE if the value is zero.

See Also: isTrue (3-23)

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):

```
<$isFalse(strEquals("abc","abc"))$>
```

isTrue

Description

The function returns TRUE with specified parameters.

If the function parameter is a string:

- Returns TRUE if the string begins with a T, t, Y, y (or 1).

If the function parameter is not a string:

- Returns TRUE if the value is not 0.
- Returns FALSE if the value is zero.

See Also: isFalse (3-22)

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):

```
<$isTrue(strEquals("abc","abc"))$>
```

isUserOverrideSet

Description

Checks whether the user override is set.

- Used when a non-author of a content item is allowed to check in for others.
- When set to TRUE, enables a non-author to check in content items using the name of another author. Affects the pull-down list of authors and is generally used by the administrator.
- Returns TRUE if override is enabled.

Example

Enables a non-author to check in content items using the name of other authors:

```
<$isUserOverrideSet(true)$>
```

js

Description

Converts a string for use with JavaScript.

- Takes one parameter.
- Given a string, converts it for use in a "..."JavaScript literal string declaration. Performs string manipulation such as changing double quotes to single quotes.

Example

Formats the string *variablestring* for use in a JavaScript string declaration:

```
<$js("variablestring")$>
```

loadDocMetaDefinition

Description

Loads a table as a result set.

- The function has no output and takes no parameters.
- Loads the table *DocMetaDefinition* into the active data as a result set. After it is loaded it can be looped on.

Example

Loads *DocMetaDefinition* table into the active data as a result set.

```
<$loadDocMetaDefinition()$>
```

loadCollectionInfo

Description

Enables collection information loading.

- Used by the search service in the metadata fields stored in the collection. Enabled as part of installation process. Not intended for user configuration.
- No return value specified.

Example

Loads collection information:

```
<$loadCollectionInfo()$>
```

optList

Description

Function for iterating over the contents of an option list.

- Passes in the option list intended for display.
- Used for the building of HTML selection lists. This is a built-in function for iterating over the contents of an option list.
- The optList function is used extensively in files such as: */config/resources/std_page.htm*.

Example

This script generates a list of possible authors:

```
<$optList docAuthors$>
```

This script generates a list of Regions. This was created as a custom content item information field set up as an option list:

```
<$optList xRegion.options$>
```

pneNavigation

Description

Enables the sidebar navigation.

- This display resource is enabled by default using a flag is set to the numeric value 1. To disable this flag, the definition must be set to a null string.
- Default is 1 (enabled).

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:

```
<$pneNavigation=0$>
```

parseDate

Description

Allows date and time manipulation and arithmetic.

- Parses a string and creates a date object formatted for the locale. This function parses the date and time and make it possible to do date/time manipulation and arithmetic.
- A common usage is to use the current time in a time multiplication expression with the milliseconds in a second, seconds in a minute, minutes in a hour, hours in a day, and days in a year.

Example

Returns the date and time one day in the past:

```
<$parseDate(dateCurrent())-(1000*60*60*24)$>
```

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:

Note: The `dOutDate` specifies the content item expiration date and is an optional parameter for the CHECKIN_UNIVERSAL service.

```
<$if dOutDate$>
   <$if dateCurrent(7) > parseDate(dOutDate)$>
   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:

```
<$if parseDate(wfCurrentGet("lastEntryTs")) <
dateCurrent(-4)$>
      <$wfSet("wfJumpName", "lateJump")$>
      <$wfSet("wfJumpTargetStep", "step_1@wf_late")$>
      <$wfSet("wfJumpReturnStep", wfCurrentStep(1))$>
      <$wfSet("wfJumpEntryNotifyOff", "0")$>
<$endif$>
```

proxiedBrowserFullCgiWebUrl

Description

Returns the complete CGI path of the proxied server.

- Returns a string. Takes one parameter.
- This function takes as a parameter the relative web root. The relative web root is found in the Idoc Script Predefined Variable *HttpRelativeWebRoot*. This function returns the complete CGI path for the proxied server when called by the master instance.

See Also: HttpRelativeWebRoot (5-128)

Example

Returns the complete CGI path for the proxied server as a string.
For example, *http://<localhost>/intradoc-cgi/idc_cgi_isapi.dll/intradoc4/pxs*.

```
<$proxiedBrowserFullCgiWebUrl("/intradoc4/")$>
```

proxiedCgiWebUrl

Description

Returns the CGI path of the proxied server.

- Returns a string. Takes one parameter.
- This function takes as a parameter the relative web root. The relative web root is found in the Idoc Script Predefined Variable *HttpRelativeWebRoot*. This function returns the CGI path for the proxied server when called by the master instance.

See Also: HttpRelativeWebRoot (5-128)

Example

Returns the CGI path for the proxied server as a string.
For example, */intradoc-cgi/ idc_cgi_isapi.dll/intradoc4/pxs*.

```
<$proxiedCgiWebUrl("/intradoc4/")$>
```

rsFindRowPrimary

Description

Searches the first column of a result set.

- Takes two parameters. The first parameter designates the name of a result set whose first column is to be searched. The second parameter is a value to be used for comparison. The first column of the specified result set is searched until a matching value is found.
- Returns TRUE on success.
- Returns FALSE if the ResultSet is empty.

Example

Searches the first column of *resultSet1* until a value matching *value1* is found.

```
<$rsFindRowPrimary("resultSet1","value1")$>
```

rsDocInfoRowAllowDisclosure

Description

Evaluates whether the user can view the referenced URL.

- Relates to the currently logged on user. Takes one parameter. Takes as a parameter a result set whose current row has information about a content item.
- This function determines if the current user can view the URL of the content item referenced by the current row. This function is useful for selectively showing the URLs of a result set generated by a content item query.
- Returns FALSE if the ResultSet is empty.

Example

Determines whether the user can view the URL referenced in the current row of *resultSet1*.

```
<$rsDocInfoRowAllowDisclosure("resultSet1")$>
```

rsFirst

Description

Moves the ResultSet to the first row.

- Given the name of a ResultSet, moves the ResultSet to the first row.
- Returns a Boolean value:
  - Returns TRUE on success.
  - Returns FALSE if the ResultSet is empty.

Example

Advances the ResultSet called “SearchResults” to the first row:

```
<$exec rsFirst ("SearchResults")$>
```

rsNext

Description

Advances the ResultSet to the next row.

- Given the name of a ResultSet, advances the ResultSet to the next row.
- Returns TRUE on success.
- Returns FALSE if the ResultSet is empty.

Example

Advances the ResultSet called “SearchResults” to the next row:

```
<$exec rsNext ("SearchResults")$>
```

rsSetRow

Description

Sets the ResultSet to the specified row.

- Given the name of a ResultSet and an index, sets the ResultSet to the specified row.
- Returns a Boolean value:
  - Returns TRUE on success.
  - Returns FALSE if the ResultSet is empty.

Example

Starts the ResultSet called “SearchResults” at the 10th row:

```
<$exec rsSetRow("SearchResults",10)$>
```

setResourceInclude

Description

Enables a *resource include* to be assigned dynamically constructed script.

- This function has no output.Takes two arguments. The first parameter designates a *resource include,* the second a string contains Idoc Script.
- Enables a *resource include* to be assigned dynamically constructed script (much as the *eval* function enables such script to be parsed and evaluated). If the include exists, then it can be referred in the new script by using the *super* keyword.

Example

Uses the string “My name is *resource include*” to dynamically construct script:

```
<$setResourceInclude("my_name","My name is <$my_name$>")$>
```

stdSecurityCheck

Description

Performs a standard security check of the user.

- This function assumes that the active data holds the content item information for a content item.
  - The active data is used to determine if the standard (or default) security model enables the user to have access to the content item.
  - Access is based on the privilege or security level of the content item being requested. This enables a custom implementation of security to still execute the standard security model as a baseline.
- Returns a Boolean value and has no parameters.
- Relates to the user currently logged in.
- Used only for conditional disclosure.

Example

Compares the permission level of the user to the requested content item:

```
<$stdSecurityCheck()$>
```

strCenterPad

Description

Pads equal space on both sides.

- Pads equal spaces on each side of text to make it at least the specified string length. A character will be added to the length if required. Takes a string and a length as parameters.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

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>:

```
<$strCenterPad("inf", 9)$>
```

strConfine

Description

Confines a string and appends padding.

- Confines a string to a maximum length. Takes a string and a length as parameters. The character used for padding can be specified by altering the configuration variable *StrConfineOverflowChars*.
- If the string is shorter than the defined length, it is unaffected.
- If the string is longer than the defined length, it is shortened and extra characters are appended.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Confines the string to a maximum of ten characters in length, this six character string is unaffected:

```
<$strConfine("inform", 10)$>
```

Confines the string to a maximum of six characters in length, this six character string is unaffected:

```
<$strConfine("inform", 6)$>
```

Confines the string and appends padding to make it a string five characters long, using the form in<dot><dot><dot>:

```
<$strConfine("inform", 5)$>
```

strEquals

Description

Evaluates string equality.

- ❖ Given two strings, tests if they are equal.
- ❖ Returns a Boolean value:
  - Returns TRUE if strings are equal.
  - Returns FALSE if strings are not equal.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Evaluates whether the strings “Home” and “Home” are equal and returns TRUE (1):

```
<$strEquals("home","Home")$>
```

Evaluates whether the strings “home” and “Home” are equal and returns FALSE (0):

```
<$strEquals("home","Home")$>
```

strEqualsIgnoreCase

Description

Compares two strings without regard to case.

- ❖ Given two strings, tests if they are equal. Ignores the case between strings.
- ❖ Returns a Boolean value:
  - Returns TRUE if strings are equal.
  - Returns FALSE if strings are not equal.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Evaluates whether the strings “home” and “Home” are equal ignoring case and returns TRUE (1):

```
<$strEqualsIgnoreCase("home","Home")$>
```

Evaluates whether the strings “home” and “page” are equal and returns FALSE (0):

```
<$strEqualsIgnoreCase("home","page")$>
```

strIndexOf

Description

Checks for substring.

- Given two strings, determines if the second string is a substring of the first. The function returns an index value relating to the substring placement (see examples). The first character has the index value of 0. If the second string is not a substring the return value is -1.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

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

Description

Left fills a string.

- Given a string, a character, and an integer length, this function returns a string with the specified length left filled, if necessary, by the specified character.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

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*:

```
<$strLeftFill("sleep",'Z',5)$>
```

strLeftPad

Description

Pads extra space to the left of text to make it the specified string length.

- Takes a string and a length as parameters.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Pads extra space on the left to make it a string five characters long, using the form <space><space>inf:

```
<$strLeftPad("inf", 5)$>
```

strLength

Description

Evaluates the length of a string.

- Given a string, computes its length.
- Returns an integer value.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Evaluates the length of the string “home” and returns the integer 4:

```
<$strLength("home")$>
```

strLower

Description

Takes a string as a parameter and returns the lowercased string.

- Given a string, returns a lowercased string.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Evaluates the string “Home” and returns *home*.

```
<$strLower("Home")$>
```

strRightFill

Description

Returns a string with the specified length right filled.

- Given a string, a character, and an integer length, this function returns a string with the specified length right filled.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Formats the string *sleep* by right filling with the character "Z" to ten spaces. This returns the string *sleepZZZZZ*:

```
<$strRightFill("sleep",'Z',10)$>
```

Returns the string "sleep":

```
<$strRightFill("sleep",'Z',5)$>
```

strRightPad

Description

Pads extra space to the right of text to make it the specified string length.

- Takes a string and a length as parameters.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Pads extra space on the right to make it a string five characters long. using the the form, inf <space><space>:

```
<$strRightPad("inf", 5)$>
```

strRemoveWs

Description

Removes empty spaces in a string.

- Given a string, removes all empty spaces.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Formats the string “h o m e” as the string *home*:

```
<$strRemoveWs("h o m e")$>
```

strReplace

Description

Replaces a string.

- Given a string, substitutes a replacement string.
- This function takes three parameters. The first is the string on which the substitution will be performed. The second is the string to replace and the third is the replacement string. If there are multiple occurrences of the second string in the first string they will all be replaced by the third string (see second example).
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

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":

```
<$strReplace(formatDateOnly(dateCurrent()),"/",".")$>
```

strSubstring

Description

Retrieves characters from a string.

- Extracts a contiguous range of characters from the specified string. The function takes either two or three parameters. The first parameter is the string, the second parameter is the start index, and the third parameter is the stop index and is optional. If the stop index is not specified, the entire string from the start index is returned.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Retrieves the first two characters of the string *my*:

```
<$strSubstring("mystring",0,2)$>
```

Retrieves the string after the second character *string*:

```
<$strSubstring("mystring",2)$>
```

strTrimWs

Description

Removes empty spaces from beginning and end of a string.

- Given a string, trims empty space from the beginning and end.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Formats the string " homepage " as the string *homepage*:

```
<$strTrimWs(" homepage ")$>
```

strUpper

Description

Takes a string as a parameter and returns the uppercased string.

- Given a string, returns an uppercased string.
- Returns the formatted string.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Evaluates the string “home” and returns *HOME*.

```
<$strUpper("home")$>
```

toInteger

Description

Converts a string to an integer.

- Given a string, converts it to an integer.
- Returns an integer value.

Note: Use the Special String Operators for string concatenation, string inclusion, and simple comparison.

See Also: Special String Operators (2-21)

Example

Converts the string “4’ to an integer and returns the value 4:

```
<$toInteger("4")$>
```

trace

Description

Outputs a string to the debug trace.

- This function returns no value.
- This function takes a string parameter. This string is added to the accumulated debug trace. The *ScriptDebugTrace* flag must be set for this function to execute.

See Also: ScriptDebugTrace (4-74)

Example

Outputs a string to the accumulated debug trace:

```
<$trace("string")$>
```

url

Description

Formats a string as a URL.

- Given a string, converts it for use in a URL. Blank spaces and reserved characters are filled with an escape sequence.

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*:

```
<$url("home?page")$>
```

userHasRole

Description

Checks if the current user has the specified role.

- Returns the formatted string.The parameter is a string, which is the name of the role.
- Returns FALSE if the user does not have the role.

Example

Evaluates whether the user has the specified role:

```
<$userHasRole("admin")$>
```

utGetValue

Description

Returns the value of the property in the topic specified.

- This function has no output.
- Given a topic name and a property name, returns the value.

See Also: utLoad (3-52)

Example

Returns the value of *property1* in *topic1*:

```
<$utGetValue(“topic1”,"property1")$>
```

utLoad

Description

Loads topic.

- This function has no output.
- Given a topic name, loads the topic and makes the topic available for utLoadResultSet and utGetValue.

Example

Loads *topic1* and makes it available to other functions:

```
<$utLoad("topic1")$>
```

utLoadResultSet

Description

Loads the result set into the data binder.

- Given a topic name and a result set, loads the result set into the data binder.
- Returns TRUE (1) if the result set is successfully loaded into the DataBinder.

See Also: utLoad (3-52)

Example

Loads the result set *resultSet1* of *topic1* into the data binder:

```
<$utLoadResultSet("topic1","resultSet1")$>
```

xml

Description

Escapes non-alphanumeric characters.

- Returns a string. Takes a string parameter. This function escapes non-alphanumeric characters contained in a string and returns an xml formatted string.
- In the returned formatted string, the non-alphanumeric characters are replaced with the correct xml sequence that represents the character. For example, the ampersand "&" character is replaced with the xml "&" sequence representing the ampersand.
- When using a double-quote character within a string, the 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.

Example

Escapes the ampersand and returns the xml formatted string, "Me & you."

```
<$xml("Me & you.")$>
```

Escapes the non-alphanumeric characters and returns the xml formatted string, “Test the ", >, and < characters.”

```
<$xml("Test the \", >, and < characters.")$>
```

Chapter

4

IDOC SCRIPT PREDEFINED VARIABLES

OVERVIEW.

These Idoc Script variables are predefined in the Java class files. These variables reference code within the system server to execute a function, compute a true or false value, or return a value such as a string or an integer.

Idoc Script is made up of four basic types of variables:

- Variables used primarily as configuration file settings.
- Variables that are dynamic (including conditional dynamic).
- Variables that are setable.
- Variables that are evaluated once at the beginning of the service call (value variables).

Note: Those Idoc Script variables used primarily as configuration file settings are listed in *Chapter 7: Idoc Script Configuration Variables*.

Variable Types

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. Dynamic variables generally return a value such as a string or an integer.

AllowIntranetUsers	BrowserVersionNumber	CURRENT_DATE
CURRENT_ROW	DelimitedUserRoles	DocTypeSelected
DownloadSuggestedName	FIRSTREV	HasOriginal
HeavyClient	HttpAbsoluteCgiPath	HttpAdminCgiPath
HttpBrowserFullCgiPath	HttpCgiPath	HttpEnterpriseCgiPath
HttpWebRoot	IdcRegistrationPath	IsCheckinPreAuthed
IsCriteriaSubscription	IsCurrentNav	IsDynamic
IsExternalUser	IsFilePresent	IsFullTextIndexed
IsLocalSearchCollectionID	IsLocalSearchCollections	IsLoggedIn
IsMac	IsMultiPage	IsPromptingForLogin
IsRequestError	IsSubAdmin	IsSun
IsUploadSockets	IsUserEmailPresent	IsWindows
MSIE	NoMatches	OneMatch
UploadApplet	UseHtmlOrTextHighlightInfo	UserAccounts
UserAddress	UserAppRights	UserDefaultAccount
UserFullName	UserIsAdmin	UserName
UserRoles	UseXmlUrl	VDKSUMMARY

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 dynamic conditional variables will only provide a Boolean response and do not return a value such as a string or integer. In addition, dynamic conditional variables will not accept the *#active* keyword prefix. Thus, an error report is printed to the server debug output if the variable is not found.

Setable Variables

These 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. Generally, setting one of these variables changes the content of the page.

Value Variables

A value variable is evaluated once at the beginning of the service call and that value is used throughout the service call. The variable is then re-evaluated 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 through out the service call.

Predefined Variable Organization

This chapter organizes variables into general groups by functionality for easy reference. The chapter follows this outline:

- Content Item Related Variables
- Page Display Related Variables
- Search Related Variables
- Service Related Variables

CONTENT ITEM RELATED VARIABLES

DocTypeSelected

Description

Evaluates whether the content item types match. Used on a page to compare values when listing content item types.

Returns a Boolean value:

- Returns TRUE if the current content item type matches the content item type for the result set.
- Returns FALSE if the content item types do not match.

Example

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

```
<$DocTypeSelected$>
```

HasOriginal

Description

Checks whether an original content item exists for a revision. Generally used to query the user whether to overwrite when downloading.

Returns a Boolean value:

- Returns TRUE if an original content item exists and check in is generated.
- Returns FALSE if no check in is generated for this revision or there is no original content item name (`dOriginalName`).

Example

Checks for original content item:

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

HasUrl

Description

Checks whether a web layout file exists. Conditional dynamic variable (must be used within a conditional statement).

Returns a Boolean value:

- Returns TRUE if a web layout file exists.
- Returns FALSE if no web layout file exists.

Example

Checks for a web layout file:

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

IsCriteriaSubscription

Description

Evaluates whether subscription is criteria based rather than based on the content item name (`dDocName`).

Returns a Boolean value:

- Returns TRUE if subscription is criteria based.
- Returns FALSE if subscription is other than criteria based.

Example

Evaluates whether subscription is criteria based:

```
<$IsCriteriaSubscription$>
```

IsEditRev

Description

Check whether revision is in a workflow and enables editing.

Provides the ability to review and edit when part of a workflow. Links are provided to the reviewer to check out and edit the workflow content item.

This is only available on check in page and can be can be controlled from the Workflow Admin page. Value variable (evaluated once at the beginning of the service call).

- If set to true TRUE, provides the ability to review and edit.
- If set to FALSE, the ability to review and edit is disabled.

Example

Provides workflow details:

```
<$if IsEditRev$>
   addCheckinValue("IdcService", "WORKFLOW_CHECKIN");
if (form.isFinished.checked)
   addCheckinValue("isFinished", form.isFinished.value);
<$else$>
   addCheckinValue("IdcService", "CHECKIN_SEL");
<$endif$>
```

IsFailedConversion

Description

Checks whether the Inbound Refinery has failed the conversion process. Value variable (evaluated once at the beginning of the service call).

Returns a Boolean value:

- 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>
<$endif$>
```

IsFailedIndex

Description

Checks whether Indexer has failed to index the content item.

