



CMS Encore Pro Developers Guide (Incorporating the eScript Function Reference)



CMS ENCORE PRO

The Advanced Desktop Content Management System

CMS Encore Pro Developers Guide

*Author: BIIT Software Systems
Document Version 1.0
28 February 2005
CMS Encore Pro 1.0 (2005-02-28)
BIIT Software Systems*

Introduction:	8
Section A: The Developers Guide To Encore:	9
1. Integrated Development Environment (IDE):	9
1.1 Syntax Highlighting:	9
1.2 Code Completion:	10
1.2.1 Completing Function Names:.....	10
1.2.2 Getting a List of Parameters:.....	13
1.3 Find and Replace:	15
1.4 Incremental Search:	16
1.5 Indent and Un-indent of Code:	16
1.6 Bracket Matching:	16
1.7 Setting IDE Editor Properties:	17
2. Introduction to Encore Scripting Language (eScript).	18
2.1 What is eScript?	18
2.2 How to use eScript?	18
3. eScript Language Constructs.	19
3.1 Operators and Operands:	19
3.1.1 Relational Operators:.....	19
3.1.2 Boolean Operators:.....	20
3.1.3 Arithmetic Operators:.....	20
3.1.4 String Operators:.....	20
3.2 Assignment Operarators:	21
3.3 Datatypes:	21
3.3.1 Boolean:.....	21
3.3.2 DateTime:.....	22
3.3.3 Float:.....	22
3.3.4 Integer:.....	22
3.3.5 String:.....	23
3.3.6 Variant:.....	23
3.3.7 Built in Classes:.....	23
3.4 Constants:	24
3.5 Variables:	24
3.5.1 Declaring Variables:.....	24
3.6 eScript Reserved Words:	25
3.7 If Statements:	25
3.9 Control Loops:	27
3.9.1 Repeat Statements:.....	27
3.9.2 While Statements:.....	28
3.9.3 For Statements:.....	28
4. eScript Functional Elements:	29
4.1 eScript Functions and Code:	29
4.2 FOREACH Statements:	30
4.2.1 Using a <FOREACH> to simulate a "between" statement:.....	33
4.3 SQL Structures:	34
4.4 ANCHOR Tags:	35
4.5 ESP <esp> Tags:	37
4.6 Macros:	39
4.6.1 Global/Project Properties:.....	39
4.6.2 Global Properties (Internal):.....	40
4.6.3 Local Properties:.....	41

4.6.4: Images/Media:.....	42
4.6.5: Content/Channel Macros:.....	43
4.6.6 Query Field Macros:.....	44
4.6.7 Using Macro Suffixes to format text:.....	44
Section B: The eScript Function Reference.....	45
1. Channel(Folder) Related Functions and Procedures:.....	45
GetAbsChannelLink function.....	45
GetChannelIdByName function.....	45
GetChannelIdByPath function.....	46
GetChannelIdByNamePath function.....	47
GetChannelIdByShortName function.....	47
GetChannelImageSrc function.....	48
GetChannelImageSrc2 function.....	48
GetChannelImageSrc2ById function.....	49
GetChannelImageSrcById function.....	49
GetChannelLink function.....	50
GetChannelListStr function.....	51
GetChannelMTitle function.....	52
GetChannelName function.....	52
GetCurrChannelId function.....	53
GetCurrChannelLevel function.....	53
GetCurrChannelName function.....	53
GetCurrChannelParentId function.....	54
GetFullChannelPagePath function.....	54
GetFullChannelPath function.....	54
GetStartChannelId function.....	55
GetHomePath function.....	55
GetMainParentChannelId function.....	56
GetMainParentChannelName function.....	57
2. Content (Article) Related Functions/Procedures:.....	58
GetAbsContentLink function.....	58
GetContent function.....	58
GetContentEx function.....	58
GetContentByTitle function.....	59
GetContentByTitleEx function.....	59
GetContentIdByGlobalRefName function.....	59
GetContentImageSrc function.....	60
GetContentImageSrc2 function.....	60
GetContentImageSrc2ById function.....	61
GetContentImageSrcById function.....	61
GetContentImageSrcLink function.....	62
GetContentLink function.....	63
GetContentLinkAdv function.....	63
GetContentPropValue function.....	64
GetContentPropValueEx function.....	64
GetContentSummary function.....	64
GetContentSummaryEx function.....	65
GetContentSummaryByTitle function.....	66
GetContentSummaryByTitleEx function.....	66
GetContentTitle function.....	66
GetCurrContentId function.....	67
3. Template Related Functions/Procedures:.....	68
GetTemplate function.....	68
GetTemplateByName function.....	68
GetTemplateImageSrc function.....	69
GetTemplateImageSrc2 function.....	69
GetTemplateImageSrc2ById function.....	70
GetTemplateImageSrcById function.....	70
GetTemplateImageSrcLink function.....	71
4. Query Related Functions/Procedures:.....	72
QueryChannel function.....	72
QueryChannelList function.....	72
QueryChannelListAdv function.....	73
QueryChannelListAdv2 function.....	74
QueryChannelListDesc function.....	75
QueryChannelListNext function.....	76
QueryChannelListPrev function.....	76
QueryChannelMediaList function.....	77

QueryContent function.....	77
QueryContentList function.....	78
QueryContentListAdv function.....	79
QueryContentListAdv2 function.....	80
QueryContentListDesc function.....	81
QueryContentListIncl function.....	82
QueryContentListInclDesc function.....	82
QueryContentListInclNext function.....	83
QueryContentListInclPrev function.....	84
QueryContentListLastModified function.....	85
QueryContentListLastModifiedGlobal function.....	85
QueryContentListLatest function.....	86
QueryContentListLatestGlobal function.....	86
QueryContentListNext function.....	87
QueryContentListPrev function.....	88
QueryContentListSorted function.....	89
QueryContentListSortedDesc function.....	90
QueryContentListSortedNext function.....	91
QueryContentListSortedPrev function.....	92
QueryContentMediaList function.....	92
QueryFieldAsDateTime function.....	93
QueryFieldAsFloat function.....	93
QueryFieldAsInteger function.....	94
QueryFieldAsString function.....	94
QueryFieldByName function.....	95
QueryFieldIsNull function.....	95
QueryMediaList function.....	96
QueryIsStartChannel function.....	96
QueryIsParentChannel function.....	97
QueryIsCurrChannel function.....	97
QueryIsCurrContent function.....	98
QueryIsCurrMedia function.....	98
QueryMedia function.....	99
QueryParamAsInteger function.....	99
QueryParamAsFloat function.....	100
QueryParamAsString function.....	100
QueryParamAsDateTime function.....	100
QueryParamByName function.....	101
QueryParamClear function.....	101
QueryOpen function.....	101
QueryNext function.....	101
QueryFirst function.....	102
QueryBOF function.....	102
QueryEOF function.....	102
QueryClose function.....	103
QueryProjectMediaList function.....	103
QueryRecordCount function.....	104
QueryRelatedContentList function.....	105
5. Date/Time Related Functions/Procedures:.....	106
Date function.....	106
DateTimeToStr function.....	106
DateToStr function.....	106
DayOfWeek function.....	107
DecodeDate procedure.....	107
DecodeTime procedure.....	108
EncodeDate function.....	109
EncodeTime function.....	110
FormatDateTime function.....	111
IsLeapYear function.....	112
Now function.....	112
StrToDateTime function.....	113
Time function.....	113
TimeToStr function.....	113
6. String Related Functions and Procedures:.....	114
AnsiLowerCase function.....	114
AnsiQuotedStr function.....	114
AnsiUpperCase function.....	115
Chr function.....	115

ColorToHTMLColorString function.....	115
CompactNewLine function.....	116
Copy function.....	116
Delete procedure.....	117
DoubleQuotedStr function.....	117
ExpandNewLine function.....	118
FloatToStr function.....	118
HTMLToText function.....	118
FormatFloat function.....	119
Insert procedure.....	120
IntToStr function.....	120
LeftStr function.....	121
Length function.....	121
LoadStringFromFile function.....	121
LowerCase function.....	122
MidStr function.....	122
Ord function.....	122
Pos function.....	123
PosEx function.....	123
RemoveHTMLTags function.....	124
RepeatStr function.....	124
ReplaceString function.....	125
RightStr function.....	125
SendLn procedure.....	125
StrToInt function.....	126
StrToIntDef function.....	127
StrToFloat function.....	127
StrToFloatDef function.....	128
TextToHTML function.....	128
Trim function.....	128
TrimLeft function.....	129
TrimRight function.....	129
UpperCase function.....	129
7. System Related Functions/Procedures:.....	130
GetAbsLinkPrefix function.....	130
SetAbsLinkPrefix procedure.....	130
GetAppInfo function.....	131
GetBaseFormatType function.....	131
GetPublishDateTime function.....	132
WhereAmI function.....	132
WhereAmINow function.....	133
GetPropValue function.....	133
GV function.....	134
IsDesigning function.....	134
IsHelpFile function.....	135
IsLiveEdit function.....	135
IsLocalFileOutput function.....	135
IsManual function.....	136
IsPreviewing function.....	136
IsPublishing function.....	136
IsRelativeLocalPaths function.....	137
ReplaceMacros function.....	137
ReplaceMacrosEx function.....	137
ShowMessage procedure.....	138
SiteFilePath function.....	138
SiteFilePath2 function.....	138
CreateOleObject function.....	139
CleanupGlobalVars procedure.....	142
ReadGlobalVar function.....	142
ReadGlobalVarDef function.....	143
WriteGlobalVar procedure.....	143
8. Anchor Related Functions and Procedures:.....	144
GetAnchor function.....	144
AddToAnchor function.....	144
AddUniqueToAnchor function.....	144
AnchorExists function.....	145
ClearAllAnchors procedure.....	145
ClearAnchor function.....	145

SetAnchor function.....	145
9. Logical (Conditional) Related Functions and Procedures:.....	146
IIF function.....	146
10. Glossary/WordScan Related Functions and Procedures:.....	147
GlossaryWordScan function.....	147
GlossaryWordScanAdv function.....	148
11. LiveEdit Related Functions and Procedures:.....	149
LiveEditContent function.....	149
LiveEditProp function.....	150
LiveEditPropT function.....	150
12. Image/Media Functions:.....	151
GetCurrMediaId function.....	151
GetImageSrc function.....	151
GetMediaLink function.....	152
GetAbsMediaLink function.....	152
13. Imaging Template Functions (only valid on Image Templates):.....	153
procedure ImageAssign.....	153
procedure ImageClear.....	153
procedure ImageClearBackground.....	153
procedure ImageEffects.....	153
procedure ImageEllipse.....	153
function ImageGetHeight.....	153
function ImageGetPixel.....	153
function ImageGetTextHeight.....	153
function ImageGetTextWidth.....	153
function ImageGetWidth.....	153
procedure ImageGifAnimateAddFrame.....	153
procedure ImageGifAnimateBegin.....	153
procedure ImageGifAnimateEnd.....	153
procedure ImageGradientBackground.....	153
procedure ImageInit.....	153
function ImageIsAllowed.....	153
procedure ImageLine.....	153
procedure ImageLineTo.....	153
procedure ImageLoadFromFile.....	153
procedure ImageMoveTo.....	153
procedure ImageRectangle.....	153
procedure ImageSaveToFile.....	153
procedure ImageSetFont.....	153
procedure ImageSetHeight.....	153
procedure ImageSetPixel.....	153
procedure ImageSetWidth.....	153
procedure ImageSetWidthHeight.....	153
procedure ImageTextOutXY.....	153
procedure ImageTextOutXYAdv.....	153
14. File - I/O Related Functions/Procedures:.....	156
AppendStringToFile procedure.....	156
ChangeFileExt function.....	156
ChDir procedure.....	156
CreateDir function.....	157
DeleteFile function.....	157
ExtractFileDir function.....	157
ExtractFileDrive function.....	157
ExtractFileExt function.....	158
ExtractFileName function.....	158
ExtractFilePath function.....	158
FileExists function.....	158
FileSearch function.....	159
GetCurrentDir function.....	159
RemoveDir function.....	159
RenameFile function.....	159
SaveStringToFile procedure.....	160
SetCurrentDir function.....	160
15. Mathematical/Statistical Functions/Procedures:.....	161
Abs function.....	161
ArcCos function.....	161
ArcCosh function.....	161
ArcSin function.....	161

ArcSinh function.....	162
ArcTan function.....	162
ArcTanh function.....	162
Cos function.....	162
Cosh function.....	163
Cotan function.....	163
DegToRad function.....	163
Exp function.....	163
Frac function.....	164
Hypot function.....	164
Inc function.....	164
Int function.....	164
IntToHex function.....	165
Ln function.....	165
Log10 function.....	165
Log2 function.....	165
LogN function.....	166
Max function.....	166
Min function.....	166
Pi function.....	166
Power function.....	167
RadToDeg function.....	167
RandG function.....	167
Random function.....	167
RandomInt function.....	168
Randomize procedure.....	168
RandSeed funtion.....	168
Round function.....	168
SetRandSeed procedure.....	169
Sin function.....	169
Sinh function.....	169
Sqr function.....	169
Sqrt function.....	169
Tan function.....	170
Tanh function.....	170
Trunc function.....	170

Introduction:

CMS Encore Pro has a rich set of scripting functions available to the developer. These functions have been categorized into their functional areas and are listed in this document. Where possible we have tried to include examples with the functions as well as a brief explanation of how the function operates.

eScript - which stands for "Encore Script" is a derivative of Delphi (Object Pascal), a very popular programming language. Because it is based on a very rich development language, eScript will be able to handle most of your scripting requirements.

This document consists of two sections. **Section A: "The Developers Guide To Encore"** deals with topics such as the Encore IDE (Integrated Development Environment), the eScript language structure, etc.

Section B: "The eScript Function Reference" displays a categorical list of the eScript functions, with short descriptions and examples.

Section A: The Developers Guide To Encore:

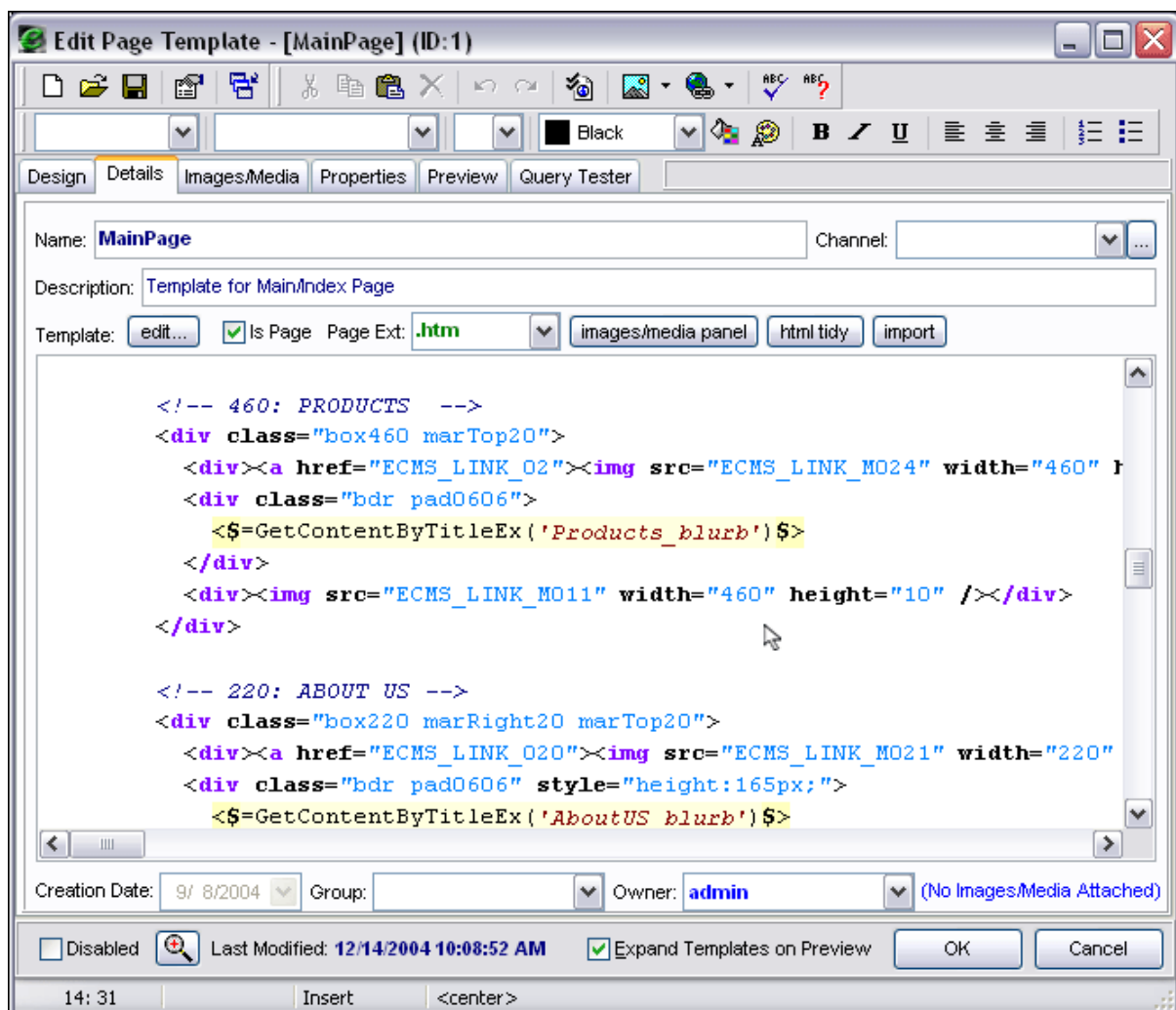
1. Integrated Development Environment (IDE):

The IDE is where you enter the code for your templates and components. It has built in tools and features to make it easier for you to develop your code.