After the content item has successfully passed through the Inbound Refinery is it indexed by the search engine. Value variable (evaluated once at the beginning of the service call).

Returns a Boolean value:

- Returns TRUE if unable to index content item.
- Returns FALSE if content item is successfully indexed.

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

Description

Checks for the requested file and evaluates whether the requested file is the most current revision in the system before displaying.

Returns a Boolean value:

- Returns TRUE if the requested file is detected.
- Returns FALSE if the requested file can not be located.

Example

Displays a table and specifies formatting information if the requested file is present:

```
<$if IsFilePresent$>
   <td width=15% align=center><font
   color="<$strongHighlightFieldColor$>">
   <$dRevLabel$></font></td>
   <td width=35%><font
   color="<$strongHighlightFieldColor$>">
   <$dInDate$></font></td>
   <td width=25%><font
   color="<$strongHighlightFieldColor$>">
   <$dStatus$></font></td>
<$else$>
```

IsNotLatestRev

Description

Checks whether the revision is the last revision to be checked in. This is not necessarily the released revision. Value variable (evaluated once at the beginning of the service call).

Returns a Boolean value:

- Returns TRUE if the content item is other than the latest revision.
- Returns FALSE if the content item is the latest revision.

See Also: ClientControlled (4-50)

HasLocalCopy (4-56)

Example

Evaluates with latest revision:

```
<$IsNotLatestRev$>
```

IsNotSyncRev

Description

Checks whether local copy matches the most current revision by performing a content ID (`dID`) comparison.

Generally used to display an error message when the local copy of a content item has not been updated to the latest revision. Conditional dynamic variable (must be used within a conditional statement).

Returns a Boolean value:

- Returns TRUE if revisions do not match or newest revision could not be converted.
- 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 <$dDocName$>.

```
<$endif$>
```

SingleGroup

Description

Evaluates whether revision is in a contributor step. Used for presentation of workflow revisions.

Returns a Boolean value:

- Returns TRUE if revision is in a contributor step.
- Returns FALSE if the revision is not in a contributor step.

Example

Evaluates whether revision is in a contributor step:

```
<$if not SingleGroup$>
```

PAGE DISPLAY RELATED VARIABLES

AllowIntranetUsers

Description

Enables Windows NT/2000 security login (provides second method to login using Windows NT/2000 security).

Returns a Boolean value:

- Returns TRUE if NtmlSecurityEnabled is enabled (set to TRUE):
- Returns FALSE if NtmlSecurityEnabled not enabled.

See Also: NtmlSecurityEnabled (5-30)

Example

Evaluates Windows NT/2000 security:

```
<$AllowIntranetUsers$>
```

FIRSTREV

Description

Returns the first revision label and is used on the “Check in New” page.

- Returns the first revision label of the content item as string.
- Default is 1.

Example

Returns the first revision label (usually “1”):

```
<$FIRSTREV$>
```

HttpAdminCgiPath

Description

Defines the administrative CGI path (points to the CGI path to the Admin Server).

The Admin Server administrates the content servers and enables a user to stop/start/configure multiple content servers. Specifies the administrative CGI path as a relative URL.

- Returns defined administrative CGI path as string.

Example

Returns the administrative CGI path:

```
<$HttpAdminCgiPath$>
```

HttpBrowserFullCgiPath

Description

Defines a complete URL for the browser CGI path.

Specifies the CGI path as a complete URL. For example, specifies *http://www.mycomputer.com/stellent/intradoc-cgi/*iis_idc_cgi.dll. instead of /*intradoc-cgi.*

When set to TRUE, the complete URL for the CGI path is used instead of a relative root. When set to FALSE, enables a relative root to be used.

- Returns the CGI path as string.
- Default setting is FALSE.

Example

Returns full CGI path as string:

```
<$HttpBrowserFullCgiPath$>
```

HttpCgiPath

Description

Defines the CGI path.

Uses these flags to determine the value:

Flag	Description
UseSSL	When set to TRUE, the secure sockets layer is used.
isAbsoluteCgi	This is an internal flag set by the content server and is not intended for user configuration. This flag defines whether the complete URL for the CGI path is used instead of a relative root.
isAbsoluteWeb	This is an internal flag set by the content server and is not intended for user configuration. This flag defines whether the complete URL to the weblayout directory is used instead of a relative root.

❖ Returns the defined CGI path as string (defined in configuration files).

See Also: HttpWebRoot (4-26)

UseSSL (5-37)

Example

Returns the calculated CGI path:

```
<$HttpCgiPath$>
```

HttpCommonRoot

Description

Defines the root directory.

HttpCommonRoot is the prefix to use for accessing the shared common directory where web applets are located. Multiple content servers can share resources from one content server install. This attribute is the URL path to the <*home*>/common directory of the content server whose resources are being shared.

Value variable (evaluated once at the beginning of the service call). Specifies the URL of the common root directory. If the URL is external, the complete URL rather than the root directory defined in the configuration file is returned.

- Used as script and as a configuration file setting.
- Returns the defined root directory as string (defined in configuration files).

Example

As configuration setting, defines the root directory:

```
HttpCommonRoot=http://www.mycomputer.com/
```

As script, returns the defined root directory as string:

```
<$HttpCommonRoot$>
```

As script, the variable *HttpSharedRoot* can be used to designate the shared URL root path:

```
<$HttpCommonRoot$>=<$HttpSharedRoot$>common
```

HttpEnterpriseCgiPath

Description

Specifies the enterprise CGI path as a relative URL.

Points to the CGI path of the master content server. When multiple content servers share the same web login, one of them is nominated as the master or enterprise server.

- Returns a string.
- Returns defined enterprise CGI path (defined in configuration files).

See Also: HttpCommonRoot (4-21)

Example

Returns the enterprise CGI path as string.

```
<$HttpEnterpriseCgiPath$>
```

HttpHelpRoot

Description

Defines the URL to the help directory.

Value variable (evaluated once at the beginning of the service call). Specifies the path to the help directory as a relative URL. If the URL is external, the complete URL rather than the root directory defined in the configuration file is returned.

- Used as script and as a configuration file setting.
- Returns the defined help directory as string (defined in configuration files).

See Also: HttpCommonRoot (4-21)

Example

As configuration setting, defines help directory:

```
HttpHelpRoot=/stellent/weblayout/help/
```

As script, returns defined help directory as string:

```
<$HttpHelpRoot$>
```

HttpImagesRoot

Description

Defines the URL to the *images* directory

Value variable (evaluated once at the beginning of the service call). Specifies the path to the *images* directory and as a relative URL. If the URL is external, the complete URL rather than the root directory defined in the configuration file is returned.

- Used as script and as a configuration file setting.
- Returns the defined *images* directory as string (defined in configuration files).

Example

As configuration setting, defines *images* directory:

```
HttpImagesRoot=/stellent/weblayout/images/
```

As script, returns the defined *images* directory as string:

```
<$HttpImagesRoot$>
```

HttpSharedRoot

Description

Defines the shared URL path to the proxied server.

- Specifies the URL of the common root directory. This is the shared URL path for the proxied server.
- Returns the defined root directory as string.

See Also: HttpCommonRoot (4-21)

Example

As configuration setting, defines the shared directory:

```
HttpSharedRoot=http://www.mycomputer.com/common
```

As script, returns the defined directory as string:

```
<$HttpSharedRoot$>
```

As script, can be assigned to the variable *HttpSharedRoot* for sharing resources between content server installations:

```
<$HttpCommonRoot$>=<$HttpSharedRoot$>common
```

HttpWebRoot

Description

Defines the path to the *weblayout* directory.

Specifies the path to the *weblayout* directory as a relative URL. These flags in the configuration file change the content of this variable:

Flag	Description
UseSSL	When set to TRUE, the secure sockets layer is used.
isAbsoluteCgi	This is an internal flag set by the content server and is not intended for user configuration. This flag defines whether the complete URL for the CGI path is used instead of a relative root.
isAbsoluteWeb	This is an internal flag set by the content server and is not intended for user configuration. This flag defines whether the complete URL to the weblayout directory is used instead of a relative root.

❖ Returns the defined directory as string.

See Also: HttpCgiPath (4-20)

UseSSL (5-37)

Example

Returns the defined path as string:

```
<$HttpWebRoot$>
```

IsCurrentNav

Description

Used to build navigation on search results page.

Returns a Boolean value:

- Returns TRUE if this is the current page of a set of pages returned from a search request.
- Returns FALSE if not the currently displayed page.

Example

As part of a search results page:

```
<$loop NavigationPages$>
<$if IsCurrentNav$>
   <$HeaderPageNumber$>
<$else$>
   <a
   href="<$strRemoveWs(inc("searchapi_navigation_specific_p
   age"))$>">
   <$HeaderPageNumber$></a>
<$endif$>
<$endloop$>
```

IsMultiPage

Description

Checks whether multiple pages are needed for search results.

- This variable is dependent on the number of rows displayed per page.
- Default is 25.
- Set to TRUE if the results from a search are greater than the number of rows permitted per page as defined by the variable `isMaxRows`.

Example

Evaluates number of rows to display pages:

```
<$IsMultiPage$>
```

IsSavedQuery

Description

Determines whether a query has been saved to the portal for personal navigation. Setable variable (can be set within script or CGI string).

Returns a Boolean value:

- Returns TRUE if query has been saved in the database.
- Returns FALSE if query has not been saved or no query is found.

Example

Evaluates query status:

```
<$IsSavedQuery$>
```

PageParent

Description

Evaluates hierarchy of the directory page and is used in the creation of web layout pages. Value variable (evaluated once at the beginning of the service call).

Returns a Boolean value:

- Returns TRUE if the directory page is a child (subfolder) of another directory page.
- Returns FALSE if the directory page is not a sub folder.

Example

Checks if the directory page is a subfolder:

```
<$PageParent$>
```

ResultsTitle

Description

Provides a title for the search results page.

Value variable (evaluated once at the beginning of the service call). 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.

❖ Returns a string.

Example

As HDA entry, names the search results page:

```
@Properties LocalData
ResultsTitle=Content Items
@end
```

As script, returns the defined name:

```
<$if ResultsTitle$>ResultsTitle=<$url(ResultsTitle)$>
```

StdPageWidth

Description

Displays the standard page width.

- Provides a value indicating the page width in pixels. Generally this value is either 530 or 600 depending on the placement of the side bar.
- This value is returned as a string.

Example

Returns the page width as string:

```
<$StdPageWidth$>
```

TemplateName

Description

Provides the template name.

This is a template-related variable that 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. Value variable (evaluated once at the beginning of the service call).

Returns the internal name of the template (as string). For example, DOC_INFO or CHECKIN_NEW_FORM.

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>
```

TemplateClass

Description

Provides the classification of the template.

This is a template-related variable that make 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. Value variable (evaluated once at the beginning of the service call).

- Returns a string.
- For Templates, this is the value defined in the class column.
- For Reports, this evaluates to “Reports.”
- For SearchResultTemplates, this evaluates to *Results*.

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>
```

TemplateType

Description

Provides the template type.

This is a template-related variable that 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. Value variable (evaluated once at the beginning of the service call).

- Returns a string.
- For Templates, this is the value defined in the *formtype* column.
- For Reports, this is the value of the data source column.

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>
```

TemplateFilePath

Description

Provides the template file path.

This is a template-related variable providing the file location from where the template was actually loaded. Use this variable within a template page to determine the source of the pages delivered by the server.

This variable make it possible to create conditional content in a template based on the identity of the template. Value variable (evaluated once at the beginning of the service call). 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>
```

SEARCH RELATED VARIABLES

IsFullTextIndexed

Description

Checks whether Indexer has fully indexed the content item.

See Also: IsFailedIndex (page 4-11)

After the content item has successfully passed through the Inbound Refinery is it indexed by the search engine.

Returns a Boolean value:

- Returns TRUE if all text is indented on the page.
- Returns FALSE if full text is not indented.

Example

Provides a specified URL if the content item is fully indexed:

```
<$if IsFullTextIndexed$>
   <a href="<$redirect$>">
<$endif$>
```

IsLocalSearchCollectionID

Description

Checks whether the content item ID is in the local collection.

When configured with multiple instances, a search is performed to check whether a content item is in the local collection or in an external collection.

Searches on the assigned content item ID.

Returns a Boolean value:

- Returns TRUE if the content item is in 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-->
<$exec IsLocalSearchCollection=""$>
```

NoMatches

Description

Checks whether no matches were found.

Generally used to display a message on the search results page to the user. Check whether no matches were found from a search query.

Returns a Boolean value:

- Returns TRUE if no matches for search query 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>
<$endif$>
```

OneMatch

Description

Checks whether one match was found from a search query.

Generally used to display a message on the search results page to the user.

Returns a Boolean value:

- Returns TRUE if no matches for search query were found.
- Returns FALSE if any matches 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>
<$endif$>
```

UseHtmlOrTextHighlightInfo

Description

Checks whether search keyword highlighting is enabled and format supports highlighting, such as PDF, HTM, or TXT.

This value is enabled in the System Properties utility as: *Enable search keyword highlighting.*

Returns a Boolean value:

- Returns TRUE if highlighting is enabled.
- Returns FALSE if option is not enabled.

See Also: UseXmlUrl (4-42)

EnableDocumentHighlight (5-25)

Example

Return search keyword highlighting status:

```
<$UseHtmlOrTextHighlightInfo$>
```

UseXmlUrl

Description

Checks whether search keyword highlighting for XML documentation is enabled. Uses an XML URL when performing a keyword search.

Returns a Boolean value:

- Returns TRUE if UseXmlUrl is enabled.
- Returns FALSE if the option is not enabled.

See Also: UseHtmlOrTextHighlightInfo (4-41)

EnableDocumentHighlight (5-25)

Example

Returns search keyword highlighting for XML documentation status.

```
<$UseXmlUrl$>
```

VDKSUMMARY

Description

Used with Verity full text search to generate a summary of a content item.

- Verity specific variable.
- Programmatically generates a summary of the content item using sentences selected with keywords.
- Returns a string.

Example

Displays a generated content item summary:

```
<$VDKSUMMARY$>
```

SERVICE RELATED VARIABLES

AdminAtLeastOneGroup

Description

Checks administrator role of the user.

Conditional dynamic variable (must be used within a conditional statement). Relates to the currently logged on user.

Returns a Boolean value:

- Returns TRUE if the user is an administrator for at least one security group.
- Returns FALSE if the user does not hold an administrator role.

See Also: UserIsAdmin (4-81)

Example

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

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

AfterLogin

Description

Checks whether the *Contents* page is created after a login.

Conditional dynamic variable (must be used within a conditional statement).

Returns a Boolean value:

- Returns TRUE if the *Contents* page (TOC) is created immediately after a login.
- Returns FALSE if page fails to display.

Example

Displays an alternate URL if the contents page was not created:

```
<$if not AfterLogin$>
   <a href="<$redirect$>">
<$endif$>
```

AllowCheckin

Description

Checks whether the user has check in rights.

Conditional dynamic variable (must be used within a conditional statement). Relates to the currently logged on user.

Returns a Boolean value:

- Returns TRUE if user has check in rights to the specified security group.
- Returns FALSE if user does not have check in rights.

Example

Can be used to do an optional presentation for a user with check in rights:

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

AllowCheckout

Description

Checks whether user has check out rights.

Conditional dynamic variable (must be used within a conditional statement). Relates to the currently logged on user.

Returns a Boolean value:

- Returns TRUE if user has check out rights for the specified security group.
- Returns FALSE if user does not have check out rights.

Example

Can be used to do an optional presentation for a user with check out rights:

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

AuthorAddress

Description

Checks for email address of content item author.

Value variable (evaluated once at the beginning of the service call) returning a string or Boolean value depending on use.

- Returns TRUE if the content item author has a defined email address.
- Returns FALSE if the content item author has no email address.

Example

Can be used to alert the content item author via email when a revision is made.

```
<$AuthorAddress$>
```

BrowserVersionNumber

Description

Checks the browser version number.

- Returns a string.
- Returns the version number of the browser.

Example

Can be used to ensure that the user has a browser version compatible with Stellent.