1.1 Syntax Highlighting:

Syntax highlighting is when the IDE highlights parts of your code in different colors, to aid in code readability. Since template code can be made up of various types of code, the IDE highlights the different code types in their own color.

See [Chapter 1.7: Setting IDE Editor Properties](#) to learn how to change syntax highlighting colors.



1.2 Code Completion:

Code completion allows for the IDE to auto complete certain sections of code for you. It can be used to complete function names, list function parameters etc.

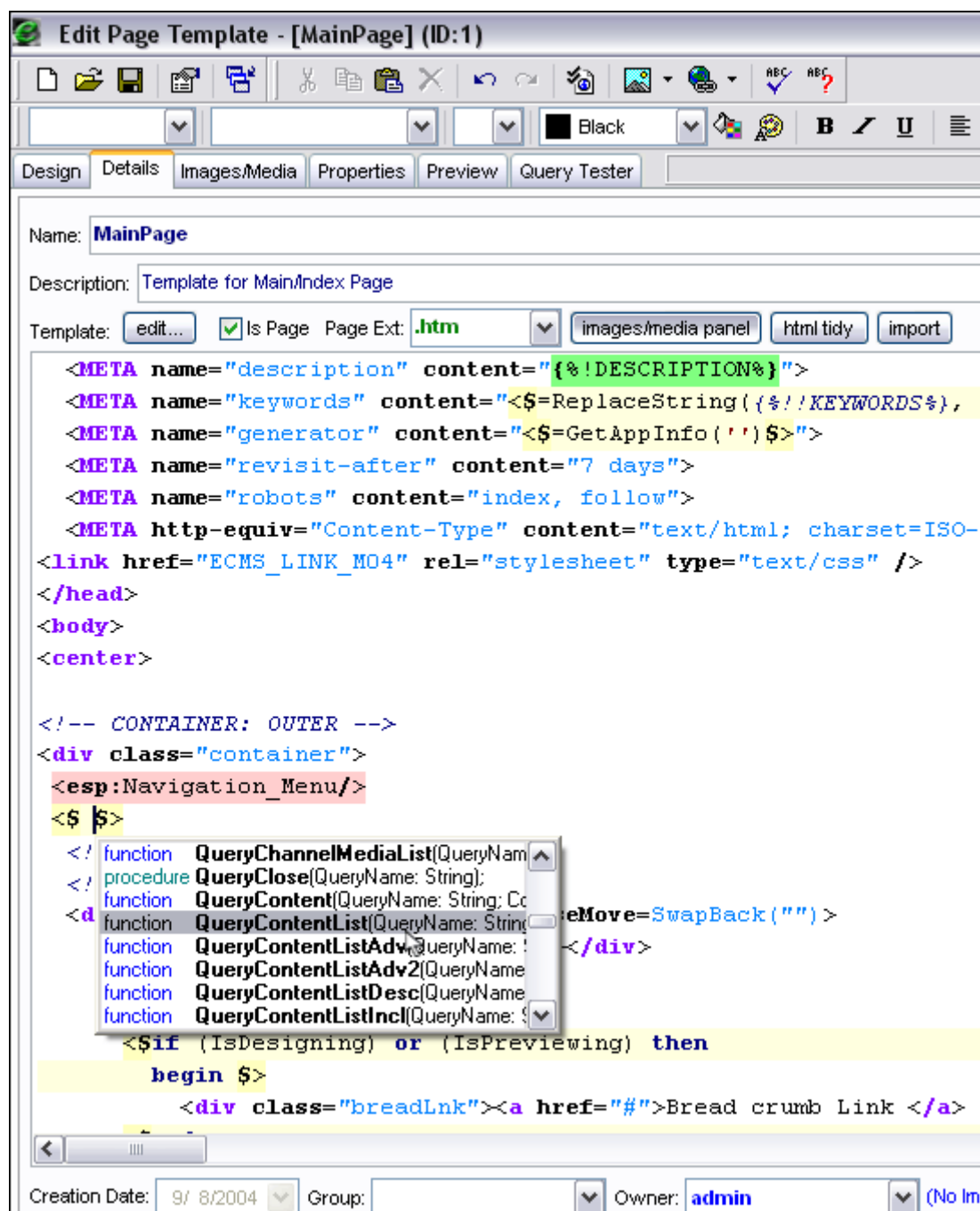
See [Chapter 1.7: Setting IDE Editor Properties](#) to learn how to edit code completion options.

1.2.1 Completing Function Names:

Usage: **Ctrl+Space**

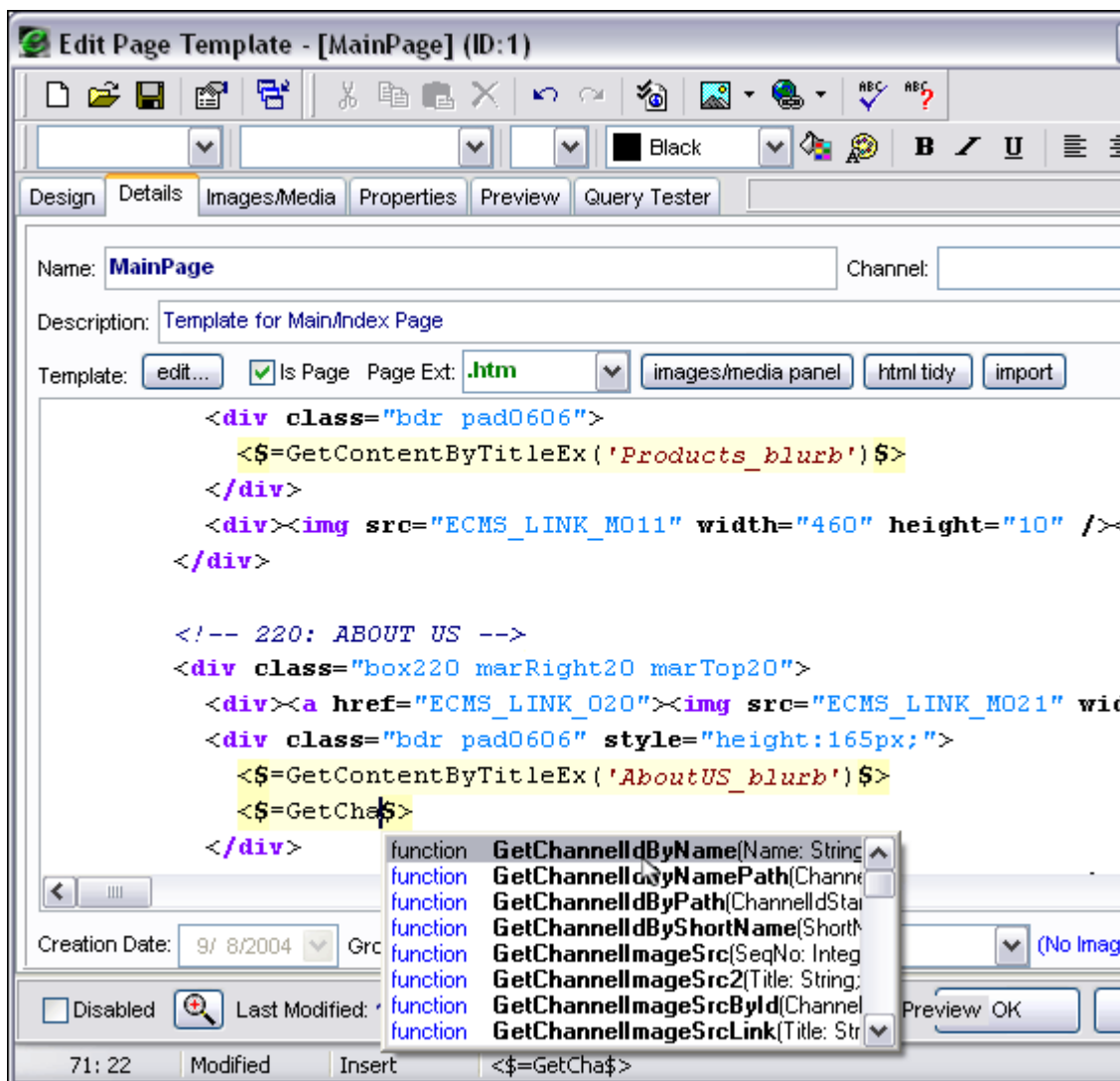
There are literally hundreds of scripting functions available in Encore. Obviously remembering them all can become a huge challenge. Code completion tries to make things simpler by listing all available functions or completing function names when you can only remember part of the function name.

Example 1: Anywhere within eScript tags, if your cursor is on empty text and you press **Ctrl+Space**, Encore will list all of the available eScript functions alphabetically in a floating drop down list. To use one, simply scroll to the function name and press Enter.

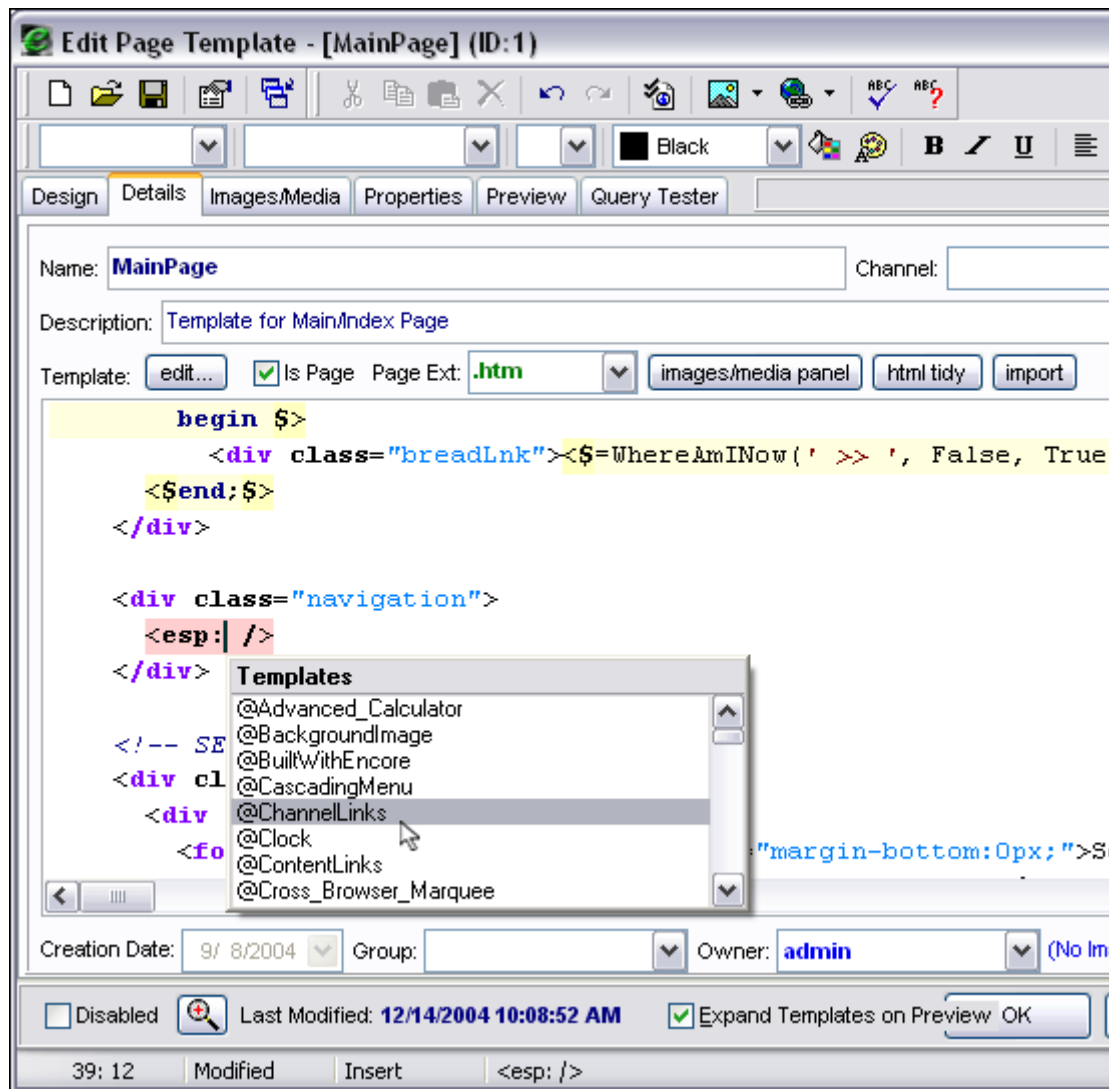


Example 2: You can also use code completion to complete a function name for you. For example, to list all functions starting with the text "Query" you would type the following:

`<${Query}<Ctrl+Space>${}>`, this will then bring up the drop down list of functions, but this time Encore will jump straight to the functions starting with "GetCha".



Example 3: Ctrl+Space can also be used outside of <\$\$> tags. You can use it in <esp> tags to list all <esp> templates. For example type the following <esp:<Ctrl+Space> /> this will then list all the templates or components that can be called in via the <esp> tag