```
<$BrowserVersionNumber$>
```

ClientControlled

Description

Checks whether the ODMA Client string for controlling the update process is provided by the client.

Setable variable (can be set within script or CGI string).

Returns a Boolean value:

- Returns TRUE if client controlled.
- Returns FALSE if not client controlled or string is not defined.

See Also: HasLocalCopy (4-56)

IsNotLatestRev (4-13)

Example

Can be used to define circumstance for controlling the update process:

```
<$ClientControlled$>
```

DelimitedUserRoles

Description

Returns the delimited user role list.

A comma separated, colon delimited, list of roles the user belongs to. This variable is read only and cannot be assigned a value. Relates to the user currently logged in.

❖ Returns a string.

See Also: UserRoles (4-83)

Example

This markup displays user variables on a page:

```
<$if DelimitedUserRoles$>
   <$include optional_field$>
<$endif$>
```

DownloadSuggestedName

Description

Provides an automatically assigned name when downloading a file.

- Returns a string. This value is defined in the System Properties utility and is included in the configuration file when the system is initialized.
- Returns the path and suggested name for the downloaded file.

Example

Returns the path and suggested name for the downloaded file:

```
<$DownloadSuggestedName$>
```

EmptyAccountCheckinAllowed

Description

Determines whether the user needs to specify an account entry on the check in page.

Conditional dynamic variable (must be used within a conditional statement). Used on the Standard Page Resources page to display a message.

Returns a Boolean value:

- Returns TRUE if an account entry is required.
- Returns FALSE if an account entry is not required.

Example

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

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

ExternalUserAccounts

Description

Enables external users to hold accounts.

Lists all external user accounts.

See Also: ExternalUserRoles (4-55)

Example

Returns a list of external user accounts:

```
<$ExternalUserAccounts$>
```

ExternalUserRoles

Description

Provides a list of roles the external user belongs to.

Returns a string. Relates to the user currently logged in. External user roles are a comma-separated list. This variable is read only and cannot be assigned a value.

See Also: UserRoles (4-83)

Example

Returns a list of external user roles:

```
<$ExternalUserRoles$>
```

HasLocalCopy

Description

Checks whether the user has a local copy of the requested content item.

Setable variable (can be set within script or CGI string). Used with ClientControlled.

Returns a Boolean value:

- Returns TRUE if a local copy is detected in the download target directory.
- Returns FALSE if a local copy is not detected.

See Also: ClientControlled (4-50)

IsNotLatestRev (4-13)

Example

Checks for a local copy of the content item:

```
<$HasLocalCopy$>
```

HasPredefinedAccounts

Description

Checks whether current user has predefined accounts.

Conditional dynamic variable (must be used within a conditional statement). Accounts can be predefined on the User Administration utility.

Returns a Boolean value:

- Returns TRUE if current user has predefined accounts.
- Returns FALSE if there are no predefined accounts.

Example

Checks for predefined account:

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

HeavyClient

Description

Checks whether checking in with ODMA or Upload applet.

Conditional variable.

Returns a Boolean value:

- Returns TRUE if check in is performed with ODMA or Upload applet.
- Returns FALSE if ODMA or Upload applet is not used in the check in process.

Example

Checks for check in method:

```
<$HeavyClient$>
```

IsCheckinPreAuthed

Description

Checks whether the current user is pre-authorized to check in a content item.

Conditional variable.

Returns a Boolean value:

- Returns FALSE if user is not authorized to check in a content item.
- Returns TRUE if use is pre-authorized for content item check in.

Example

Checks for check in authorization of the user:

```
<$IsCheckinPreAuthed$>
```

IsDynamic

Description

Checks whether the page is presented dynamically to the user.

Most pages viewed by the user are dynamic. However, some pages are designed for the anonymous parts of a web site. These pages are typically built into a static location and delivered to the user without changes. Examples are the guest portal page and the content of some auto generated e-mails.

This variable is dependent on whether the page is presented dynamically or as static web page.

Returns a Boolean value:

- Returns TRUE if the page is being presented dynamically to the user.
- Returns FALSE if the page is static or can not be displayed.

Example

Evaluates whether the page is presented dynamically:

```
<$if IsDynamic$>
   <a href="<$redirect$>">
<$endif$>
```

IsExternalUser

Description

Checks whether the user is accessing the site from an external location

Conditional variable.

Returns a Boolean value:

- Returns TRUE if the user is accessing the site from an external location.
- Returns FALSE if the user is on the local system.

Example

Checks if user is at an external location:

```
<$IsExternalUser$>
```

IsJava

Description

Displays the local data of a page.

Setable variable (can be set within script or CGI string).

- When set to TRUE, displays the local data of a page.
- Returns content of the binder.

Example

Displays local data of a page:

```
<$IsExternalUser=1$>
```

IsLoggedIn

Description

Checks whether a user is logged in.

Conditional variable.

Returns a Boolean value:

- Returns TRUE if the user has logged into the system.
- Returns FALSE if the user has not logged in.

Example

Checks whether the user is logged in and has an email address before performing a function.

```
<@dynamichtml subscription_action_script@>
function allowSubscription(form)
{<$if IsLoggedIn$>
<$if IsUserEmailPresent$>
<$else$>
…}
<@end@>
```

IsMac

Description

Checks operating system of client.

Checks operating system of client and is used to redirect for a Macintosh. Conditional variable.

Returns a Boolean value:

- Returns FALSE if any other operating system.
- Returns TRUE if client browser is running on a Macintosh operating system.

Example

Displays a specified URL to a browser running on a Macintosh.

```
<$if IsMac $>
   <a href="<$redirect$>">
<$endif$>
```

IsNew

Description

Checks whether the content item is new or a revision.

Value variable (evaluated once at the beginning of the service call).

Returns a Boolean value:

- Returns TRUE if the content item is new when performing a check in.
- Returns FALSE if the content item is a revision.

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 account of the user currently logged in is used:

```
<$if isNew$>
   <$defaultAccount$>
<$endif$>
```

IsPromptingForLogin

Description

Determines method of login prompt.

Conditional variable. Evaluates whether the server is set to prompt for login or if login is being handled programmatically.

Returns a Boolean value:

- Returns TRUE if 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:

```
<$IsPromptingForLogin$>
```

IsRequestError

Description

Determines whether there is a request error condition present on the server.

Conditional variable. Determines whether there is a request error condition present on the server by evaluating the status code. The value *StatusCode* is set as a side effect to certain Idoc Script function calls such as *executeService*. The status code returns a negative numeric value if there is a request error condition present on the server.

If *IsRequestError* is TRUE, a request error condition is present on the server and the typical behavior is to abort the display of the current page and substitute an error page.

Returns a Boolean value:

- Returns TRUE if there is a request error condition present on the server.
- Returns FALSE if the value of *StatusCode* is other than a negative numeric value.

See Also: abortToErrorPage (3-2)

executeService (3-9)

Example

Evaluates request error status:

```
<$IsRequestError$>
```

IsSubAdmin

Description

Checks whether a user has sub admin rights.

Conditional variable.

Returns a Boolean value:

- Returns TRUE if the user has sub admin rights to at least one administrative application.
- Returns FALSE if the user does not have sub admin rights.

Example

Checks whether the user is logged in and has sub admin rights before performing a function.

```
<@dynamichtml subscription_action_script@>
function allowSubscription(form)
{<$if IsLoggedIn$>
<$if IsSubAdmin$>
<$else$>...}
<@end@>
```

IsUploadSockets

Description

Used by the Multi Upload applet to determine whether the upload socket should be used.

Conditional variable. Used on install. Not intended for user configuration.

Returns a Boolean value:

- Returns TRUE if the upload socket is defined for use with the Multi Upload applet.
- Returns FALSE if the upload socket should not be used.

Example

N/A

IsUserEmailPresent

Description

Checks whether the email address is defined for the user.

Conditional variable.

Returns a Boolean value:

- Returns TRUE if the specified user has a defined email address.
- Returns FALSE if email is not defined.

Example

Checks whether the user is logged in and has an email address before performing a function.

```
<@dynamichtml subscription_action_script@>
function allowSubscription(form)
{<$if IsLoggedIn$>
<$if IsUserEmailPresent$>
<$else$>...}
<@end@>
```

IsWindows

Description

Checks operating system of client.

Conditional variable.

Returns a Boolean value:

- Returns TRUE if client browser is running on a Windows operating system.
- Returns FALSE if any other operating system.

Example

Displays a specified URL to a browser running on a Macintosh:

```
<$if IsWindows$>
   <a href="<$redirect$>">
<$endif$>
```

MSIE

Description

Determines whether client is running Internet Explorer.

Returns a Boolean value:

- Returns TRUE if the client is running Microsoft Internet Explorer browser.
- Returns FALSE if the Internet Explorer browser is not detected.

Example

Determines if browser is Internet Explorer:

```
<$MSIE$>
```

NumAdditionalRenditions

Description

This variable evaluates the number of renditions of a content item.

- Affects the search results page.
- There is no default value.
- Set to `NumAdditionalRenditions=1`

Example

Used to determine if `useThumbnails` should be set:

```
<@dynamichtml searchapi_result_definitions@>
<$if NumAdditionalRenditions and NumAdditionalRenditions > 0$>
   <$useThumbnails=1$>
<$endif$>
<@end@>
```

ScriptDebugTrace

Description

Enables a trace of all *resource includes* and calls to the Idoc Script function *eval.*

The contents of the *eval* function and any dynamically assigned *resource 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.*

Setable variable (can be set within script or CGI string). Can be used as a setting in the configuration files or as a parameter to a service call.

- If set to TRUE, performs a trace of all *resource includes*, calls to the Idoc Script function *eval*, and displays the results of the trace on the page.
- If set to FALSE, the trace and function call are not executed..

See Also: ScriptErrorTrace (4-75)

setResourceInclude (3-34)

trace (3-50)

Example

As parameter to a service call, all debug trace information is added to the bottom of the displayed page:

```
http://<home>/intradoc-
cgi/idc_cgi_isapi.dll?IdcService=GET_DOC_PAGE&Action=
GetTemplatePage&Page=HOME_PAGE&ScriptDebugTrace=true
```

As script, returns the value of the configuration setting:

```
<$ScriptDebugTrace$>
```

ScriptErrorTrace

Description

Enables a trace of script errors.

When used as a parameter to a service call, script error information can be added to the bottom of the displayed page. If no errors are detected the message "No Errors" is displayed.

:Setable variable (can be set within script or CGI string). Can be used as a setting in the configuration files or as a parameter to a service call.

- If set to TRUE, performs a trace of script errors and displays the results.
- If set to FALSE, no trace of script errors is performed.

See Also: ScriptDebugTrace (4-74)

setResourceInclude (3-34)

trace (3-50)

Example

As parameter to a service call:

```
http://<home>/stellent-
cgi/idc_cgi_isapi.dll?IdcService=GET_DOC_PAGE&Action=
GetTemplatePage&Page=HOME_PAGE&ScriptErrorTrace=true
```

As script, returns the value of the configuration setting:

```
<$ScriptDebugTrace$>
```

UserAccounts

Description

List of accounts accessible to the user.

Evaluation: A comma separated list of accounts the user has access to. User accounts are a comma-separated list. Relates to the user currently logged in.

There are special variables to gather information about the user currently logged in or the current template. These variables are read only and cannot be assigned a value.

- Returns a list of accounts accessible to the user (as string).

Example

This markup displays user variables on a page:

```
<$if UserName$>
   Logon Name: <$UserName$><BR>
   User Name: <$UserFullName$><BR>
   email Address: <$UserAddress$><BR>
   Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserAddress

Description

The e-mail address of the user currently logged in.

- Relates to the user currently logged in.
- There are special variables to gather information about the user currently logged in or the current template. These variables are read only and cannot be assigned a value.
- Returns the e-mail address of the user currently logged in (as string).

Example

This markup displays user variables on a page:

```
<$if UserName$>
   Logon Name: <$UserName$><BR>
   User Name: <$UserFullName$><BR>
   email Address: <$UserAddress$><BR>
   Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserAppRights

Description

Checks the defined application rights of the current user.

Relates to the user currently logged in. There are special variables to gather information about the user currently logged in or the current template. This variable is read only and cannot be assigned a value.

- Return a string.
- Returns a number specifying the user rights.

Example

Displays application rights of the current user:

```
<$UserAppRights$>
```

UserDefaultAccount

Description

The default account of the user currently logged in.

- Relates to the user currently logged in.
- There are special variables to gather information about the user currently logged in or the current template. This variable is read only and cannot be assigned a value.
- Returns a string.

Example

This markup displays user variables on a page:

```
<$if UserName$>
   Logon Name: <$UserName$><BR>
   User Name: <$UserFullName$><BR>
   email Address: <$UserAddress$><BR>
   Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserFullName

Description

The full name of the user currently logged in.

- Relates to the user currently logged in.
- If a user is not currently logged in, evaluates to the string “anonymous.” There are special variables to gather information about the user currently logged in or the current template. These variables are read only and cannot be assigned a value.
- Returns the full username as string.

Example

This markup displays user variables on a page:

```
<$if UserName$>
   Logon Name: <$UserName$><BR>
   User Name: <$UserFullName$><BR>
   email Address: <$UserAddress$><BR>
   Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserIsAdmin

Description

Checks whether the current user has administrative rights.

Conditional variable:

Returns a Boolean value.

- Returns TRUE if the current user has administrative rights.
- Returns FALSE if the user does not hold admin rights.

Example

Evaluates administrative rights of user:

```
<$UserIsAdmin$>
```

UserName

Description

The user currently logged in.

- Relates to the user currently logged in.
- This is the name of the user currently logged in. If a user is not currently logged in, evaluates to the string “anonymous.” There are special variables to gather information about the user currently logged in or the current template. These variables are read only and cannot be assigned a value.
- Returns the username as string.

Example

This markup displays user variables on a page:

```
<$if UserName$> Logon Name: <$UserName$><BR>
   User Name: <$UserFullName$><BR>
   email Address: <$UserAddress$><BR>
   Default Account: <$UserDefaultAccount$><BR>
<$endif$>
```

UserRoles

Description

Provides a list of roles the user belongs to.

- Relates to the user currently logged in.
- User roles are a comma-separated list. This variable is read only and cannot be assigned a value.
- Returns a string.

Example

References the list of user roles:

```
<$if UserRoles$>
   <$include optional_field$>
<$endif$>
```

Chapter

5

IDOC SCRIPT CONFIGURATION VARIABLES

OVERVIEW

Configuration variables are Idoc Script predefined variables generally used as a setting in the configuration files. In some cases, these variables may 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 only be set to ‘on’ or ‘off’ and should be set with the value 1 or 0 (TRUE or FALSE).
- Unless otherwise specified, Boolean type configuration values will default to 0 (FALSE) and String type configuration values will default to an empty string.

Several Idoc Script configuration variables are commonly or frequently used and are of particular interest to the system administrator. These variables are listed in the *Commonly Used Configuration Variables* section of this chapter:

AuthorDelete	EnableDocumentHighlight
EnterpriseSearchAsDefault	ExclusiveCheckout
GetCopyAccess	HasExternalUsers
NtlmSecurityEnabled	SelfRegisteredAccounts
SelfRegisteredRoles	ShowOnlyKnownAccounts
SysAdminAddress	UseAccounts
UseSelfRegistration	UseSSL

Configuration Variable Organization

This chapter organizes configuration variables into general groups by functionality for easy reference. The chapter follows this outline:

- Commonly Used Configuration Variables
- Inbound Refinery Configuration Variables
- Batch Loader Configuration Variables
- Web Filter Configuration Variables
- Miscellaneous Configuration Variables

Configuration Variable Cross Reference

This configuration variable cross reference organizes variables by the configuration file (.cfg) in which they are most commonly located. Your actual application will vary depending on the criteria and selections made during installation.

Content Server Related CFG Files

These configuration files (.cfg) are located within the content server directory.

<contentserver_dir>/bin/intradoc.cfg

Configuration Variable	Brief Description
BatchLoaderPath	Sets the BatchLoader file path. See 5-87 for additional information.
BatchLoaderUserName	Sets the BatchLoader user name. See 5-88 for additional information.
CleanUp	Used for BatchLoader file clean up. See 5-89 for additional information.
EnableErrorFile	Generates a BatchLoader error file. See 5-58 for additional information.
HTMLEditorPath	Defines the selected HTML editor executable path for the Component Wizard. See 5-127 for additional information.
IntradocDir	Defines the path to the root primary directory. See 5-138 for additional information.

Configuration Variable	Brief Description
IsPhysicallySplitDir	Used to flag the server that the Vault and Weblayout directories are on different file systems. See 5-152 for additional information.
MaxErrorsAllowed	Sets the number of errors after which the BatchLoader stops processing records from the batch load file. See 5-90 for additional information.
PreviewOutputExtension	Sets the HTML Preview extension. See 5-73 for additional information.
PreviewPath	References an HTML Preview executable file. See 5-74 for additional information.
VaultDir	Defines the path to the vault directory. See 5-199 for additional information.
WebBrowserPath	Defines the path to the web browser that displays the Stellent Help files in stand-alone applications. See 5-203 for additional information.
WeblayoutDir	Defines the path to the directory named *weblayout*. This is the root directory of the content server web site. See 5-204 for additional information.

<contentserver_dir>/config/config.cfg

Configuration Variable	Brief Description
AccountMapPrefix	Defines the prefix used to identify which Windows NT/2000 groups map to accounts. See 5-104 for additional information.
AdditionalIndexBuildParams	Provides additional parameters to add to the Index command line. See 5-39 for additional information.
AllowAlternateMetaFile	Allows users to submit 'metadata-only' content for the alternate file. See 5-105 for additional information.
AllowPrimaryMetaFile	Allows users to submit 'metadata-only' content for the primary file. See 5-106 for additional information.
AuthorDelete	Enables author delete privileges. See 5-24 for additional information.
AutoNumberPrefix	Defines the automatic numbering prefix. See 5-107 for additional information.
CGI_DEBUG	Used to debug the web server filter. See 5-91 for additional information.
CGI_RECEIVE_DUMP	Defines whether the Common Gateway Interface is enabled to receive an information dump. See 5-92 for additional information.

Configuration Variable	Brief Description
CGI_SEND_DUMP	Defines whether the Common Gateway Interface is enabled to send an information dump. See 5-93 for additional information.
CgiFileName	Defines the Common Gateway Interface file name. See 5-108 for additional information.
ColumnMapFile	Required for Oracle only. References the column mapping HTM file. See 5-110 for additional information.
CreatePrimaryMetaFile	Enables author delete privileges. See 5-111 for additional information.
DatabasePreserveCase	Defines whether the character case from the database is preserved. See 5-112 for additional information.
DatedCacheIntervalDays	Defines the Dynamic Converter cache removal in days. See 5-113 for additional information.
DefaultAccounts	Sets default account information. See 5-118 for additional information.
DefaultAuthType	Sets the default authentication challenge type. See 5-94 for additional information.
DefaultHtmlConversion	References a default template for Dynamic Converter. See 5-119 for additional information.

Configuration Variable	Brief Description
DefaultMasterDomain	Defines the default master domain. If not set, this value is the domain of the Windows NT/2000 server machine that is hosting the web server. See 5-95 for additional information.
DefaultNetworkAccounts	Defines the default network account. See 5-95 for additional information.
DefaultPasswordEncoding	Defines the default password encoding type. See 5-120 for additional information.
DirectoryLockingLogPath	Defines the log file used during a temporary locking of directories. See 5-121 for additional information.
DoDocNameOrder	Presentation option for the Repository Manager application. See 5-122 for additional information.
DownloadApplet	Evaluates whether the Download applet is enabled. Used as Idoc Script and configuration variable. See 5-123 for additional information.
EnableDocumentHighlight	Enables content item highlighting. See 5-25 for additional information.
EnterpriseSearchAsDefault	Defines whether Enterprise Search is the default setting. See 5-26 for additional information.

Configuration Variable	Brief Description
ExclusiveCheckout	Defines check out rights. See 5-27 for additional information.
FILTER_DEBUG	Defines whether the web server plug-in filter enables debug output. See 5-97 for additional information.
ForceDistinctRevLabel	Defines revision label usage. See 5-124 for additional information.
ForceDocTypeChoice	Used on a displayed page to define a Type choice list with a blank entry. See 5-125 for additional information.
ForceSecurityGroupChoice	Used on a displayed page to define a Security Group choice list with a blank entry. See 5-126 for additional information.
GetCopyAccess	Permits user to get a copy of a content item when only read access is defined. See 5-28 for additional information.
HasExternalUsers	Notifies the system of an external user database. See 5-29 for additional information.
HttpRelativeWebRoot	Defines the web root directory. See 5-128 for additional information.
HttpServerAddress	Defines the server address. See 5-129 for additional information.
IDC_Name	Defines the unique instance name. See 5-131 for additional information.

Configuration Variable	Brief Description
IndexerLargeFileSize	Defines the maximum file size for the Indexer. See 5-134 for additional information.
InstanceDescription	Defines the server name for that server instance. See 5-136 for additional information.
InstanceMenuLabel	Provides the defined description of the server instance menu label. See 5-137 for additional information.
IntradocRealm	Defines the server realm. See 5-98 for additional information.
IntradocServerHostName	Defines the host name to use when opening a socket connection to the content server. See 5-139 for additional information.
IntradocServerPort	Defines the port that the ISAPI filter or any other application should use to talk to the content server. See 5-140 for additional information.
IsAutoArchiver	Enables or disables the automatic import or transfer of content items. See 5-141 for additional information.
IsAutoNumber	Enables auto numbering. See 5-142 for additional information.
IsAutoQueue	Enables or disables the processing of content item post-conversion. See 5-143 for additional information.

Configuration Variable	Brief Description
IsAutoSearch	Enables or disables auto export and indexing. See 5-144 for additional information.
IsFormsPresent	Enables forms features. See 5-146 for additional information.
IsJdbcLockTrace	Enables Java Database Connectivity lock trace to view the database locking calls. See 5-148 for additional information.
IsJdbcQueryTrace	Enables Java Database Connectivity query trace to view queries being executed against the database. See 5-149 for additional information.
IsJspServerEnabled	Enables Java Server Page functionality. See 5-150 for additional information.
IsOverrideFormat	Checks whether the system properties are set to allow override format on check in. Used as Idoc Script and as a configuration variable. See 5-151 for additional information.
JdbcConnectionString	Defines the Java Database Connectivity connection including the hostname, port number, and instance name. See 5-153 for additional information.

Configuration Variable	Brief Description
JdbcPassword	Defines the SQL Server database password. See 5-155 for additional information.
JdbcPasswordEncoding	Defines the Java Database Connectivity password encoding type. See 5-156 for additional information.
JdbcUser	Defines the Java Database Connectivity user name. See 5-157 for additional information.
JspAdminQuery	Defines the files to be made available as web application files. See 5-158 for additional information.
JspDefaultIndexPage	Defines the default pages for a web application. See 5-159 for additional information.
JspEnabledGroups	Defines the security groups to be enabled for Java Server Page functionality. See 5-160 for additional information.
LocalGroupServer	Specifies the Windows NT/2000 server to interrogate for the local groups that contain the user as a member. See 5-99 for additional information.
MacSupportsSignedApplets	Relates to the acceptance of signed applets for Macintosh systems. See 5-161 for additional information.

Configuration Variable	Brief Description
MailServer	Defines the e-mail server. See 5-162 for additional information.
MajorRevSeq	Defines the Major Revision Label Sequence. See 5-163 for additional information.
MaxArchiveErrorsAllowed	Defines the number of errors that the Archiver will allow. See 5-165 for additional information.
MaxCollectionSize	Defines the total number of files passed to Indexer in one batch. See 5-166 for additional information.
MaxDocIndexErrors	Defines the number of Indexer errors allowed. See 5-167 for additional information.
MaxQueryRows	Sets the maximum number of search results rows. See 5-168 for additional information.
MaxResults	Defines the number of returned content items on a search result. See 5-169 for additional information.
MaxSearchConnections	Defines the maximum number of search connections in the system. See 5-170 for additional information.
MinorRevSeq	Defines the Minor Revision Label Sequence See 5-173 for additional information.

Configuration Variable	Brief Description
MultiUpload	Permits the upload up multiple files. See 5-175 for additional information.
NetworkAdminGroup	References the Domain Admins NT group. See 5-176 for additional information.
NoAutomation	Enables or disables automation activity. See 5-177 for additional information.
NtlmSecurityEnabled	Enables the Windows NT/2000 LAN Manager (NTLM) authentication process. See 5-30 for additional information.
SearchCacheTrace	Enables verbose output to the View Server Output window. See 5-178 for additional information.
SearchConnectionWaitTimeout	Defines the search connection wait timeout. See 5-179 for additional information.
SearchDebugLevel	Defines the search debug level. See 5-180 for additional information.
SearchDir	Defines the path to the Search directory. See 5-182 for additional information.
ShowOnlyKnownAccounts	Defines whether to display only known accounts on Check In page. See 5-33 for additional information.

Configuration Variable	Brief Description
SmtpPort	Defines the Simple Mail Transfer Protocol (SMTP) port number. See 5-184 for additional information.
SpecialAuthGroups	Defines named special authorization groups. See 5-101 for additional information.
SysAdminAddress	Defines the system administrator e-mail address. See 5-34 for additional information.
SystemDateFormat	Overrides the default date/time format used by the content server. See 5-186 for additional information.
SystemLocale	Sets the default system locale for the content server. See 5-188 for additional information.
SystemTimeZone	Overrides the default time zone. See 5-190 for additional information.
ThumbnailHeight	Sets the Thumbnail height in pixels. See 5-80 for additional information.
ThumbnailWidth	Sets the Thumbnail Width in pixels. See 5-81 for additional information.
TimeoutPerOneMegInSec	Sets the Inbound Refinery time out per one megabyte. See 5-82 for additional information.

Configuration Variable	Brief Description
UploadApplet	Evaluates whether the Upload applet is enabled. Used as Idoc Script and configuration variable. See 5-194 for additional information.
UseAccounts	Defines whether to enable the use of accounts. See 5-35 for additional information.
UseBellevueLook	Enables alternate set of navigation buttons on interface. See 5-195 for additional information.
UseFourDigitYear	Obsolete—use SystemDateFormat. Used in previous versions to define a four-digit or two-digit year format. See 5-196 for additional information.
UseStellentLook	Enables the Stellent set of navigation interface buttons. This is the default interface. See 5-197 for additional information.
UseLocalGroups	Defines whether the Windows NT/2000 server checks to see if the user belongs to any of the local groups on that server. See 5-102 for additional information.
UseSelfRegistration	Enables self-registration for users. See 5-36 for additional information.
UseXpedioLook	Enables alternate set of navigation buttons on interface. See 5-198 for additional information.

Configuration Variable	Brief Description
VerityInstallDir	Defines the Verity installation directory. See 5-201 for additional information.
VerityLocale	Defines the Verity local language selection. See 5-202 for additional information.
WebProxyAdminServer	Defines whether a web proxy administrative server is used. See 5-205 for additional information.
WebServerAuthOnly	Used to enable web server authorization only: See 5-103 for additional information.

<contentserver_dir>/search/search.cfg

Configuration Variable	Brief Description
VerityAppName	Verity License Key
VerityAppSignature	Verity License Key
VerityInstallDir	Defines the Verity installation directory. See 5-201 for additional information.

<contentserver_dir>/admin/bin/intradoc.cfg

Configuration Variable	Brief Description
IntradocDir	Defines the path to the root primary directory. See 5-138 for additional information.
WeblayoutDir	Defines the path to the directory named *weblayout*. This is the root directory of the content server web site. See 5-204 for additional information.

<contentserver_dir>/admin/config/config.cfg

Configuration Variable	Brief Description
HttpRelativeWebRoot	Defines the web root directory. See 5-128 for additional information.
CgiFileName	Defines the Common Gateway Interface file name. See 5-108 for additional information.
HttpServerAddress	Defines the server address. See 5-129 for additional information.
IDC_Admin_Name	Not currently used. Defines a separate IDC administrator name. See 5-130 for additional information.

Inbound Refinery Related CFG Files

These configuration files (.cfg) are located within the Inbound Refinery directory.

<install_dir>/IdcRefinery/connections/main/intradoc.cfg

Configuration Variable	Brief Description
ConnectionName	References the name assigned to the connection. See 5-46 for additional information.
DocConverterEngineDir	References the location of the Inbound Refinery executables. See 5-57 for additional information.
IntradocDir	Defines the path to the root primary directory. See 5-138 for additional information.
JvmCommandLine	Defines the location of the Java Virtual Machine command line. See 5-62 for additional information.
ProcessHyperlinks	Enables the processing of hyperlinks in Word and PowerPoint content items. See 5-77 for additional information.
SharedDir	Defines the path to the shared directory. See 5-183 for additional information.
ShowHyperlinkBox	Defines whether a box is placed around hyperlinks in PDF files. See 5-78 for additional information.

Configuration Variable	Brief Description
TerminateAcrobat	Defines the condition for terminating Adobe Acrobat. See 5-79 for additional information.
UseAdobePDFMaker	Defines whether to use PDFMaker for the conversion of Word files. See 5-83 for additional information.
VaultDir	Defines the path to the vault directory. See 5-199 for additional information.
VerboseMode	Specifies whether to use Verbose logging. See 5-86 for additional information.
WeblayoutDir	Defines the path to the directory named *weblayout*. This is the root directory of the content server web site. See 5-204 for additional information.

<install_dir>/IdcRefinery/shared/idcrefinery.cfg

Configuration Variable	Brief Description
AcadUseLISPInterface	Sets the AutoCad LISP interface. See 5-38 for additional information.
AdjustPrinterMargins	Adjusts the defined printer margins. See 5-41 for additional information.
AllowPassthru	Sets the handling of files that time out. See 5-42 for additional information.

Configuration Variable	Brief Description
AutoCad2000PlotterFilePath	Defines the AutoCad2000 plotter file path. See 5-43 for additional information.
ComputerName	References the name assigned to the computer. See 5-45 for additional information.
CreatePDFThumbnails	Configuration file setting. Defines whether to create PDF Thumbnails. See 5-47 for additional information.
DebugMode	Enables debug mode. See 5-50 for additional information.
DebugStdConversion	Enables debug for the standard conversion process. See 5-51 for additional information.
DefaultNativeTimeout	Sets the default native file timeout. See 5-52 for additional information.
DefaultPostscriptTimeout	Sets the default postscript timeout. See 5-53 for additional information.
DistillerNormJobSetting	Sets the path to the defined Distiller job options. See 5-55 for additional information.
DistillerOptJobSetting	Sets the path to the Distiller job options. See 5-56 for additional information.
FrameMakerexePath	Sets the FrameMaker executable path. See 5-59 for additional information.

Configuration Variable	Brief Description
ImageAlchemyExePath	Sets the Image Alchemy directory path. See 5-60 for additional information.
IsThumbnailPresent	Evaluates whether Thumbnails is present. See 5-61 for additional information.
MaxNumRecursiveStepDefinitions	Specifies the maximum number of recursive step definitions allowed. See 5-64 for additional information.
MSPubexePath	Sets the MS Publisher executable path. See 5-66 for additional information.
OptimizePDF	Optimizes the PDF files at conversion. See 5-68 for additional information.
PageMakerExePath	Defines the path to the PageMaker executable file. See 5-69 for additional information.
PostprocessPDFPath	Path to an executable that can be used to postprocess PDF files. See 5-70 for additional information.
PrinterPortPath	Defines the printer port path. See 5-76 for additional information.
UseAutoCad2000	Defines whether to process AutoCad2000 content for conversion. See 5-84 for additional information.
UseAutocadModelSpace	Defines whether to use AutoCad2000 model space. See 5-85 for additional information.

COMMONLY USED CONFIGURATION VARIABLES

AuthorDelete

Description

Enables author delete privileges.

This is a content item security option. Defines whether the author can delete authored content.

- When set to TRUE, enables the author to delete revision.
- Default is FALSE.

Example

As configuration entry, defines whether the author can delete content.

```
AuthorDelete=true
```

As script, enables the author to delete revision:

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

EnableDocumentHighlight

Description

Enables content item highlighting.

This variable is set with the System Properties Utility and is included on the search.cfg configuration file when the system is initialized.

- When set to TRUE, words are highlighted in PDF, HTML, and TXT files when returned from a full text search.
- Default is TRUE.

Returns a Boolean value:

- Returns TRUE if enabled (returns value of configuration setting).
- Returns FALSE if not enabled.

Example

As configuration setting, enables content item highlighting:

```
EnableDocumentHighlight=true
```

As script, returns the value of the configuration setting:

```
<$EnableDocumentHighlite$>
```

EnterpriseSearchAsDefault

Description

Defines whether Enterprise Search is the default setting.

- When set to TRUE, enables Enterprise Search as the default value.
- There is no default during installation.

Example

Used as configuration entry:

```
EnterpriseSearchAsDefault=true
```

ExclusiveCheckout

Description

Defines check out rights.

- When set to TRUE, only original contributor can check out a content item.
- Default is FALSE.

Returns a Boolean value:

- Returns TRUE if enabled (returns value of configuration setting).
- Returns FALSE if not enabled.

Example

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

```
ExclusiveCheckout=true
```

As script, returns the value of the configuration setting:

```
<$ExclusiveCheckout$>
```

GetCopyAccess

Description

Permits user to get a copy of a content item when only read access is defined.

- When set to TRUE, retrieves a copy when read access is defined.
- Default is FALSE.

Returns a Boolean value:

- Returns TRUE if enabled (returns value of configuration setting).
- Returns FALSE if not enabled.

Example

As configuration setting, retrieves a copy when read access is defined.

```
GetCopyAccess=true
```

As script, returns the value of the configuration setting:

```
<$GetCopyAccess$>
```

HasExternalUsers

Description

Notifies the system of an external user database.

Value variable (evaluated once at the beginning of the service call). Configured with Windows NT/2000 security and uses NtlmSecurityEnabled.

This setting notifies the content server of a custom integration with an external user database. For example, if a custom component has been written to support LDAP integration this flag should be set to TRUE.

- This configuration entry must be manually edited.
- Default is FALSE.

Returns a Boolean value:

- Returns TRUE if the system has defined external users.
- Returns FALSE if the system has no external users.

See Also: NtlmSecurityEnabled (5-30)

Example

As configuration setting, notifies the system of an external user database:

```
HasExternalUsers=true
```

As script, returns the value of the configuration setting:

```
<$HasExternalUsers$>
```

NtlmSecurityEnabled

Description

Enables the Windows NT/2000 LAN Manager (NTLM) authentication process.

This is a security option setting. Set this option only if IIS is being used and if IIS has NTLM authentication enabled and Microsoft Network integration is enabled.

- When set to TRUE, NTLM security can be used to login to the content server.
- Default is FALSE.

See Also: HasExternalUsers (5-29)

DefaultAuthType (5-94)

IntradocRealm (5-98)

Example

Enables NTLM security:

```
NtlmSecurityEnabled=true
```

SelfRegisteredAccounts

Description

Defines default account information of a self-registered user.

- Sets default account information for user with comma delimited values. Used when *UseSelfRegistration* is enabled in the configuration files.
- Returns the defined account information as string.

See Also: UseSelfRegistration (5-36)

Example

As configuration setting, defines default account information:

```
SelfRegisteredAccounts=#none(RWDA),USERS/<$NewUser$>,BOS
(R),SEA(RW),MSP(RWD)
```

As script, returns the defined account information as string:

```
<$SelfRegisteredAccounts$>
```

SelfRegisteredRoles

Description

Defines default roles of a self-registered user.

- Sets default roles for user with comma delimited values. Used when *UseSelfRegistration* is enabled in the configuration files.
- Returns the defined roles as string.

See Also: UseSelfRegistration (5-36)

Example

As configuration setting, defines default roles:

```
SelfRegisteredRoles=guest,salesRole
```

As script, returns the defined roles as string:

```
<$SelfRegisteredRoles$>
```

ShowOnlyKnownAccounts

Description

Defines whether to display only known accounts on Check In page.

- When set to TRUE, only known accounts are displayed.
- When set to FALSE, all accounts are displayed.
- Default is FALSE.

Example

Used as configuration entry:

```
ShowOnlyKnownAccounts=true
```

SysAdminAddress

Description

Defines the system administrator email address.

This is an e-mail configuration entry used with Workflow notification. This value is defined in the System Properties utility and is included in the configuration file when the system is initialized.

- A system administrator address should be defined during installation. This entry can be updated on reinstallation of the content server and Inbound Refinery.
- Returns the value defined in the configuration file.
- Returns a string.

Example

As configuration setting, defines the system administrator email address:

```
SysAdminAddress=admin@companyname.com
```

As script, returns the value of the configuration setting:

```
<$SysAdminAddress$>
```

UseAccounts

Description

Defines whether to enable the use of accounts.

- This is a security option setting.
- When set to TRUE, the use of accounts is enabled.
- Default is FALSE.

Example

Used as configuration entry:

```
UseAccounts=true
```

UseSelfRegistration

Description

Enables self-registration for users.

- Provides the functionality of self-registering to the database and becoming a global user.
- Default roles and accounts can be set for the self-registered user using *SelfRegisteredRoles* and *SelfRegisteredAccounts*.
- When set to TRUE enables an icon and link under the login link titled “Self-Registration.”
- Default is FALSE.

Returns a Boolean value:

- Returns TRUE when enabled in the configuration files.
- Returns FALSE if not enabled.

See Also: SelfRegisteredAccounts (5-31)

SelfRegisteredRoles (5-32)

Example

As configuration setting, enables self registration as a global user:

```
UseSelfRegistration=true
```

As script, returns the value of the configuration setting:

```
<$UseSelfRegistration$>
```

UseSSL

Description

Enables the Secure Sockets Layer.

This can also be enabled in the System Properties utility. Use the Secure Sockets Layer only if you are using a SSL enabled web server.

This is a flag affecting the variables *HttpWebRoot* and *HttpCgiPath* and is generally set in the configuration files.

- When set to TRUE, the secure sockets layer is used (https instead of http).
- Default is FALSE.

Returns a Boolean value:

- Returns TRUE if Secure Sockets Layer is enabled.
- Returns FALSE if SSL is not enabled.

See Also: HttpCgiPath (4-20)

HttpWebRoot (4-26)

Example

As configuration setting, enables SSL:

```
UseSSL=true
```

As script, returns the value of the configuration setting:

```
<$UseSSL$>
```

INBOUND REFINERY CONFIGURATION VARIABLES

Important: Configuration variables returning a string defining a directory or file path related to the Inbound Refinery will return that path relative to the Inbound Refinery; these paths are generally not recognizable to the content server nor usable to the user if passed as information within an HTM page.

AcadUseLISPInterface

Description

Sets the AutoCAD LISP interface.

- There is no default value on install.
- When set to TRUE the LISP interface is used. When TRUE the user will need to register the acadunsupp.arx in AutoCad 14 before converting.
- This configuration entry must be manually edited.

See Also: UseAutoCad2000 (5-84)

Example

Used as configuration entry:

```
AcadUseLISPInterface=true
```

AdditionalIndexBuildParams

Description

Provides additional parameters to add to the Index command line.

Used to add additional build parameters to every “mkvdk.exe”execution. This can be used to force merging or optimization to occur every 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 consume many file handles.

- This configuration entry must be manually edited.
- Other than `-maxmerge` these build parameters are rarely used.
- There is no default value on install.

Additional Build Parameters:

Parameter	Description
-housekeep	Enables automated clean up activity. Rarely used.
-maxclean	Remove out-of-date collection files. Rarely used.
-maxmerge	Merges or rolls up separate partitions into one partition.
-optimize	Enables automated optimization activity. Rarely used.
-purge	Purges collection files. Rarely used.
-repair	Repairs a collection. Rarely used.
-squeeze	Removes old documents from the collection. Rarely used.
-vdbopt	Configures smaller units in the collection (Verity databases or VDBs) so that the data is “linearized.” Rarely used.

Example

Used as configuration entry:

```
AdditionalIndexBuildParams=-maxmerge
```

AdjustPrinterMargins

Description

Adjusts the defined printer margins.

- Printer margin adjustment is defined in pixels.
- Entry must be manually edited.
- There is no default value on install.

Example

Used as configuration entry:

```
AdjustPrinterMargins=100
```

AllowPassthru

Description

Sets the handling of files that time out.

- Defines whether the Inbound Refinery should pass files through in their native formats when timed out.
- When set to TRUE, files are passed through in their native formats when timed out.
- When set to FALSE, the Inbound Refinery will not pass files through in their native formats when timed out.
- Default is TRUE.

Example

Used as configuration entry:

```
AllowPassthru=false
```

AutoCad2000PlotterFilePath

Description

Defines the AutoCad2000 plotter file path used by the Inbound Refinery.

- This entry is set during installation.
- Returns file path as string.

Example

As configuration entry, defines the AutoCad2000 plotter file path:

```
AutoCad2000PlotterFilePath=c:/autocad2000/plotter/
```

As script, returns the file path as string:

```
<$AutoCad2000PlotterFilePath$>
```

CaptureProgram

Description

Defines the path to the Adobe Capture executable.

- This path is defined during installation and can be updated upon reinstallation of the content server or Inbound Refinery.
- Returns a string.

See Also: PrimarySlave (5-75)

Example

Used as configuration entry:

```
CaptureProgram=c:/software/adobe/capture.exe
```

ComputerName

Description

References the name assigned to the computer.

- Returns a string.
- Default is the pre-assigned computer name.

Example

Used as configuration entry:

```
ComputerName=maincomputer
```

ConnectionName

Description

References the name assigned to the connection.

- Returns the connection name as string.
- Default is the pre-assigned connection name.

Example

Used as configuration entry:

```
ConnectionName=mainconnection
```

CreatePDFThumbnails

Description

Defines whether to create PDF Thumbnails and is set during installation of the Inbound Refinery.

- Configuration file setting.
- When set to TRUE, PDF Thumbnails is enabled.
- When set to FALSE, PDF Thumbnails is disabled.
- Default is FALSE.

Example

Used as configuration entry:

```
CreatePDFThumbnails=true
```

CustomConversionWaitTime

Description

Sets the custom conversion wait time in seconds. For example, a wait time of 600 (ten minutes) can be defined if applicable.

- This setting can be updated using the Inbound Refinery configuration options.
- Default is 60 seconds.

Example

Used as configuration entry:

```
CustomConversionWaitTime=60
```

CustomConverterPath

Description

Defines the path used as a process to perform custom conversions.

- This setting can be updated using the Inbound Refinery configuration options.
- Default is an empty string.

Example

Used as configuration entry:

```
CustomConverterPath=c:/out/custom.bat
```

DebugMode

Description

Defines whether to run in debug mode and is set during installation and on update of the content server and Inbound Refinery.

- There is no default during installation.
- When set to TRUE, the debug mode is enabled.
- When set to FALSE, the debug mode is disabled.

Example

Used as configuration entry:

```
DebugMode=true
```

DebugStdConversion

Description

Defines whether to debug the standard conversion process and is set during installation and on update of the content server and Inbound Refinery.

- There is no default during installation.
- When set to TRUE, the standard conversion process is debugged.
- When set to FALSE, the debugging is disabled.

Example

Used as configuration entry:

```
DebugStdConversion=true
```

DefaultNativeTimeout

Description

Defines the default native file timeout and can be set on the Shared Inbound Refinery Configuration screen: Timeout Values tab.

These flags in the configuration file change the content of this variable:

Flag	Description
#factor	Specifies the Native Conversion Factor, which is the value the file size is multiplied by the factor to determine the amount of time to process a content item. The default value is 3. Increase this value for busy or slow systems.
#max	Specifies the Minimum Conversion Wait Time, which is the minimum time in seconds that the Inbound Refinery waits for the conversion process (native or PostScript) to complete. The default time is 60 seconds.
#min	Specifies the Maximum Conversion Wait Time, which is the maximum time in seconds that the Inbound Refinery waits for the conversion process to complete (native or PostScript). The default time is 60 seconds.

❖ Returns the native file timeout value (defined in the configuration file).

Example

Used as configuration entry:

```
DefaultNativeTimeout#factor=3
```

DefaultPostscriptTimeout

Description

Defines the default postscript timeout and can be set on the Shared Inbound Refinery Configuration screen: Timeout Values tab.

The flags in the configuration file change the content of this variable:

Flag	Description
#factor	Specifies the Postscript Conversion Factor, which is the value the file size is multiplied by the factor to determine the amount of time to process a postscript file. The default value is 4. Increase this value for busy or slow systems.
#max	Specifies the Minimum Conversion Wait Time, which is the minimum time in seconds that the Inbound Refinery waits for the conversion process (native or PostScript) to complete. The default time is 60 seconds.
#min	Specifies the Maximum Conversion Wait Time, which is the maximum time in seconds that the Inbound Refinery waits for the conversion process to complete (native or PostScript). The default time is 60 seconds.

- Returns the default postscript timeout value (defined in the configuration file).

Example

Used as configuration entry:

```
DefaultPostscriptTimeout#factor=4
```

DistillerInOutRootDir

Description

Defines the path to Adobe Acrobat Distiller's watched folders.

- This path is defined by default upon Adobe install during Inbound Refinery installation and can be updated upon reinstallation of the content server and Inbound Refinery.
- Returns the path as string.

Example

Used as configuration entry:

```
DistillerInOutRootDir=c:/watched_folder/IdcRefinery
```

DistillerNormJobSetting

Description

Sets the path to the Distiller job options for normal PDF conversion.

- This path is defined by default during Inbound Refinery installation and can be updated upon reinstallation of the content server and Inbound Refinery.
- Returns the path as string.

Example

Used as configuration entry:

```
DistillerNormJobSetting=C:/stellent/IdcRefinery/shared/IntradocPDF.joboptions
```

DistillerOptJobSetting

Description

Sets the path to the Distiller job options for optimized PDF conversion.

- This path is defined by default during the Inbound Refinery installation and can be updated upon reinstallation of the content server and Inbound Refinery.
- Returns the path as string.

Example

Used as configuration entry:

```
DistillerOptJobSetting=C:/stellent/IdcRefinery/shared/IntradocPDFOptimized.joboptions
```

DocConverterEngineDir

Description

References the location of the Inbound Refinery executables.

- Provides the location of the three executables required for the Inbound Refinery. All work is done in this directory (and subdirectories). This path is defined by default during Inbound Refinery installation and can be updated upon reinstallation of the content server and Inbound Refinery.
- Returns the path as string.

Example

Used as configuration entry:

```
DocConverterEngineDir=c:/stellent/IdcRefinery/connections/main
```

EnableErrorFile

Description

Generates a BatchLoader 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 entry on install.
- Must be manually set to enable.

Example

Used as configuration entry:

```
EnableErrorFile=true
```

FrameMakerexePath

Description

Sets the FrameMaker executable path.

- Defines the FrameMaker executable path. This path is defined by default during the Inbound Refinery installation and can be updated upon reinstallation of the content server and Inbound Refinery.
- Returns the path as string.

Example

Used as configuration entry:

```
FrameMakerexePath=c:/software/FrameMaker/FrameMaker.exe
```

ImageAlchemyExePath

Description

Sets the Image Alchemy directory path.

- Defines the directory path to the Image Alchemy executable file. The default path is defined during installation and can be updated upon reinstallation of the content server or Inbound Refinery.
- Returns the path as string.

Example

Used as configuration entry:

```
ImageAlchemyExePath=c:/software/ImageAlchemy.exe
```

IsThumbnailPresent

Description

Evaluates whether Thumbnails is detected during installation of the Inbound Refinery. This entry can be updated upon reinstallation of the Inbound Refinery.

- Sets entry to TRUE if Thumbnails is present.
- Sets no entry if Thumbnails is not present.

Example

Used as configuration entry:

```
IsThumbnailPresent=true
```

JvmCommandLine

Description

Defines the location of the Java Virtual Machine command line.

- Referenced in the *intradoc.cfg* file located in the <*install_dir*>/IdcRefinery/connections/main directory.
- This entry can be updated upon reinstallation of the content server and Inbound Refinery.
- There is no default entry on install.

Example

Used as configuration entry:

```
JvmCommandLine=msjavx86.exe
```

MaxConverterTimeOut

Description

Defines the maximum amount of time the Inbound Refinery will wait before timing out either the Native or Postscript conversion processes.

- Configuration file setting. Can be set on the Shared Inbound Refinery Configuration screen: Timeout Values tab. Sets the timeout in seconds.
- Default is 600 (ten minutes).

Example

Used as configuration entry:

```
MaxConverterTimeOut=600
```

MaxNumRecursiveStepDefinitions

Description

Specifies the maximum number of recursive step definitions allowed.

- There is no default on install.
- This entry must be manually edited.

Example

Used as configuration entry:

```
MaxNumRecursiveStepDefinitions=###
```

MinConverterTimeOut

Description

Sets the minimum timeout in seconds.

- Defines the minimum amount of time the Inbound Refinery will wait before timing out either the Native or Postscript conversion processes.
- This value can be set on the Shared Inbound Refinery Configuration screen: Timeout Values tab.
- Default is 60 (one minute).

Example

Used as configuration entry:

```
MinConverterTimeOut=60
```

MSPubexePath

Description

Defines the MS Publisher executable path.

- This path is set during installation and can be updated upon reinstallation of the content server or Inbound Refinery.
- Returns the path as string.

Example

Used as configuration entry:

```
MSPubexePath=c:/software/mspub.exe
```

NativeTimeConversionFactor

Description

Specifies the Native Conversion Factor, which is the value the file size is multiplied by the factor to determine the amount of time to process a postscript file.

- The default value can be increased for extremely large files or slower/ busy systems.
- Defines the length of time a file goes through the native conversion process. This entry can be updated on the Shared Inbound Refinery Configuration screen: Timeout Values tab.
- Default is a factor of 3.

Example

Used as configuration entry:

```
PostscriptTimeConversionFactor=4
```

OptimizePDF

Description

Optimizes the PDF files at conversion.

- Optimizing removes graphics and text that appear multiple times in the content item, and replaces them with pointers to the first display of them. Optimizing enables you to begin to view the first pages of a content item before the entire content item is downloaded.
- This entry can be updated using the Local Inbound Refinery Configuration screen: Native Options tab.
- When set to TRUE, optimizes the PDF files at conversion.
- Default is an empty string.

Example

Used as configuration entry:

```
OptimizePDF=true
```

PageMakerExePath

Description

Defines the path to the PageMaker executable file.

- This path is defined during installation.
- This entry can be updated upon reinstallation of the content server or Inbound Refinery.

Example

Used as configuration entry:

```
PageMakerExePath=c:/software/adobe/pagemaker.exe
```

PostprocessPDFPath

Description

Path to an executable that can be used to postprocess pdf files.

- The pdf file path is the only command line parameter. 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 DocConverter 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 DocConverter will attempt to run the process.
- Default is an empty string.
- This setting can be updated using the Inbound Refinery configuration options.

Example

Used as configuration entry:

```
PostprocessPDFPath=c:/out/custom.bat
```

PostscriptTimeConversionFactor

Description

Specifies the PostScript Conversion Factor, which is the value the file size is multiplied by the factor to determine the amount of time to process a postscript file. The default value can be increased for extremely large files or slower/ busy systems.

- Defines the length of time a file goes through the postscript conversion process. This entry can be updated on the Shared Inbound Refinery Configuration screen: Timeout Values tab.
- Configuration file setting.
- Default is a factor of 4.

Example

Used as configuration entry:

```
PostscriptTimeConversionFactor=4
```

PreConversionPath

Description

Path used as a process to perform actions prior to every conversion.

- This entry is a Inbound Refinery configuration option.
- Default is an empty string.

Example

Used as configuration entry:

```
PreConversionPath=c:/out/preconversion.bat
```

PreviewOutputExtension

Description

Defines what extension is output from Publisher that is to be used in HTML Preview.

- Default is HCSP.

Example

Used as configuration entry:

```
PreviewOutputExtension=jsp
```

PreviewPath

Description

References the tcpreview.exe file used in HTML Preview. This executable resides in the directory path of Publisher.

- This path is defined during installation.
- This entry can be updated upon reinstallation of HTML Preview.

Example

Used as configuration entry:

```
PreviewPath=C:/program files/Stellent content
publisher/tcpreview.exe
```

PrimarySlave

Description

Defines whether Adobe Capture is used as the primary slave setting.

- This entry can be updated upon reinstallation of the content server or Inbound Refinery.
- When set to NONE, a primary slave is not set.
- When set to CAPTURE, the Abobe Capture executable is used.
- Option is defined during installation.

See Also: CaptureProgram (5-44)

Example

Used as configuration entry:

```
PrimarySlave=CAPTURE
```

PrinterPortPath

Description

Defines the printer port path.

- This path is defined during installation.
- This entry can be updated upon reinstallation of the content server or Inbound Refinery.

Example

Used as configuration entry:

```
PrinterPortPath=c:/temp/idcoutput.ps
```

ProcessHyperlinks

Description

Enables the processing of hyperlinks in Word and PowerPoint content items.

- This is a Inbound Refinery configuration option.
- When set to FALSE, hyperlinks in Word and PowerPoint content items are not processed.
- Default is TRUE.

Example

Used as configuration entry:

```
ProcessHyperlinks=true
```

ShowHyperlinkBox

Description

Defines whether 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 TRUE.

Example

Used as configuration entry:

```
ShowHyperlinkBox=true
```

TerminateAcrobat

Description

Defines the condition for terminating Adobe Acrobat.

- Calls the *TerminateAcrobat.exe* executable. Used when defining the condition for terminating the Adobe Acrobat application.
- There is no default during installation.

Example

Can be used to define the conditions under which Acrobat is terminated:

```
<$TerminateAcrobat$>
```

ThumbnailHeight

Description

Sets the Thumbnail height in pixels.

- Default is 80 pixels.
- There is no default if the Thumbnails feature is not installed.

Example

As configuration setting, defines the Thumbnail height in pixels:

```
ThumbnailHeight=80
```

As script, returns the value of the configuration setting:

```
<$ThumbnailHeight$>
```

ThumbnailWidth

Description

Sets the Thumbnail Width in pixels.

- Default is 80 pixels.
- There is no default if the Thumbnails feature is not installed.

Example

As configuration setting, defines the Thumbnail height in pixels:

```
ThumbnailWidth=80
```

As script, returns the value of the configuration setting:

```
<$ThumbnailWidth$>
```

TimeoutPerOneMegInSec

Description

Sets the time out in seconds multiplied by one megabyte.

- Configuration file setting.
- This entry is defined during installation of the Inbound Refinery.

Example

Used as configuration entry:

```
TimeoutPerOneMegInSec=###
```

UseAdobePDFMaker

Description

Defines whether to use PDFMaker for the conversion of Word files.

- This entry enables the user to utilize the Adobe PDFMaker application to convert Word files into PDF format. Using this option makes the PDFMaker responsible for processing all Word conversions. When set to TRUE, the PDFMaker application is used for the conversion of Word files.
- Default is FALSE
- This is a Inbound Refinery configuration option.

Example

Used as configuration entry:

```
UseAdobePDFMaker=true
```

UseAutoCad2000

Description

Defines whether to process AutoCad2000 content for conversion.

- If set to FALSE, AutoCad2000 content is not processed.
- Default is set to TRUE if AutoCad2000 is detected during installation of the Inbound Refinery.
- There is no default entry if AutoCad2000 is not detected.

Example

Used as configuration entry:

```
UseAutoCad2000=true
```

UseAutocadModelSpace

Description

Defines whether to use AutoCad2000 model space.

- If set to FALSE, AutoCad2000 model space is not used.
- Default is set to TRUE if AutoCad2000 is detected during installation of the Inbound Refinery.
- There is no default entry if AutoCad2000 is not detected.

Example

Used as configuration entry:

```
UseAutocadModelSpace=true
```

VerboseMode

Description

Specifies whether to use Verbose logging.

- If verbose logging is enabled, the Inbound Refinery log shows the related configuration information. For example, whether the PDF is optimized, where the custom converter path is located, and how much time is allowed for custom conversions.
- When set to TRUE, verbose logging is enabled. This option can be set on the Local Inbound Refinery Configuration screen: General tab.
- Default is FALSE.

Example

Used as configuration entry:

```
VerboseMode=false
```

BATCH LOADER CONFIGURATION VARIABLES

BatchLoaderPath

Description

Defines the BatchLoader file path used by the Inbound Refinery.

- Returns file path as string.
- Path is set during installation.

See Also: BatchLoaderUserName (5-88)

CleanUp (5-89)

MaxErrorsAllowed (5-90)

Example

As configuration entry (Windows NT/2000 example), defines the BatchLoader file path:

```
BatchLoaderPath=c:/stellent/BatchLoader/batchfile.txt
```

As configuration entry (Solaris/UNIX example), defines the BatchLoader file path:

```
BatchLoaderPath=/u1/intradoc3/batLd/Load4
```

As script, returns the file path as string:

```
<$BatchLoaderPath$>
```

BatchLoaderUserName

Description

Defines the authorized user name for the BatchLoader utility. The user name used must belong to the admin role

- Default is `sysadmin`.
- Returns the BatchLoader user name as string.

See Also: BatchLoaderPath (5-87)

CleanUp (5-89)

MaxErrorsAllowed (5-90)

Example

Used as configuration entry:

```
BatchLoaderUserName=sysadmin
```

CleanUp

Description

Used for BatchLoader file clean up.

- When enabled, cleans up files after successful check in using the BatchLoader.
- When set to YES, deletes the file from the hard drive after it is successfully checked in or updated.
- Default is NO.

See Also: BatchLoaderPath (5-87)

BatchLoaderUserName (5-88)

MaxErrorsAllowed (5-90)

Example

Used as configuration entry:

```
CleanUp=No
```

MaxErrorsAllowed

Description

Sets the number of errors after which the BatchLoader stops processing records from the batch load file.

- If you plan to run the BatchLoader with a large number of content items overnight, then increase the default value. If you monitor the BatchLoader with a small amount of content items, then decrease the default value.
- Default is 50.
- Returns value as integer.

See Also: BatchLoaderPath (5-87)

BatchLoaderUserName (5-88)

CleanUp (5-89)

Example

Used as configuration entry:

```
MaxErrorsAllowed=50
```

WEB FILTER CONFIGURATION VARIABLES

CGI_DEBUG

Description

Used to debug the web server filter.

- Defines whether to log all in going and out going traffic. This configuration entry instructs the code that translates from a *call to the web server* to a *request to the content server* to log all in going and out going traffic. This configuration entry must be manually edited.
- When set to TRUE, logs all in going and out going traffic.
- Default is FALSE.

See Also: CGI_RECEIVE_DUMP (5-92)

CGI_SEND_DUMP (5-93)

Example

Used as configuration entry:

```
CGI_DEBUG=true
```

CGI_RECEIVE_DUMP

Description

Defines whether the Common Gateway Interface is enabled to receive an information dump.

- This configuration entry must be manually edited.
- When set to TRUE, the CGI is enabled to receive an information dump.
- There is no default value on install.

See Also: CGI_DEBUG (5-91)

CGI_SEND_DUMP (5-93)

Example

Used as configuration entry:

```
CGI_RECEIVE_DUMP=true
```

CGI_SEND_DUMP

Description

Defines whether the Common Gateway Interface is enabled to send an information dump.

- This configuration entry must be manually edited.
- When set to TRUE, the CGI is enabled to send an information dump.
- There is no default value on install.

See Also: CGI_DEBUG (5-91)

CGI_RECEIVE_DUMP (5-92)

Example

Used as configuration entry:

```
CGI_SEND_DUMP=true
```

DefaultAuthType

Description

Sets the default authentication challenge type.

- This is a security option setting. Set this to *NTLM* to define the default authentication challenge as a Microsoft Network login when accessing a protected Stellent resource.
- For example, if a user sends a secured URL by e-mail to a user who has never logged into the system, when the recipient clicks on the URL they will receive an NTLM challenge.
- Once the user has logged in, the browser will remember which login choice was made (the database or Microsoft Network login) even after the browser is closed and restarted. This means if you click on a URL in e-mail it will remember which authentication protocol to use (HTTP Basic for Stellent database or NTLM for Microsoft Network).
- Used as Windows NT/2000 only security option with an IIS install. This setting is not used if Netscape is the default browser. Set this only if you want the current default behavior to change. When set to NMTL, a Microsoft Network login NTLM challenge is displayed. This configuration entry must be manually edited.

See Also: NtlmSecurityEnabled (5-30)

IntradocRealm (5-98)

Example

Used as configuration entry:

```
DefaultAuthType=NTLM
```

DefaultMasterDomain

Description

Defines the default master domain. If not set, this value is the domain of the Windows NT/2000 server machine that is hosting the web server.

- This value can be set to override the standard behavior, and force the ISAPI filter to designate a different domain as its default master domain.
- In particular, Windows NT/2000 groups from that domain will not have a `DOMAINNAME\` prefix added to their name before being translated to roles and if a user logs in without specifying a domain, the default master domain is assumed.
- This configuration entry can be edited in the *config.cfg* file or included in the registry under HKEY_LOCAL_MACHINE/SOFTWARE/IntraNet/IdcAuth/. If the value is added to the registry, the entry will apply to all the ISAPI filters installed on that machine.
- Default is the domain of the Windows NT/2000 server machine that is hosting the web server.
- Returns the specified domain name as string.

Example

Used as configuration entry:

```
DefaultMasterDomain=masterdomain
```

DefaultNetworkAccounts

Description

Defines the default network account.

- A user is automatically assigned the `#none` account. By setting this value, a different set of accounts can be automatically granted to all users.
- The accounts should be put into a comma-separated list with no spaces in between, for example: `#none,publicweb,notices`.
- The `#none` entry grants privileges to content items that have no account assigned. The `#all` entry grants privileges to all accounts. This entry is ignored if the user is defined in the content server database.
- This configuration entry can be edited in the *config.cfg* file or included in the registry under HKEY_LOCAL_MACHINE/SOFTWARE/IntraNet/IdcAuth/. If the value is added to the registry, the entry will apply to all the ISAPI filters installed on that machine.
- Default is #none.

Example

Used as configuration entry:

```
DefaultNetworkAccounts=#none
```

FILTER_DEBUG

Description

Defines whether the web server plug-in filter enables debug output.

- When set to TRUE, the web server plug-in filter enables the debug output. This entry must be manually edited.
- Default is FALSE.

Example

Used as configuration entry:

```
FILTER_DEBUG=false
```

IntradocRealm

Description

Defines the server realm.

- This entry specifies the realm for either 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, and must be authenticated against the entire Microsoft Network. This file must be manually edited.
- There is no default during installation.

See Also: NtlmSecurityEnabled (5-30)

DefaultAuthType (5-94)

Example

Used as configuration entry:

```
IntradocRealm=main
```

LocalGroupServer

Description

Specifies the Windows NT/2000 server to interrogate for the local groups that contain the user as a member.

- If this entry is not set, the controller of the default master domain is used. This entry can be updated on reinstallation of the content server and Inbound Refinery.
- This configuration entry can be edited in the *config.cfg* file or included in the registry under HKEY_LOCAL_MACHINE/SOFTWARE/IntraNet/IdcAuth/. If the value is added to the registry, the entry will apply to all the ISAPI filters installed on that machine.
- There is no default entry, a local group server name should be provided during installation.

Example

Used as configuration entry:

```
LocalGroupServer=localservername
```

NetworkAdminGroup

Description

Maps the Domain Admin Windows NT/2000 group to the content server role admin.

- The content server maps the Domain Admin Windows NT/2000 group in the default master domain to the content server role admin. By setting this value, a different Windows NT/2000 group can be mapped to the admin role.
- This configuration entry can be edited in the *config.cfg* file or included in the registry under HKEY_LOCAL_MACHINE/SOFTWARE/IntraNet/IdcAuth/. If the value is added to the registry, the entry will apply to all the ISAPI filters installed on that machine.
- Default setting is to map.

Example

Used as configuration entry:

```
NetworkAdminGroup=c:/stellent/admin/
```

SpecialAuthGroups

Description

Defines named special authorization groups.

- User must define named special authorization groups. This entry must be manually edited.
- There is no default during installation.
- Returns a string.

Example

Used as configuration entry:

```
SpecialAuthGroups=executive
```

UseLocalGroups

Description

Defines whether the Windows NT/2000 server checks to see if the user belongs to any of the local groups on that server.

- The local groups are mapped to roles and accounts as if they were domain groups in the default master domain. When set to TRUE, the Windows NT/2000 server checks whether the user belongs to any of the local groups on that server.
- This configuration entry can be edited in the *config.cfg* file or included in the registry under HKEY_LOCAL_MACHINE/SOFTWARE/IntraNet/IdcAuth/. If the value is added to the registry, the entry will apply to all the ISAPI filters installed on that machine.
- Default is FALSE.

Example

Used as configuration entry, enables Windows NT/2000 security:

```
UseLocalGroups=1
```

WebServerAuthOnly

Description

Used to enable web server authorization only:

- If enabled (set to the value 1), the ISAPI filter will not validate passwords against the internal user names and password from the content server database. In particular, if Basic Authentication is turned on in IIS then this option must be enabled. This enables browsers to use Basic authentication when logging into the Microsoft Network. In particular, a user can login using a Microsoft username and password using the Netscape browser.
- If this option is enabled, new browser users can only be added by adding them to an Windows NT/2000 domain since the users are not authenticated against the content server user database. However, if the user database happens to have a user of the same name as an Windows NT/2000 name, then the content server security profile is used instead of the Windows NT/2000 profile.
- This configuration entry can be edited in the *config.cfg* file or included in the registry under HKEY_LOCAL_MACHINE/SOFTWARE/IntraNet/IdcAuth/. If the value is added to the registry, the entry will apply to all the ISAPI filters installed on that machine.
- Default is 0 (not enabled).

Example

Used as configuration entry to enable web server authorization only:

```
WebServerAuthOnly=1
```

MISCELLANEOUS CONFIGURATION VARIABLES

AccountMapPrefix

Description

Defines the prefix used to identify which Windows NT/2000 groups map to accounts.

- The default account symbol that is used as a prefix for Windows NT/2000 groups is @. This entry can be used to set a different sequence of characters to be used to identify which Windows NT/2000 groups map to accounts.
- This configuration entry can be edited in the *config.cfg* file or included in the registry under HKEY_LOCAL_MACHINE/SOFTWARE/IntraNet/IdcAuth/. If the value is added to the registry, the entry will apply to all the ISAPI filters installed on that machine.
- Default is @.

Example

Used as configuration entry:

```
AccountMapPrefix=@
```

AllowAlternateMetaFile

Description

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

- When set to TRUE, allows users to submit 'metadata-only' content for the alternate file.
- Default is FALSE.
- This entry must be manually edited.

Example

Used as configuration entry:

```
AllowAlternateMetaFile=true
```

AllowPrimaryMetaFile

Description

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

- AllowPrimaryMetaFile is used to enable the meta file creation capability and the flag createPrimaryMetaFile is used to direct the system to actually create the primary file.
- If set to TRUE in the config.cfg file, it will change the look of the check in page by adding an extra check box. If this check box is checked, the createPrimaryMetaFile is set. 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.
- Default is FALSE.
- This entry must be manually edited.

See Also: CreatePrimaryMetaFile (5-111)

Example

Used as configuration entry:

```
AllowPrimaryMetaFile=true
```

AutoNumberPrefix

Description

Defines the automatic numbering prefix.

- This value is defined in the System Properties utility and is included in the configuration file when the system is initialized.
- Returns the automatic numbering prefix (returns value in configuration settings).
- Returns a string.

Example

As configuration setting, defines the automatic numbering prefix.

```
AutoNumberPrefix=HR
```

As script, returns the value of the configuration setting:

```
<$AutoNumberPrefix$>
```

CgiFileName

Description

Defines the Common Gateway Interface file name.

- Specifies the Dynamic Linked Library (.dll) CGI file and the UNIX shared libraries.
- Returns the Common Gateway Interface file name (returns entry defined in configuration files).
- Returns a string.

See Also: HttpCgiPath (4-20)

Example

As configuration setting, specifies the CGI file name:

```
CgiFileName=iis_idc_cgi.dll
```

As script, returns the defined CGI file name as string:

```
<$CgiFileName$>
```

CharMap

Description

Used for localization. Sets the number of characters of the character map.

- Value is set on install and can be updated upon reinstallation of the content server and Inbound Refinery.
- Default is 850.

Example

Used as configuration entry:

```
CharMap=850
```

ColumnMapFile

Description

Required for Oracle only. References the column mapping HTM file.

- Set during installation and can be updated upon reinstallation of the content server and Inbound Refinery.
- There is no default entry.

Example

Used as configuration entry:

```
ColumnMapFile=upper_clmns_map.htm
```

CreatePrimaryMetaFile

Description

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

- The AllowPrimaryMetaFile entry is used to enable the meta file creation capability and the flag createPrimaryMetaFile is used to direct the system to actually create the primary file.
- If the AllowPrimaryMetaFile flag is set to TRUE in the config.cfg file, it will change the look of the check in page by adding an extra check box. If this check box is checked, the createPrimaryMetaFile is set. 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.
- For use in the Batch Loader, the content item needs the CreatePrimaryMetaFile entry to be set to TRUE in the batch load file.

See Also: AllowPrimaryMetaFile (5-106)

Example

Used as configuration entry:

```
CreatePrimaryMetaFile=true
```

DatabasePreserveCase

Description

Defines whether the character case from the database is preserved.

- Required for Oracle and any database that is case sensitive.
- When set to TRUE, the database case for files is preserved.
- Default is TRUE.

Example

Used as configuration entry:

```
DatabasePreserveCase=true
```

DatedCacheIntervalDays

Description

Defines the Dynamic Converter cache removal in days.

- The Dynamic Converter cache removal setting is defined in days. This relates to the dated cache removal functionality.
- There is no default entry.

Example

Used as configuration entry:

```
DatedCacheIntervalDays=100
```

DateOutputFormat

Description

Sets the date output format for Indexer.

- There is no default during installation.
- Defined during installation and on update of the content server and Inbound Refinery.

Example

Used as configuration entry:

```
DateOutputFormat=M/d/yyyy hh:mm:ss a
```

DCMaxFileSize

Description

Sets the maximum size of file for conversion.

- If a file is larger than the specified value, the conversion will not occur.
- Default is 20 megabytes.
- Relates to the DynamicConverter component version 5.0 and greater.

Example

Used as configuration entry:

```
DCMaxFileSize=20000000
```

DCTimeOut

Description

Sets the time out value for the conversion process.

- Defines the number of minutes to wait for the dynamic conversion of a document into HTML. The default time out value for the conversion process is three minutes. If the conversion for large PDF files on your system is slow and times out before three minutes, increase the timeout setting.
- Default is 3 minutes.

Example

Used as configuration entry:

```
DCTimeOut=3
```

DCViewFormat

Description

Sets the file viewing option.

- Content items can be viewed in either their native format or in a web viewable format.
- When set to NATIVE, the content item is viewed in the native format for that file.
- When set to WEBVIEWABLE, the content item is viewed in a web viewable format.
- If no value is specified, this defaults to NATIVE.
- Relates to the DynamicConverter component version 5.0 and greater.

Example

Used as configuration entry:

```
DCViewFormat=webviewable
```

DefaultAccounts

Description

Defines the default account information for the anonymous user.

- When assigning accounts, the read-write privilege must be specified and accounts must be comma delimited.
- Default is NONE.
- Returns the defined account information as string.

Example

As configuration setting, defines default account information:

```
DefaultAccounts=#none(RWDA)
```

As configuration setting, can be used to define access to accounts:

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

As script, returns the account information as string:

```
<$SelfRegisteredAccounts$>
```

DefaultHtmlConversion

Description

References a default template for Dynamic Converter.

- Dynamic Converter specific variable. Defines the default template for conversions. Takes as an entry the content Id of any checked-in template that you want to specify as the default.
- There is no default entry.
- This entry must be manually edited.

Example

Used as configuration entry:

```
DefaultHtmlConversion=
```

DefaultPasswordEncoding

Description

Defines the default password encoding type.

- Indicates the type of password encoding to use when storing passwords. The default is to use the Secure Hash Algorithm update 1 (SHA1). If any other nonempty value is assigned, then no encoding is performed.
- The suggested value to use if you want to have open text passwords is OpenText. This configuration entry must be manually edited.
- Default is SHA1-CB.

Example

Used as configuration entry:

```
DefaultPasswordEncoding=SHA1-CB
```

DirectoryLockingLogPath

Description

Defines the log file used during a temporary locking of directories.

- Enables logging for the directory locking logic. Any errors that occur during the temporary locking of directories will be reported to this log.
- When set to TRUE, named log file is used during a temporary locking of directories.
- There is no default entry.
- This entry must be manually edited.

Example

Used as configuration entry:

```
DirectoryLockingLogPath=C:/temp/locking.log
```

DoDocNameOrder

Description

Presentation option for the Repository Manager application.

- When enabled, the Repository Manager results are sorted by `dDocName`. This option is often disabled (set to FALSE) for better performance.
- When set to TRUE, the results are sorted by `dDocName`.
- When set to FALSE, the results are not sorted.
- Default is TRUE.

Example

Used as configuration entry:

```
DoDocNameOrder=true
```

DownloadApplet

Description

Evaluates whether the Download applet is enabled. Used as Idoc Script and configuration variable.

- Value variable (evaluated once at the beginning of the service call). Checks from the search results page whether the Download applet is enabled. Used as Idoc Script and configuration variable. Retrieves the value of the configuration setting.
- Used as configuration setting to enable the Download applet.

Returns a Boolean value:

- Returns TRUE if the Download applet has been enabled in the System Properties utility.
- Returns FALSE if the Download applet has not been enabled.

See Also: MultiUpload (5-175)

Example

As configuration setting, enables the Download applet:

```
DownloadApplet=true
```

As script, evaluates condition of Download applet:

```
<$DownloadApplet$>
```

ForceDistinctRevLabel

Description

Defines revision label usage.

- Defines whether additional revisions for the same content 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 will not be allowed to have the same revision label.
- Default setting is FALSE.

Example

Used as configuration entry:

```
ForceDistinctRevLabel=true
```

ForceDocTypeChoice

Description

Used on a displayed page to define a Type choice list with a blank entry.

- This is a variable defined in Idoc Script and set on the displayed page.
- Used on pages as a display option.
- When set to TRUE, the *Content Check In* form displays a Type choice list with a blank entry. This requires the user to select a type.
- Default setting is FALSE.

Example

Used as configuration entry:

```
ForceDocTypeChoice=true
```

ForceSecurityGroupChoice

Description

Used on a displayed page to define a Security Group choice list with a blank entry.

- This is a variable defined in Idoc Script and set on the displayed page.
- Used on pages as a display option.
- When set to TRUE, the *Content Check In* form displays a Security Group choice list with a blank entry. This requires the user to select a security group.
- Default setting is FALSE.

Example

Used as configuration entry:

```
ForceSecurityGroupChoice=true
```

HTMLEditorPath

Description

Defines the selected HTML editor executable path for the Component Wizard.

- This path is defined during installation. This entry can be updated upon reinstallation of the content server and Inbound Refinery.
- There is no default entry.

Example

Used as configuration entry:

```
HTMLEditorPath=c:/software/Wordpad.exe
```

HttpRelativeWebRoot

Description

Defines the web root directory.

- Specifies the web root directory as a relative URL. A relative root such as */stellent/* is used rather than a full root such as *http://www.mycomputer.com/stellent/*
- Returns the defined relative web root directory (defined in configuration files).
- Returns a string.

See Also: HttpCommonRoot (4-21)

Example

As configuration setting, defines the relative web root:

```
HttpRelativeWebRoot=/stellent/
```

As script, returns the relative web root as string:

```
<$HttpRelativeWebRoot$>
```

HttpServerAddress

Description

Defines the server address.

- Specifies the server address as a partial URL. A partial URL such as *mycomputer.com* is used rather than a full address such as *http://www.mycomputer.com/*
- Returns the defined server address (defined in configuration files).
- Returns a string.

Example

As configuration setting, defines the relative web root:

```
HttpServerAddress=www.mycomputer.com
```

As script, returns the server address as string:

```
<$HttpServerAddress$>
```

IDC_Admin_Name

Description

Defines the IDC administrator user name.

- Not currently used. Defines a separate IDC administrator name.
- There is no default during installation.

Example

Used as configuration entry:

```
IDC_Admin_Name=sysadmin
```

IDC_Name

Description

Defines the unique instance name.

- This is an option setting. This value must be set on a new install or upgrade.
- The value displays on various interfaces.
- Defined during installation.

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.

See Also: InstanceDescription (5-136)

InstanceMenuLabel (5-137)

Example

Used as configuration entry:

```
IDC_Name=StellentMSP
```

IdcHttpHeaderVariables

Description

Defines data parameters to encode in an HTTP header response.

This configuration variable holds a comma separated list of data parameters to encode in an HTTP header response. The name of the HTTP header generated by the content server when sending a response to an HTTP based client is *IdcVariables*. 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 (less than 0x20, '+', or '%') have been %xx encoded. For example, if the value in the config.cfg was `IdcHttpHeaderVariables=IdcServer,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 change every line feed character into the %0A characters. The blDateFormat is the system date format for the content server.

This configuration entry is specifically designed for web server plug-ins that wish to audit the requests made by the client. The plug-ins can examine the HTTP headers in the content server 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.

Example

Used as configuration entry:

```
IdcHttpHeaderVariables=IdcServer,dDocName
```

IndexerLargeFileSize

Description

Defines the maximum file size for the Indexer.

- There is no default defined during installation.
- This entry can be updated using the System Properties stand-alone application on the content server.

Example

Used as configuration entry:

```
IndexerLargeFileSize=1000
```

IndexerPath

Description

Defines the path to the Indexer program.

- This entry can be updated using the System Properties stand-alone application on the content server.
- The default path is defined during installation.

Example

Used as configuration entry:

```
IndexerPath=c:/search97/_nti31/bin/mkvdk.exe
```

InstanceDescription

Description

Defines the server name for that server instance.

- Provides the defined name of the server instance, such as Master_on_secondserver. Defined during installation and can be updated upon reinstallation of the content server or Inbound Refinery.
- Used as an Idoc Script command within a template file, returns the variable Master_on_<server_instance>

See Also: IDC_NAME (5-131)

InstanceMenuLabel (5-137)

Example

As configuration entry:

```
InstanceDescription=mainserver
```

As script, returns the value of the configuration setting (returns the server instance name):

```
<$InstanceDescription=mainserver$>
```

InstanceMenuLabel

Description

Provides the defined description of the server instance menu label.

- This entry should be defined during installation. This entry can be updated using the System Properties stand-alone application on the content server or upon reinstallation of the content server or Inbound Refinery.
- There is no default defined during installation.

See Also: IDC_NAME (5-131)

InstanceDescription (5-136)

Example

Used as configuration entry:

```
InstanceMenuLabel=mainmenu
```

IntradocDir

Description

Defines the path to the root primary directory.

- The default path to the root primary directory should be defined during installation.
- There is no default entry.

Example

Used as configuration entry:

```
IntradocDir=c:/stellent/
```

IntradocServerHostName

Description

Defines the host name to use when opening a socket connection to the content server.

- This entry defines the host name to use when opening a socket connection to the content server. This entry is used by the CGI code that translates calls from the web server to requests to the Server. This entry must be manually edited.
- Default value is `localhost`.

Note: To call services remotely, IntradocServerHostName must be set to IP or DNS. See *Calling Services Remotely* in the IdcCommand Reference Guide for additional information.

Example

Used as configuration entry:

```
IntradocServerHostName=localhost
```

IntradocServerPort

Description

Defines the port that the ISAPI filter or any other application should use to talk to the content server.

- This entry 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.

This entry defines the port that the web server filter or any other application should use to talk to this content server. In IIS the CGI code is implemented by the iis_idc_cgi.dll ISAPI extension. This entry must be manually edited.

- Default is 4444.

Note: To call services remotely, the server port must be defined. See *Calling Services Remotely* in the IdcCommand Reference Guide for additional information.

Example

Used as configuration entry:

```
IntradocServerPort=4444
```

IsAutoArchive

Description

Enables or disables the automatic import or transfer of content items.

- If set to FALSE, the content server will not automatically import or transfer archives.
- Default is TRUE.

Example

Used as configuration entry:

```
IsAutoArchive=false
```

IsAutoNumber

Description

Enables auto numbering.

- When set to TRUE, content item auto numbering is enabled. Auto numbering automatically assigns a title with the specified prefix. The user does not need to specify a content item name.
- Default is FALSE.

Example

Used as configuration entry:

```
IsAutoNumber=true
```

IsAutoQueue

Description

Enables or disables the processing of content item post-conversion.

- If set to FALSE, the content server will not process content item converted by the Inbound Refinery.
- Default is TRUE.

Example

Used as configuration entry:

```
IsAutoQueuer=false
```

IsAutoSearch

Description

Enables or disables auto export and indexing.

- When set to FALSE, the content server will not automatically index content items or automatically export documents to an archive.
- Default is TRUE.

Example

Used as configuration entry:

```
IsAutoSearch=true
```

IsDynamicConverterEnabled

Description

Enables dynamic converter functionality.

- Defines whether the dynamic converter functionality for the content server is enabled.
- When set to TRUE, the dynamic converter functionality for the content server is enabled.
- Default is FALSE

Example

As configuration setting, enables dynamic converter functionality:

```
IsDynamicConverterEnabled=true
```

As script, used to evaluate whether dynamic converter functionality is enabled:

```
<$if IsDynamicConverterEnabled and
isTrue(IsDynamicConverterEnabled)$>
?IdcService=GET_TEMPLATE_CONVERSIONS"" src="" width=""
border=0?IdcService=GET_TEMPLATE_CONVERSIONS"
?IdcService=GET_TEMPLATE_CONVERSIONS"<$lc("wwTemplateCon
versions")$>
<$endif$>
```

IsFormsPresent

Description

Enables forms features.

- If this field is not present in the configuration file or does not evaluate to TRUE, the forms features are disabled. License checks are not performed and the user will get an error message indicating this parameter is not set when attempting to submit a form.
- Default is TRUE

Example

Used as configuration entry:

```
IsFornsPresent=true
```

IsJdbc

Description

Checks whether the system properties are enabled to allow Java Database Connectivity.

The database must be manually configured if the database is changed or if the system could not find the database during the installation. The database is automatically configured during the installation if there are no network errors that make it impossible for the system to connect to the database. If the runtime version of Microsoft Access is used, there are no database configuration options to set.

This entry is set on install but may be reconfigured by the user. The System Properties utility can be used to enable Java Database Connectivity.

- Defined during installation.

Returns a Boolean value:

- Returns TRUE if Java Database Connectivity is enabled in the System Properties utility (returns the value of the configuration setting).
- Returns FALSE if Jdbc is not enabled.

See Also: JdbcConnectionString (5-153)

JdbcDriver (5-154)

Example

As configuration setting, set the Java Database Connectivity as disabled:

```
IsJdbc=no
```

As script, returns the value of the configuration setting:

```
<$IsJdbc$>
```

IsJdbcLockTrace

Description

Enables Java Database Connectivity lock trace to view the database locking calls.

- This is a Debug setting.
- When set to TRUE, the database locking calls can be viewed in the server output.
- Default is FALSE.

Example

Used as configuration entry:

```
IsJdbcLockTrace=true
```

IsJdbcQueryTrace

Description

Enables Java Database Connectivity query trace to view queries being executed against the database.

- This is a Debug setting.
- When set to TRUE, the queries being executed against the database can be viewed in the server output.
- Default is FALSE.

Example

Used as configuration entry:

```
IsJdbcQueryTrace=true
```

IsJspServerEnabled

Description

Enables Java Server Page functionality.

Java Server Page support enables developers to access and modify Stellent Content Server content, ResultSets, personalization and security definitions, and predefined variables and configuration settings through Java Server Pages rather than through standard Stellent component architecture. Stellent services and Idoc Script functions can also be executed from Java Server Pages, which reside as executable content in the content server.

- This entry must be manually edited.
- Default is FALSE.

See Also: JspEnabledGroups (5-160)

Example

Used as configuration entry:

```
IsJspServerEnabled=true
```

IsOverrideFormat

Description

Checks whether the system properties are set to allow override format on check in. Used as Idoc Script and as a configuration variable.

- Conditional variable.
- The System Properties utility can be used to enable this option. Changes you make to System Properties appear in the configuration files
- When set to TRUE, enables the ability to override the formats on check in.
- Default is FALSE.

Returns a Boolean value:

- Returns TRUE if override format is allowed on check in.
- Returns FALSE if no override format is in use.

Example

As configuration setting, enables override format:

```
IsOverrideFormat=true
```

As script, checks override format:

```
<$IsOverrideFormat$>
```

IsPhysicallySplitDir

Description

Used to flag the server that the Vault and Weblayout directories are on different file systems.

- Flags the server that the Vault and Weblayout directories are on different file systems.
- When set to TRUE, flags the server concerning the split directory system.
- Default is FALSE.

See Also: WeblayoutDir (5-204)

Example

Used as configuration entry:

```
IsPhysicallySplitDir=true
```

JdbcConnectionString

Description

Defines the Java Database Connectivity connection including the hostname, port number, and instance name.

The JdbcConnectionString must be manually changed if the database is changed or if the system could not find the database during the installation. The database is automatically configured during the installation if there are no network errors that make it impossible for the system to connect to the database. If the runtime version of Microsoft Access is used, there are no database configuration options to set. The Jdbc Connection String can also be defined with the System Properties utility.

❖ Used on install. May be reconfigured by the developer.

See Also: JdbcDriver (5-154)

Example

Defines the Jdbc connection path:

```
JdbcConnectionString=jdbc:oracle:thin:@hostname:port_number
:instance_name
```

JdbcDriver

Description

Defines the Java Database Connectivity device driver name.

The JdbcDriver must be manually changed if the database is changed or if the system could not find the database during the installation. The database is automatically configured during the installation if there are no network errors that make it impossible for the system to connect to the database. If the runtime version of Microsoft Access is used, there are no database configuration options to set.

This entry is set on install but may be reconfigured by the developer. The Jdbc Driver Name can also be defined with the System Properties utility.

❖ There is no default entry.

See Also: IsJdbc (5-147)

JdbcConnectionString (5-153)

Example

Defines the Jdbc device driver:

```
JdbcDriver=oracle.jdbc.driver.OracleDriver
```

JdbcPassword

Description

Defines the SQL Server database password.

- User enters a password upon install. This entry can be updated using the System Properties stand-alone application on the content server or upon reinstallation of the content server.
- There is no default value.

See Also: JdbcUser (5-157)

Example

Used as configuration entry:

`JdbcPassword=`*`password`*

JdbcPasswordEncoding

Description

Defines the Java Database Connectivity password encoding type.

This entry indicates the type of password encoding to use when storing the password. The default is to use the Secure Hash Algorithm update 1 (SHA1). If any other nonempty value is assigned, then no encoding is performed. The suggested value to use if you want to have open text passwords is *OpenText*.

❖ Default is SHA1-CB.

Example

Used as configuration entry:

```
JdbcPasswordEncoding= SHA1-CB
```

JdbcUser

Description

Defines the Java Database Connectivity user name.

- A user name should be entered during install. This entry can be updated using the System Properties stand-alone application.
- Default is *sa* if a user name is not entered during install.

See Also: JdbcPassword (5-155)

Example

Used as configuration entry:

```
JdbcUser=sa
```

JspAdminQuery

Description

Defines the files to be made available as web application files. Generally used to define the query to find the web application archiver (.war file) when deploying the Stellent JavaBean.

See Also: IsJspServerEnabled (5-150)

Example

Used as configuration entry:

```
JspAdminQuery=dExtension <matches> war
```

JspDefaultIndexPage

Description

Defines the default pages for a web application.

- If using a web application file, defines which page is the default page of the application. A comma-separated list is used to define the search sequence.
- Default is index.html, index.htm, index.jsp

See Also: IsJspServerEnabled (5-150)

Example

Used as configuration entry:

```
JspDefaultIndexPage=index.html,index.htm,index.jsp
```

JspEnabledGroups

Description

Defines the security groups to be enabled for Java Server Page functionality.

Used to define a comma-separated list of security groups that can host Java Server Pages. Since Java Server Pages usually have full privileges to any resource on the hosting machine, it can be important to restrict Java Server Pages to security groups which allow only privileged contributors.

See Also: IsJspServerEnabled (5-150)

Example

Used as configuration entry:

```
JspEnabledGroups=jsp,group1
```

MacSupportsSignedApplets

Description

Relates to the acceptance of signed applets for Macintosh systems.

- Used only with Macintosh installations.
- If set to FALSE signed applets are not accepted.
- Default is TRUE if the system is evaluated to support signed applets.

Example

Used as configuration entry:

```
MacSupportsSignedApplets=true
```

MailServer

Description

Defines the e-mail server.

- This is an e-mail configuration entry used with Workflow notification.
- An e-mail server should be defined during installation and can be updated on reinstallation of the content server and Inbound Refinery.
- There is no default entry.

Example

Used as configuration entry:

```
MailServer=mail.company.com
```

MajorRevSeq

Description

Defines the Major Revision Label Sequence.

- This variable is set with the System Properties Utility and is included in the configuration file when the system is initialized.
- Returns the major revision label sequence (returns the value of the configuration setting).
- Returns a string.

The DocInfo field named Revision has a default revision label of the number sequence 1, 2, 3, 4, 5, and so forth. This number increments automatically for each revision of a content item. You can override the Revision default by changing the definition of the revision label.

The revision label consists of two parts: a major and minor revision sequence. The Major Revision Label Sequence is the first number or letter. For example, in the revision sequence 1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b, 3c, the numbers 1, 2, 3 are the major revision sequence.

The major revision sequence is defined as a range of numbers or letters. The major revision sequence may have multiple ranges.

The following are the restrictions on defining the range: Only numbers or letters may be used but not both. For example, 1-10 is a valid range but A-10 is not a valid range. Letter ranges may only have one letter. For example, A-Z is a valid range but AA-ZZ is not a valid range.

These are examples of different revision sequences and how you would define the major and minor revision entries in the *config.cfg* file:

Example 1: The revision sequence is A, B, C, D, 1, 2, 3, 4, and so forth.
MajorRevSeq=A-D,1-99.

Example 2: The revision sequence is 1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b, 3c, and so forth.

See Also: MinorRevSeq (5-173)

Example

As configuration setting, defines the major revision sequence:

```
MajorRevSeq=1-99
```

As script, returns the value of the configuration setting:

```
<$MajorRevSeq$>
```

MaxArchiveErrorsAllowed

Description

Defines the number of errors that the Archiver will allow.

- This entry defines the number of errors that the Archiver will allow on either an export or import before it aborts the archive action.
- This entry must be manually edited.
- Default is 50.

Example

Used as configuration entry:

```
MaxArchiveErrorsAllowed=50
```

MaxCollectionSize

Description

Defines the total number of files passed to Indexer in one batch.

- Used for loading content items into Indexer. Edited in the Repository Manager on the Indexer tab.
- Returns the defined maximum collection size (returns value from configuration files).
- Default is 25.

Example

As configuration setting, defines batch size:

```
MaxCollectionSize=25
```

As script, returns value from configuration files:

```
<$MaxCollectionSize$>
```

MaxDocIndexErrors

Description

Defines the number of Indexer errors allowed.

- This entry must be manually edited.
- The background indexing session is terminated if the number of Indexer errors that are allowed are exceeded. For example, if during a rebuild the indexing engine discovers that the file it is about to index no longer exists, it reports an error and increments an internal error counter.

Important: If the counter becomes greater than this configuration value, the rebuild terminates.

- Default is 50.

Example

Used as configuration entry:

```
MaxDocIndexErrors=50
```

MaxQueryRows

Description

Sets the maximum number of search results rows.

This value is used as a presentation option. Increasing the maximum rows will slow response time. The default can be overridden by setting the major and minor revision sequences. This option affects the number of rows that are displayed on the Repository Manager, the Active Reports, and the Work In Progress page.

- If no value is set, the default value is used.
- Default is 200.

See Also: MajorRevSeq (5-163)

MinorRevSeq (5-173)

Example

Used as configuration entry:

```
MaxQueryRows=50
```

MaxResults

Description

Defines the number of returned content items on a search result.

This value is used as a presentation option. Value can be set to increase or decrease the number of returned content items on a search result. Entry must be manually edited.

- Changing this entry overrides the programmed default.
- Default is 200.

Example

Used as configuration entry:

```
MaxResults=200
```

MaxSearchConnections

Description

Defines the maximum number of search connections in the system.

- The maximum number of search connections in the system must be defined during installation.
- There is no default entry.

Example

Used as configuration entry:

```
MaxSearchConnections=##
```

MemoFieldSize

Description

The field size created in the database for memo fields.

- The field size created in the database for memo fields. Ensure that the database supports whatever size is chosen.
- Default is 255.

See Also: MinMemoFieldSize (5-172)

Example

Used as configuration entry:

```
MemoFieldSize=255
```

MinMemoFieldSize

Description

Defines the minimum memo field size.

Defines the cutoff for the size of a custom metadata field when it changes from a long text field to a memo field. The content server supports a database administrator (DBA) editing the field sizes of the DocMeta table directly. The content server then classifies the *varchar* field types by their lengths.

❖ Default is 255.

These are the field type rules:

Field Types	Evaluation Rules
Text	<50
Long Text	>= 50 and < MinMemoFieldSize
Memo	>= MinMemoFieldSize

See Also: MemoFieldSize (5-171)

Example

Used as configuration entry:

```
MinMemoFieldSize=255
```

MinorRevSeq

Description

Defines the Minor Revision Label Sequence

This variable is set with the System Properties Utility and is included in the configuration file when the system is initialized.

- Returns the major revision label sequence (returns the value of the configuration setting).
- Returns a string.

The DocInfo field named Revision has a default revision label of the number sequence 1, 2, 3, 4, 5, and so forth. This number increments automatically for each revision of a content item. You can override the Revision default by changing the definition of the revision label.

The revision label consists of two parts: a major and minor revision sequence. The Minor Revision Label Sequence follows the Major Revision Label Sequence. For example, in the revision sequence 1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b, 3c, the letters a, b, c are the minor revision sequence.

The and minor revision sequences is defined as a range of numbers or letters. The minor revision sequence may only have one range.

The following are the restrictions on defining the range: Only numbers or letters may be used but not both. For example, 1-10 is a valid range but A-10 is not a valid range. Letter ranges may only have one letter. For example, A-Z is a valid range but AA-ZZ is not a valid range.

The following are examples of different revision sequences and how you would define the major and minor revision entries in the *config.cfg* file.

Example 1: The revision sequence is A, B, C, D, 1, 2, 3, 4, and so forth. MajorRevSeq=A-D,1-99.

Example 2: The revision sequence is 1a, 1b, 1c, 2a, 2b, 2c, 3a, 3b, 3c, and so forth.

See Also: MajorRevSeq (5-163)

Example

As configuration setting, defines the minor revision label sequence:

```
MinorRevSeq=a-c
```

As script, returns the value of the configuration setting:

```
<$MinorRevSeq$>
```

MultiUpload

Description

Permits the upload up multiple files.

- When set to TRUE, multiple files can be zipped as a bundle and uploaded.
- There is no default entry.

Returns a Boolean value:

- Returns TRUE if enabled (returns value of configuration files).
- Returns FALSE if not enabled.

See Also: DownloadApplet (5-123)

UploadApplet (5-194)

Example

As configuration setting, permits the upload of multiple files:

```
MultiUpload=true
```

As script, returns the value of the configuration setting:

```
<$MultiUpload$>
```

NetworkAdminGroup

Description

References the Domain Admins NT group.

- The content server maps the Domain Admins NT group in the default master domain to the content server role admin. By setting this value, a different NT group can be mapped to the admin role. Set during installation of the content server and Inbound Refinery.
- This entry must be manually edited.

Example

Used as configuration entry:

```
NetworkAdminGroupt=admin
```

NoAutomation

Description

Enables or disables automation activity.

- When enabled, the content server will perform no automation activity of any type.This setting overrides all other related configuration other settings.
- This entry must be manually edited.
- Default is TRUE.

Example

Used as configuration entry:

```
NoAutomation=true
```

SearchCacheTrace

Description

Enables verbose output to the View Server Output window.

- To set verbose output to cache, enter this variable in the primary instance config.cfg file and assign the value to TRUE. Verbose output (the most detail) will display to the View Server Output window when enabled.
- When set to TRUE, verbose output is enabled.
- There is no default entry.

Example

Used as configuration entry:

```
SearchCacheTrace=true
```

SearchConnectionWaitTimeout

Description

Defines the search connection wait timeout.

- Sets the search connection wait timeout in seconds. This entry must be manually edited.
- There is no default entry.

Example

Used as configuration entry:

```
SearchConnectionWaitTimeout=600
```

SearchDebugLevel

Description

Defines the search debug level.

The entry defines how detailed the indexing messages are in the log files. The log files are located at *\weblayout\groups\secure\logs\verity*. The first two options are designed for the System Administration and the remaining two options are designed for a developer debugging a problem with the search engine. This entry can be updated from the Repository Manager.

These flags in the configuration file change the content of this variable:

Flag	Description
all	A full report including debug, trace, and verbose information is generated.
debug	Additional information at the functional level is generated. This flag is intended for a developer debugging a problem with the search engine.
none	No log is generated.
trace	Information is logged as each activity is performed. This flag is intended for a developer debugging a problem with the search engine.
verbose	The Inbound Refinery log shows the related configuration information. For example, whether the PDF is optimized, where the custom converter path is located, and how much time is allowed for custom conversions.

- Default is ALL.

Example

Used as configuration entry:

```
SearchDebugLevel=all
```

SearchDir

Description

Defines the path to the Search directory.

- Path must be defined during installation. This entry can be updated upon reinstallation of the content server or Inbound Refinery.
- There is no default entry.

Example

Used as configuration entry:

```
SearchDir=c:/stellent/search/
```

SharedDir

Description

Defines the path to the shared directory.

Actions such as Replication require a shared directory between the importing and exporting instances. For example, you can use replication to automatically export from one instance and import to another instance, synchronizing the second site with the initial instance. This path is defined during installation. This entry can be updated upon reinstallation of the content server or Inbound Refinery.

- There is no default entry.

Example

Used as configuration entry:

```
SharedDir=c:/stellent/shared/
```

SmtpPort

Description

Defines the Simple Mail Transfer Protocol (SMTP) port number.

- The SMTP port should be defined during installation. This entry can be updated on reinstallation of the content server and Inbound Refinery.
- There is no default entry.

Example

Used as configuration entry:

```
SmtpPort=25
```

StrConfineOverflowChars

Description

Defines a padding character.

- Defines the character used for padding by the function strConfine.
- Default entry is a dot.

Example

Used as configuration entry:

```
StrConfineOverflowChars=.
```

SystemDateFormat

Description

Overrides the default date/time format used by the content server.

This configuration entry overrides the default date/time format used by the content server. If a client is using a different locale then the system locale, then their date format will be the date format for that locale, not the system locale. The syntax for this format is an extension of the date format functionality provided in the standard Java class libraries. For example, the default US format for the content server is:

```
M/d{/yy} {h:mm[:ss] {aa}[zzz]}
```

If a letter is repeated more then once (such as mm), then that indicates that the number will be zero filled until it fills out the number of repeats (08 instead of 8). The letter H formats using a 24-hour clock while the letter h formats using the 12-hour clock. The letter z is used to indicate the time zone (three z's indicates that the value is non numerical). The usage of other letters should be deducible from their context. A left and right brace pair ({…}) indicates a part of the format that will always be part of the output for a format but does not have to be present for the date to be successfully parsed. A left and right bracket pair ([…]) indicates an element that will not be part of the output for a format, but will be successfully parsed if present. For example, the above date will format as 10/11/01 5:03 PM but it can successfully parse any of

```
10/11
10/11/01 5:03:33 PM
10/11/2001 17:03:33 America/New_York
```

If the meridian symbol is missing (AM/PM in the US) then the hour is parsed using the 24-hour clock (even if h is used instead of H in the format string). The content server can parse both two digit and four digit years.

Important: If SystemLocale is not specified as a configuration entry, 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 format from the content server configuration tables for that locale will be used instead.

- Default US format: M/d{/yy} {h:mm[:ss] {aa}[zzz]}

See Also: SystemLocale (5-188)

SystemTimeZone (5-190)

Example

Used as configuration entry to set the standard US date time using the four digit year format:

```
SystemDateFormat=M/d{/yyyy} {h:mm[:ss] {aa}[zzz]}
```

SystemLocale

Description

Sets the default system locale for the content server.

This configuration value sets the default system locale for the content server. The value can control the strings used, the character encoding for the page, the date format, and the parameters used to build the full text indexing. Any of these individual configurations can be controlled by separate configuration entries or by changing the settings for the SystemLocale. The defaults for a particular locale can be changed by using the System Properties application and using the Localization tab.

Content server settings are available for English-US, English-UK, Français, and Deutsch. Of these only the English-US formats the date with the month leading (1/31/01 instead of 31/1/01).

The configuration entries VerityLocale and VerityEncoding will still override settings from the SystemLocale, but configuring the SystemLocale directly is the preferred solution since there are many choices for VerityLocale and VerityEncoding that are not compatible.

Important: If SystemLocale is not specified as a configuration entry, 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 format from the content server configuration tables for that locale will be used instead.

See Also: SystemDateFormat (5-186)

SystemTimeZone (5-190)

Example

Used as configuration entry:

```
SystemLocale=true
```

SystemTimeZone

Description

- Overrides the default time zone for SystemLocale.
- The table SystemTimeZones in the *std_locale.htm* file lists all the time zones usable with the content server.

See Also: SystemDateFormat (5-186)

SystemLocale (5-188)

Example

Used as configuration entry:

```
SystemTimeZone=STZ:0.0,2,-1,1,1.0,9,-1,1,1.0
```

TraceResourceConflict

Description

When the content server is started from a command line, lists each system resource that is overridden twice by component resources.

- When set to 1, system resources that are overridden by two or more component resources are output to the command line.
- Default is 0.

See Also: *TraceResourceLoad (page 5-192)*

TraceResourceOverride (page 5-193)

Example

```
TraceResourceConflict=1
```

TraceResourceLoad

Description

When the content server is started from a command line, lists all resource loading activities.

- When set to 1, all resources loaded, resource overrides, resource conflicts, and resource merges are output to the command line.
- Default is 0.

See Also: *TraceResourceConflict (page 5-191)*

TraceResourceOverride (page 5-193)

Example

```
TraceResourceLoad=1
```

TraceResourceOverride

Description

When the content server is started from a command line, lists system resources and non-system component resources that are overridden by a component resource.

- When set to 1, system resources and non-system resources that are overridden by one or more component resources are output to the command line.
- Default is 0.

See Also: *TraceResourceConflict (page 5-191)*

TraceResourceLoad (page 5-192)

Example

```
TraceResourceOverride=1
```

UploadApplet

Description

Evaluates whether the Upload applet is enabled. Used as Idoc Script and configuration variable.

- Conditional variable. This should not be enabled when MultiUpload is enabled. For use by legacy products—not intended for regular use.
- Retrieves the value of the configuration setting. Used as configuration setting to enable or disable the Upload applet.
- Used as configuration setting to enable the Upload applet.

Returns a Boolean value:

- Returns TRUE if the Upload applet has been enabled in the System Properties utility.
- Returns FALSE if the Upload applet has not been enabled.

See Also: MultiUpload (5-175)

Example

As configuration setting, enables the Upload applet:

```
UploadApplet=true
```

As script, evaluates condition of Upload applet:

```
<$UploadApplet$>
```

UseBellevueLook

Description

Enables alternate set of navigation buttons on interface.

- When set to TRUE, the “Bellevue” look is used.
- Default is FALSE

See Also: UseStellentLook (5-197)

UseXpedioLook (5-198)

Example

Used as configuration entry:

```
UseBellevueLook=false
```

UseFourDigitYear

Description

Obsolete—use SystemDateFormat. Used in previous versions to define a four-digit or two-digit year format.

- The configuration entry UseFourDigitYear is obsolete. The standard content server install as of 5.1 and greater does not use the four digit year as the default.
- When set to TRUE, a four-digit year format is used.
- When set to FALSE, a two-digit year format is used.
- This entry defaulted to TRUE in previous versions.
- This entry must be manually edited.

Note: The configuration entry SystemDateFormat overrides the default date/time format used by the content server.

See Also: SystemDateFormat (5-186)

Example

Used as configuration entry in previous versions:

```
UseFourDigitYear=true
```

UseStellentLook

Description

Enables the Stellent set of navigation interface buttons. This is the default interface.

- To enable an alternate set of navigation buttons on the interface, set this entry to FALSE, then define and enable the alternate set.
- When set to FALSE, an alternate look can be set.
- Default is TRUE

See Also: UseBellevueLook (5-195)

UseXpedioLook (5-198)

Example

Used as configuration entry:

```
UseStellentLook=true
```

UseXpedioLook

Description

Enables alternate set of navigation buttons on interface.

- When set to TRUE, the “Xpedio” look is used.
- Default is FALSE

See Also: UseBellevueLook (5-195)

UseStellentLook (5-197)

Example

Used as configuration entry:

```
UseXpedioLook=false
```

VaultDir

Description

Defines the path to the vault directory.

- This is the root directory of the content server native file repository. This entry is only required if the vault directory is not located in the content server root directory. This path is user defined during installation. Directory path can be updated upon reinstallation of the content server.
- There is no default entry.

Example

Used as configuration entry:

```
VaultDir=f:/vault
```

VerityEncoding

Description

Defines the Verity UTF8 file encoded language selection.

Verity specific variable. Allows the use of UTF8 file encoding for all languages. Set during installation of the content server and Inbound Refinery.

For example, the EUC_KR (Korean) and GBK (Simplified Chinese) UTF8 file encoding can be set by specifying EUC_KR or GBK as the value for VerityEncoding in the config.cfg configuration file.

- There is no default entry.
- This entry must be manually edited.

Example

Used as configuration entry:

```
VerityEncoding=EUC_KR
```

VerityInstallDir

Description

Defines the Verity installation directory.

Verity specific variable. Set by default during installation of the content server and Inbound Refinery.

- NOT FOR USER EDIT.

Example

Used as configuration entry:

```
VerityInstallDir=c:/search97/common
```

VerityLocale

Description

Defines the Verity local language selection.

- Verity specific variable. Set during installation of the content server and Inbound Refinery.
- Default is ENGLISH.

Example

Used as configuration entry:

```
VerityLocale=english
```

WebBrowserPath

Description

Defines the path to the web browser that displays the Stellent Help files in stand-alone applications.

- This path is user defined during installation. Can be set using the System Properties stand-alone application on the content server.
- There is no default entry.

Example

Used as configuration entry:

```
WebBrowserPath=c:/Software/Microsoft/Iexplore.exe
```

WeblayoutDir

Description

Defines the path to the directory named *weblayout*. This is the root directory of the content server web site.

- This entry is only required if the *weblayout* directory is not located in the content server root directory. This path is user defined during installation. Directory path can be updated upon reinstallation of the content server.
- There is no default entry.

Example

Used as configuration entry:

```
WeblayoutDir=f:/weblayout
```

WebProxyAdminServer

Description

Defines whether a web proxy administrative server is used.

- When set to TRUE, a proxy administrative server is used. Set during installation and can be updated upon reinstallation of the content server.

Returns a Boolean value:

- Returns TRUE if enabled for a proxy administrative server.
- Returns FALSE if a proxy administrative server is not used.

Example

Used as configuration entry:

```
WebProxyAdminServer=true
```

Chapter

6

WEB SERVER VARIABLES

OVERVIEW

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 as well as environment variables. These environment variables can be used to output information to a log file or used within Idoc Script statements and as part of evaluations.

This Idoc Script statement evaluates whether the remote host address (returned as a string) matches a defined string:

```
<$if strEquals(“207.0.0.1”,REMOTE_HOST)$>
```

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

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

VARIABLE DESCRIPTIONS

CONTENT_LENGTH

Description

Returns the length of the requested content.

- Provides the length of the requested content as supplied by the client. This environment variable is specific to the current gateway program request.
- Returns a string.

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-->
```

GATEWAY_INTERFACE

Description

Returns the CGI specification revision level.

- Provides the revision level of the CGI specification to which this server complies.This environment variable is not request-specific and is set for all requests:
- Returns a string.
- Uses 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-->
```

HTTP_COOKIE

Description

Returns the cookie HTTP request header.

- Provides the name/value pair of the matching cookie in the HTTP request.
- Returns a string.
- Uses 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

Description

Returns the Web server name.

- Provides the name of the Web server.
- Returns 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_REFERER

Description

Returns the URL of the referenced directory.

- Provides the complete URL of the referenced directory on the local server.
- Returns a string.

Example

As information output on a page or to a log:

```
HTTP_REFERER=http://centralserver/stellent/
```

As part of an Idoc Script statement or evaluation:

```
<$if HTTP_REFERER$>
```

```
<!—statement-->
```

HTTP_USER_AGENT

Description

Returns browser information.

- Provides the browser type, version number, library, and platform the browser is configured for. This references the browser that the client is using to send the request.
- Returns a string.
- Uses this 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-->
```

PATH_INFO

Description

Returns virtual path information from the client.

- Provides additional path information from the client. When the virtual path is returned, any additional information at the end of this path is also returned. The additional information is sent as PATH_INFO.

- Returns a string.
- Use PATH_TRANSLATED to display this information on a page or to a log.
- This environment variable is specific to the current gateway program request.

Example

As part of an Idoc Script statement or evaluation:

```
<$if PATH_INFO$>
<!—statement-->
```

QUERY_STRING

Description

Returns the query information.

- Provides the information that follows the ? delimiter in the URL. This environment variable is specific to the current gateway program request.
- Returns a string.

Example

As information output on a page or to a log:

```
QUERY_STRING=IdcService=GET_DOC_PAGE&Action=GetTemplateP
age&Page=STD_QUERY_PAGE
```

As part of an Idoc Script statement or evaluation:

```
<$if QUERY_STRING$>
<!—statement-->
```

REMOTE_ADDR

Description

Returns the IP address of the remote host.

- The IP address of the remote host making the request. This environment variable is specific to the current gateway program request.
- Returns a string.

See Also: REMOTE_HOST (6-7)

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-->
```

REMOTE_HOST

Description

Returns the name of the remote host.

- Provides the hostname making the request. If the hostname is unknown to the server REMOTE_ADDR is returned. This environment variable is specific to the current gateway program request.
- Returns a string.

See Also: REMOTE_ADDR (6-7)

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-->
```

REQUEST_METHOD

Description

Returns the method that the request was made.

- Provides the request access method used. This environment variable is specific to the current gateway program request.
- Returns 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-->
```

SCRIPT_NAME

Description

Returns the path to the CGI linking file.

- Provides the relative path to the nph-idc_cgi.exe file for Solaris.
- Returns a string.

Example

As information output on a page or to a log:

```
SCRIPT_NAME=/intradoc-cgi/iis_idc_cgi.dll
```

As part of an Idoc Script statement or evaluation:

```
<$if SCRIPT_NAME$>
<!—statement-->
```

SERVER_NAME

Description

Returns the hostname, DNS alias, or IP address.

- Provides the hostname, DNS alias, or IP address of the server as it would appear in a self-referencing URL. This environment variable is not request-specific and is set for all requests:
- Returns 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

Description

Returns the server port number.

- Provides the port number to which the request was sent. This environment variable is specific to the current gateway program request.
- Returns 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

Description

Returns the name and revision protocol of the server.

- Provides the information protocol name and revision level of the incoming request. This environment variable is specific to the current gateway program request.
- Returns a string.
- Uses 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

Description

Returns the name and version of the server software.

- Provides the name and version of the information server software answering the request. This is also the server running the gateway. This environment variable is not request-specific and is set for all requests.
- Returns a string.
- Uses 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-->
```

Index

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X