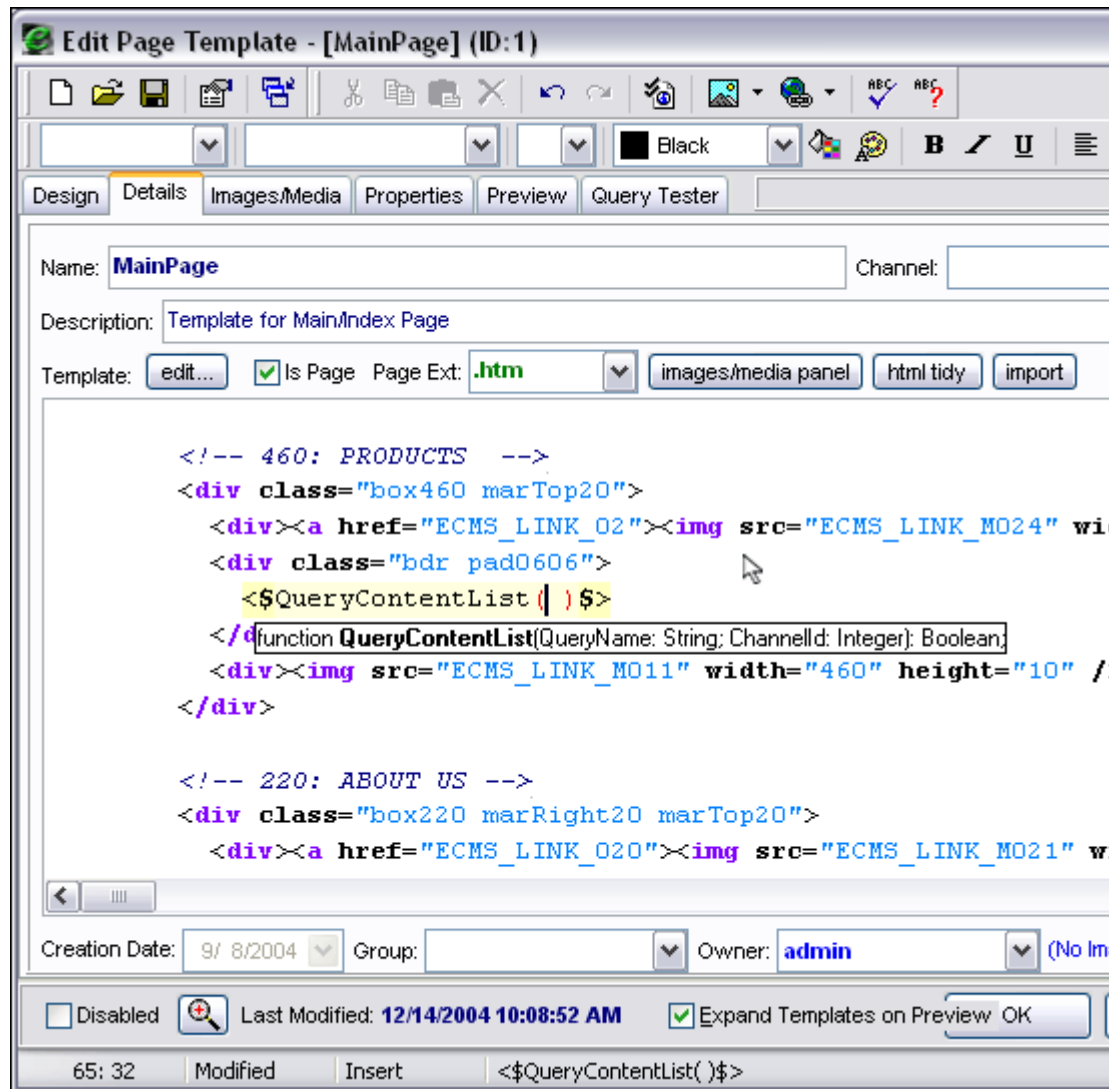


1.2.2 Getting a List of Parameters:

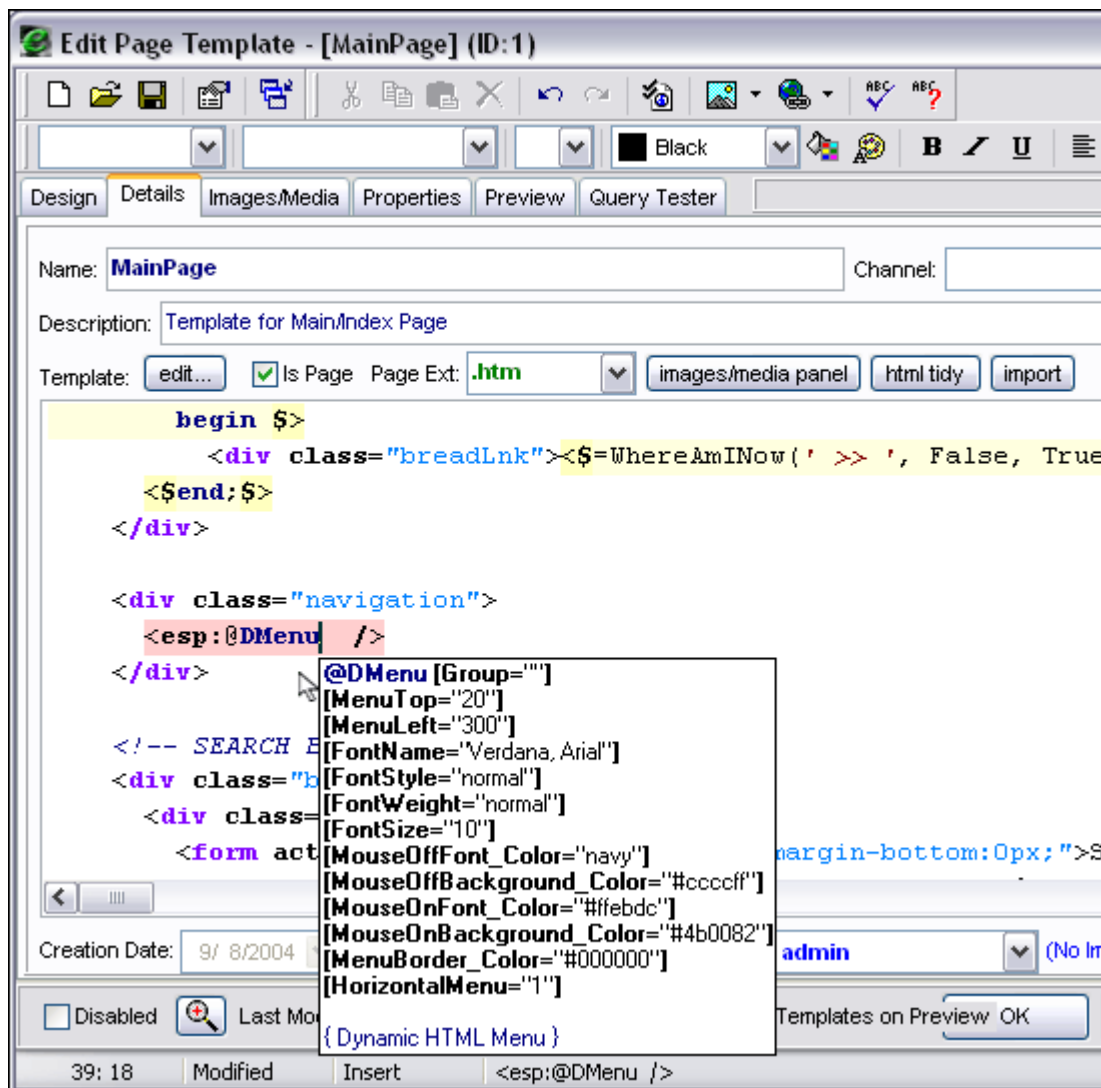
Usage: **Ctrl+Shift+Space**

Often when using a function, you might forget what parameters it accepts, the parameter datatypes and so on. By pressing **Ctrl+Shift+Space** you can easily obtain the list of parameters for a function, `<esp>` tag or `<FOREACH>` statement.

Example 1: To get the list of parameters for an eScript function, place your cursor after the first parenthesis and press **Ctrl+Shift+Space**. Encore will then list the parameters for this function as well as their datatypes:



Example 2: To obtain the list of parameters for an <esp> tag, place your cursor after the template/component name and press **Ctrl+Shift+Space**. Encore will then list the parameters for this function as well as their datatypes:

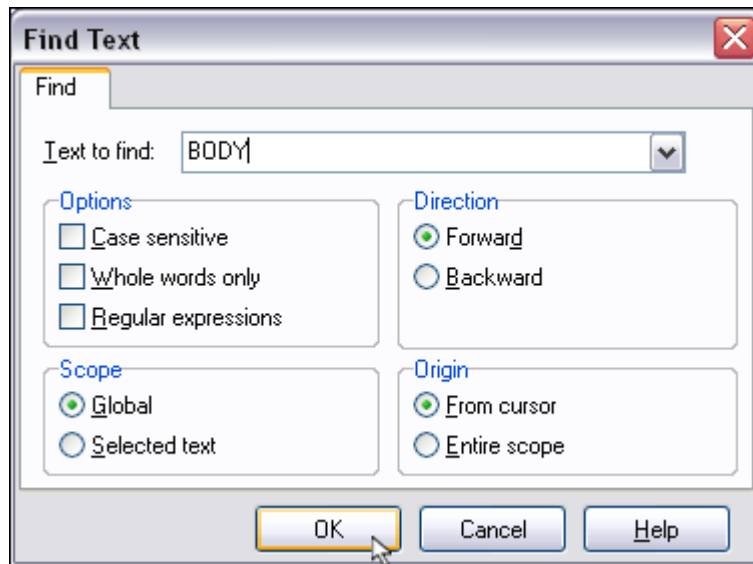


Example 3: To obtain the list of parameters for an <FOREACH> tag, place your cursor after FOREACH and press **Ctrl+Shift+Space**. Encore will then list the parameters for this function as well as their datatypes:

1.3 Find and Replace:

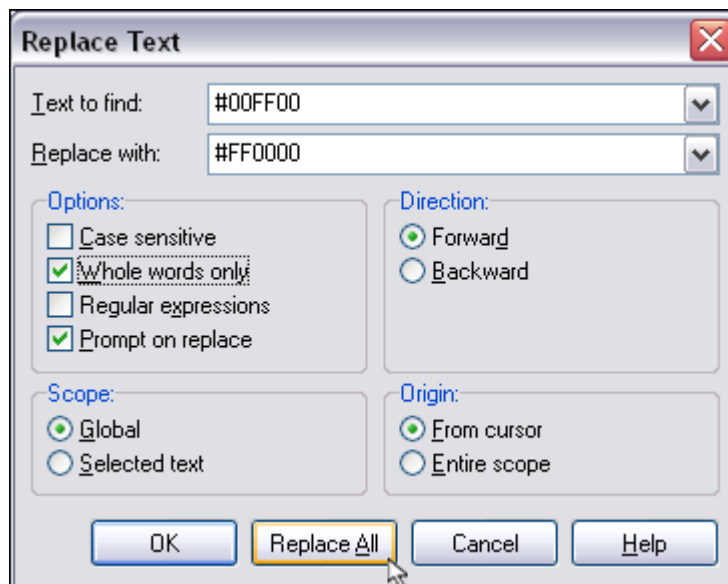
The IDE also allows you to do search and replace of text. To search for text, press **Ctrl+F** this will launch the "Find" dialog. Allowing you to search for specific text.

If you highlight a word in the IDE and then press **Ctrl+F**, the find dialog will launch with the selected word as the text to find.



To do a find and replace, press **Ctrl+R**, then specify the text to search for and the text to replace with.

If you highlight a word in the IDE and then press **Ctrl+F**, the find dialog will launch with the selected word as the text to find.



1.4 Incremental Search:

Press **Ctrl+E** to bypass the Find Text dialog box by moving the cursor directly to the next occurrence of text that you type.

When you are performing an incremental search, the Code editor status line reads "Searching For:" and displays each letter you have typed.

For example, if you type "class" the cursor moves to the next occurrence of the word, highlighting each letter as you type it. This behavior continues until a new occurrence of the string is not found, the IDE loses focus, or you press Enter or Esc.

1.5 Indent and Un-indent of Code:

Indenting and un-indenting allows you to move a block of text/code either one unit to the left or right.

Select a block of text/code and then press **Ctrl+Shift+I** to move block one unit to the right, or press **Ctrl+Shift+U** to move block one unit to the left.

1.6 Bracket Matching:

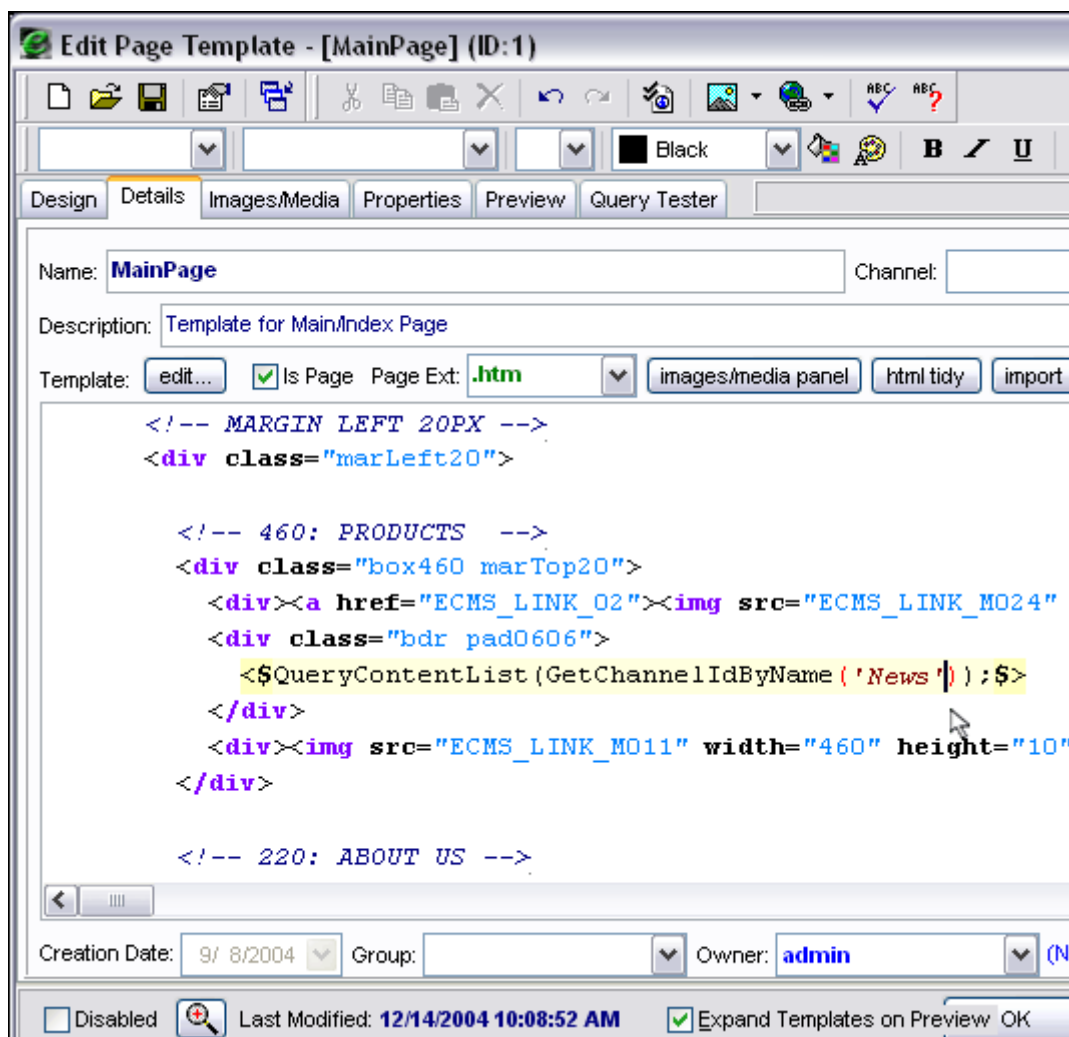
The Encore IDE also supports bracket matching to allow you to see which parentheses match up. For example, when you have a complicated statement with many parentheses, Encore will highlight matching sets of parenthesis to easily see if there are too many or too few of them.

The following sets of parenthesis are supported for bracket matching:

<> Outside of <\$ \$> tags

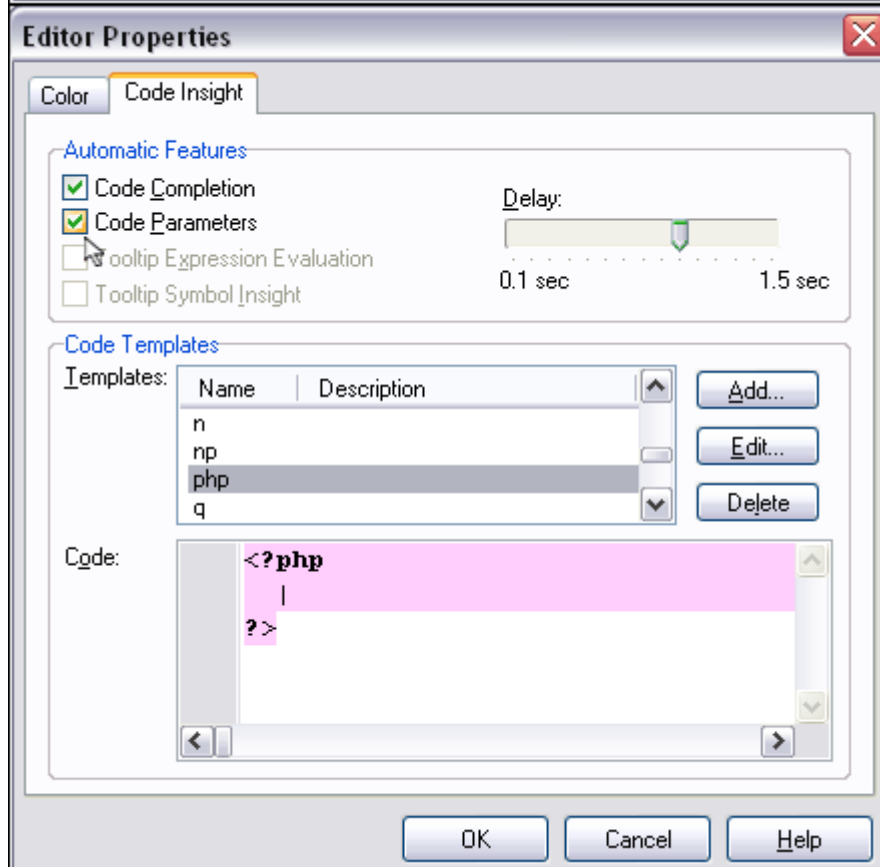
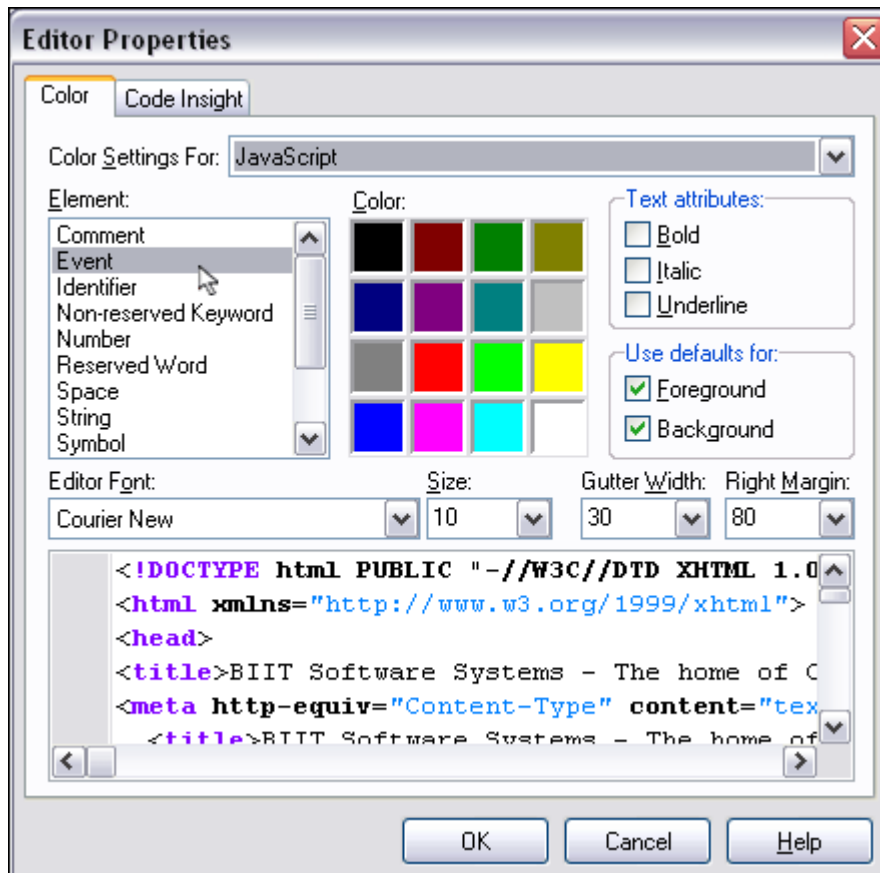
() Inside of <\$ \$> tags

[] Inside of <\$ \$> tags, in the example, note how the matching "(" are highlighted.



1.7 Setting IDE Editor Properties:

You can configure various aspects of the IDE via the Editor Properties. Right click inside the IDE editor window and select the "Editor Properties" option and configure the necessary settings. The Editor Properties dialog allows you to set syntax highlighting colors and code completion options.



Note: To expand code templates, enter the name followed by Shift+Space, example: `php<Shift+Space>`

2. Introduction to Encore Scripting Language (eScript).

2.1 What is eScript?

CMS Encore Pro supports a fully featured scripting language called eScript (Encore Script). It is based on DWS (Delphi Web Script), a subset (with some extra features) of Borland Delphi. So if you are familiar with Delphi or the Object Pascal language or any high level programming language then eScript will be very easy and intuitive to use almost immediately. So while eScript closely matches the Object Pascal language, we will not endeavor to explain programming concepts in this document.

If you wish to learn more about the Object Pascal language, there is a vast amount of information available on the internet on Object Pascal constructs, and also additionally if you wish to learn more about programming concepts in general.

eScript has been enhanced further with CMS specific functionality as used in Encore.

eScript supports programmatic operations such as being able to declare variables & perform assignments (with type checking), perform addition, subtraction, multiplication, division, etc. on variables. Use looping constructs such as for, while and repeat statements, use conditions such as if and case statements, as well as being able to create and call functions and procedures as well as many other common programming constructs found in most high level languages. It also fully supports object orientated programming (OOP).

2.2 How to use eScript?

eScript can be used inside of any functional elements of Encore e.g. templates, content, components, etc.

eScript code is always contained within eScript tags: <\$ \$>. Any code contained within eScript tags will be executed as eScript code, also the convenience of syntax highlighting means that eScript code is easily noticeable.

Example:

```
<b>TITLE: <${UpperCase(QueryFieldAsString('TITLE'))}$></b>
```

3. eScript Language Constructs.

As mentioned before, eScript is a higher level programming (scripting) language that follows the rules of most higher level programming languages. For this reason it is made up of certain language constructs such as loops, variable manipulation, etc.

3.1 Operators and Operands:

Operators behave like predefined functions that are part of the the eScript language. For example, the expression (X + Y) is built from the variables X and Y, called **operands**, with the + operator; when X and Y represent integers or floats, (X + Y) returns their sum.

Some operators behave differently depending on the type of data passed to them. For example, not performs bitwise negation on an integer operand and logical negation on a Boolean operand. Such operators appear below under multiple categories.

3.1.1 Relational Operators:

Relational operators are used to compare two operands.

Operator	Operation	Operand types	Result type	Example
=	Equality	simple, class, class reference, string	Boolean	x = y
<>	inequality	simple, class, class reference, string	Boolean	x <> y
<	less-than	simple, string	Boolean	x < y
>	greater-than	simple, string	Boolean	x > y
<=	less-than-or- equal-to	simple, string	Boolean	x <= y
>=	greater-than-or- equal-to	simple, string	Boolean	x >= y

For most simple types, comparison is straightforward. For example, I = J is True just in case I and J have the same value, and I <> J is True otherwise. The following rules apply to relational operators:

- Operands must be of compatible types, except that a real and an integer can be compared.
- Strings are compared according to the ordinal values that make up the characters that make up the string. Character types are treated as strings of length 1.

3.1.2 Boolean Operators:

The Boolean operators not, and, or, and xor take operands of any Boolean type and return a value of type Boolean.

Operator	Operation	Operand types	Result type	Example
not	negation	Boolean	Boolean	not (x or y)
and	conjunction	Boolean	Boolean	x and y
or	disjunction	Boolean	Boolean	x or y
xor	exclusive disjunction	Boolean	Boolean	a xor b

These operations are governed by standard rules of Boolean logic. For example, an expression of the form x and y is True if and only if both x and y are True.

3.1.3 Arithmetic Operators:

Arithmetic operators, which take real or integer operands, include +, -, *, /, div, and mod.

Operator	Operation	Operand types	Result type	Example
+	addition	integer, float	integer, float	x + y
-	subtraction	integer, float	integer, float	x - y
*	multiplication	integer, float	integer, float	x * y
/	Division	integer, float	float	x / 2
div	integer division	Integer	integer	x div y
mod	remainder	Integer	integer	y mod 6

The following rules apply to arithmetic operators:

- The value of x/y is of type **Float**, regardless of the types of x and y. For other arithmetic operators, the result is of type **Float** whenever at least one operand is a **Float**; otherwise, the result is of type **Integer** when at least one operand is of type **Integer**.
- The value of x div y is the value of x/y rounded in the direction of zero to the nearest integer.
- The mod operator returns the remainder obtained by dividing its operands. In other words, $x \text{ mod } y = x - (x \text{ div } y) * y$.
- A runtime error occurs when y is zero in an expression of the form x/y, x div y, or x mod y.

3.1.4 String Operators:

The relational operators =, <>, <, >, <=, and >= all take string operands (see Relational operators). The + operator concatenates two strings.

Operator	Operation	Operand types	Result type	Example
+	concatenation	string, character	string	'a' + 'b'

3.2 Assignment Operators:

An assignment statement has the form:

```
variable := expression
```

Where variable is any variable reference, and expression is any assignment-compatible expression. The "==" symbol is sometimes called the assignment operator.

An assignment statement replaces the current value of variable with the value of expression. For example,

```
I := 3;
```

Assigns the value 3 to the variable I. The variable reference on the left side of the assignment can appear in the expression on the right. For example,

```
I := I + 1;
```

Increments the value of I.

3.3 Datatypes:

A datatype is essentially a name for a kind of data. When you declare a variable you must specify its type, which determines the set of values the variable can hold and the operations that can be performed on it. Every expression returns data of a particular type, as does every function. Most functions and procedures require parameters of specific types.

eScript like the Object Pascal language is a "strongly typed" language, which means that it distinguishes a variety of data types and does not always allow you to substitute one type for another. This is usually beneficial because it lets the engine/compiler treat data intelligently and validate your code more thoroughly, preventing hard-to-diagnose runtime errors. When you need greater flexibility, however, there are mechanisms to circumvent strong typing.

3.3.1 Boolean:

Syntax:

```
aValue: Boolean;
```

Description:

Boolean values are denoted by the predefined constants True and False.

Example:

```
<$  
var isEvenNumber: Boolean;  
var X: Integer;  
  
X := RandomInt(100);  
if (X mod 2) = 0 then  
begin  
    isEvenNumber := True;  
    SendLn('Number '+ IntToStr(X) + ' is even.');end  
else  
begin  
    isEvenNumber := False;  
    SendLn('Number '+ IntToStr(X) + ' is odd.');end;  
$>
```

3.3.2 DateTime:

Syntax:

```
aValue: DateTime;
```

Description:

The integral part of a DateTime value is the number of days that have passed since 12/30/1899. The fractional part of the DateTime value is fraction of a 24 hour day that has elapsed.

Example:

```
<$  
var LastModified: DateTime;  
  
QueryContent('c', 1);  
  
LastModified := QueryFieldAsDateTime('c.LAST_MODIFIED');  
SendLn(LastModified);  
$>
```

3.3.3 Float:

Syntax:

```
aValue: Float;
```

Description:

Defines a set of numbers that can be represented with floating-point notation.

Range: 2.9×10^{-39} .. 1.7×10^{38}

Significant digits: 11-12

Example:

```
<$  
var X: Float;  
  
X := Power(2, 3);  
SendLn(FloatToStr(X));  
$>
```

3.3.4 Integer:

Syntax:

```
aValue: Integer;
```

Description:

An integer type represents a subset of the whole numbers.

Range: -2147483648..2147483647 (signed 32-bit)

Example:

```
<$  
var I: Integer;  
  
I := 100 + (14 * 10);  
SendLn(IntToStr(I));  
$>
```

3.3.5 String:

Syntax:

```
aValue: String;
```

Description:

A string represents a sequence of characters.
Maximum length: $\sim 2^{31}$ characters

Example:

```
<$  
var S: String;  
  
S := 'The rain in <b>spain</b> falls mainly <i>in</i> the  
      <b>plain</b>.';  
  SendLn(S);  
$>
```

3.3.6 Variant:

Syntax:

```
aValue: Variant;
```

Description:

Sometimes it is necessary to manipulate data whose type varies or cannot be determined at publish time. In these cases, one option is to use variables and parameters of type Variant, which represent values that can change type at publish time. Variants offer greater flexibility but consume more memory than regular variables, and operations on them are slower than on statically bound types.

Example:

```
<$  
var V: Variant;  
  
V := 100 + (0.5 * 2000);  
  SendLn(V);  
$>
```

3.3.7 Built in Classes:

- TObject
- TStringList
- Exception
- TSymbols
- TIniFiletype
- THashtable
- TIntegerHashtable
- TList
- TQueue
- TStack
- TStringHashtable

For more information about these classes, see the Object Pascal Reference Guide.

3.4 Constants:

A constant is a declared identifier whose value cannot change. For example, `const MaxValue = 237;` declares a constant called `MaxValue` that returns the integer 237.

The syntax for declaring a constant is:

const identifier = constantExpression

Here are some examples of constant declarations:

```
<$ const Min = 0;
const Max = 100;
const Center = (Max - Min) div 2;
const Message = 'Out of memory';
$>
```

3.5 Variables:

A variable is an identifier whose value can change at runtime. Put differently, a variable is a name for a location in memory; you can use the name to read or write to the memory location. Variables are like containers for data, and, because they are typed, they tell the compiler how to interpret the data they hold.

3.5.1 Declaring Variables:

The basic syntax for a variable declaration is **var identifierList: datatype;**

where `identifierList` is a comma-delimited list of valid identifiers and `type` is any valid datatype.

For example:

```
<$ var I: Integer; $>
```

declares a variable `I` of type `Integer`, while

```
<$var X, Y: Float;$>
```

declares two variables, `X` and `Y` of type `Float`.

Global variables can be initialized at the same time they are declared, using the syntax

```
<$var identifier: type = constantExpression; $>
```

where `constantExpression` is any constant expression representing a value of type `datatype`.

Thus the declaration

```
<$var I: Integer = 7; $>
```

can also be used.

If you don't explicitly initialize a global variable, the eScript initializes it to 0.

When you declare a variable, you are allocating memory which is freed automatically when the variable is no longer used. In particular, local variables exist only until the program exits from the function or procedure in which they are declared.

3.6 eScript Reserved Words:

The following reserved words are internal to eScript and cannot be redefined or used as identifiers.

and, array, as, begin, case, class, const, constructor, destructor, div, do, downto, else, end, except, finally, for, forward, function, if, inherited, is, label, mod, nil, not, of, or, procedure, property, raise, record, repeat, string, then, to, try, type, until, var, while, xor

3.7 If Statements:

There are two forms of if statement: **if...then** and the **if...then...else**. The syntax of an if...then statement is:

```
if expression then
  statement
```

Where expression returns a **Boolean** value. If expression is **True**, then statement is executed; otherwise it is not. For example,

```
<$
var I, J, Result: Integer;

I := 1000;
J := 10;
if J <> 0 then
  Result := I/J;
$>
```

The syntax of an if...then...else statement is

```
if expression then
  statement1
else
  statement2
```

Where expression returns a **Boolean** value. If expression is **True**, then statement1 is executed; otherwise statement2 is executed. For example,

```
<$
var I, J, Result: Integer;

I := 1000;
J := 10;
if J <> 0 then
  Result := I/J
else
  Result := -1;
$>
```

The then and else clauses contain one statement each, but it can be a structured statement. For example,

```
<$  
var I, J, Count, Result: Integer;  
  
I := 1000;  
J := 10;  
Count := 0;  
if J <> 0 then  
begin  
    Result := I/J;  
    Count := Count + 1;  
end  
else  
begin  
    Result := -1;  
    Count := -1;  
end;  
$>
```

Notice that there is **never** a semicolon between the then clause and the word else.

You can place a semicolon after an entire if statement to separate it from the next statement in its block, but the then and else clauses require nothing more than a space or carriage return between them.

Placing a semicolon immediately before else (in an if statement) is a common programming error.

A special difficulty arises in connection with nested if statements. The problem arises because some if statements have else clauses while others do not, but the syntax for the two kinds of statement is otherwise the same. In a series of nested conditionals where there are fewer else clauses than if statements, it may not seem clear which else clauses are bound to which ifs. Consider a statement of the form

```
if expression1 then if expression2 then statement1 else statement2;
```

There would appear to be two ways to parse this:

```
if expression1 then [ if expression2 then statement1 else statement2];
```

```
if expression1 then [ if expression2 then statement1] else statement2;
```

The eScript interpreter compiler always parses in the first way. That is, in real code, the statement

```
if ... { expression1 } then
  if ... { expression2 } then
    ... { statement1 }
  else
    ... { statement2 } ;
```

is equivalent to

```
if ... { expression1 } then
begin
  if ... { expression2 } then
    ... { statement1 }
  else
    ... { statement2 }
end;
```

The rule is that nested conditionals are parsed starting from the innermost conditional, with each else bound to the nearest available if on its left. To force the compiler to read our example in the second way, you would have to write it explicitly as

```
if ... { expression1 } then
begin
  if ... { expression2 } then
    ... { statement1 }
end
else
  ... { statement2 } ;
```

3.9 Control Loops:

Loops allow you to execute a sequence of statements repeatedly, using a control condition or variable to determine when the execution stops. eScript has three kinds of control loop: repeat statements, while statements, and for statements.

You can use the standard Break and Continue procedures to control the flow of a repeat, while, or for statement. Break terminates the statement in which it occurs, while Continue begins executing the next iteration of the sequence.

3.9.1 Repeat Statements:

The syntax of a repeat statement is:

```
repeat
  statement1; ...; statementn;
until expression
```

where expression returns a Boolean value. The repeat statement executes its sequence of constituent statements continually, testing expression after each iteration. When expression returns True, the repeat statement terminates. The sequence is always executed at least once because expression is not evaluated until after the first iteration.

Example:

```
<$  
  var Start, Stop: Integer;  
  
  Start := 0;  
  Stop := 1000;  
  repeat  
    Start := Start + 10;  
  until Start >= Stop;  
$>
```

3.9.2 While Statements:

A while statement is similar to a repeat statement, except that the control condition is evaluated before the first execution of the statement sequence. Hence, if the condition is False, the statement sequence is never executed.

The syntax of a while statement is:

```
while expression do statement
```

Where expression returns a Boolean value and statement can be a compound statement. The while statement executes its constituent statement repeatedly, testing expression before each iteration. As long as expression returns True, execution continues.

Examples of while statements include:

```
<$  
  var Stop: Integer;  
  
  Stop := 1000;  
  while stop > 0 do  
  begin  
    Stop := Stop - 10;  
  end;  
$>
```

3.9.3 For Statements:

A for statement, unlike a repeat or while statement, requires you to specify explicitly the number of iterations you want the loop to go through. The syntax of a for statement is

```
for counter := initialValue to finalValue do statement
```

or

```
for counter := initialValue downto finalValue do statement
```

Where counter is a local variable (declared in the block containing the for statement) of ordinal type, without any qualifiers.

initialValue and finalValue are expressions that are assignment-compatible with counter.

statement is a simple or structured statement that does not change the value of counter.

The for statement assigns the value of initialValue to counter, then executes statement repeatedly, incrementing or decrementing counter after each iteration.

(The for...to syntax increments counter, while the for...downto syntax decrements it.)

When counter returns the same value as finalValue, statement is executed once more and the for statement terminates. In other words, statement is executed once for every value in the range from initialValue to finalValue.

If initialValue is equal to finalValue, statement is executed exactly once. If initialValue is greater than finalValue in a for...to statement, or less than finalValue in a for...downto statement, then statement is never executed. After the for statement terminates (provided this was not forced by a break or an exit procedure), the value of counter is undefined.

For purposes of controlling execution of the loop, the expressions initialValue and finalValue are evaluated only once, before the loop begins. Hence the for...to statement is almost, but not quite, equivalent to this while construction:

Examples of for statements:

```
<$  
  var  
    I: Integer;  
  
  for I := 1 to 10 do  
  begin  
    SendLn(FloatToStr(Power(2,I)));  
  end;  
$>
```

```
<$  
  var  
    I: Integer;  
  
  for I := 10 downto 1 do  
  begin  
    SendLn(FloatToStr(Power(2,I)));  
  end;  
$>
```

4. eScript Functional Elements:

eScript is composed of a number of scripting elements, including: "eScript" functions, "FOREACH" statements, "SQL" statements, "ANCHOR" statements, "esp" tags and macros.

Each of these elements work together to build your web pages.

4.1 eScript Functions and Code:

eScript functions and any scripting code you create must be contained within <\$ \$> tags.

For example:

```
<$QueryContentList('x', GetChannelIdByName('News'));$>  
  
<$  
  var i: Integer;  
  for i := 1 to 10 do  
  begin  
    SendLn(IntToStr(i));$>  
    <br>  
<$end;$>
```

Note that the html code (
 tag) is not included in the eScript tags (<\$ \$>). Any text outside these tags get rendered as normal as html code. Also, the SendLn() function above is as alternate method to render text.

4.2 FOREACH Statements:

FOREACH statements are loop control structures, that allow you to loop through datasets. Datasets are returned by eScript functions, <SQL> structures or by internal queries.

Syntax:

```
<FOREACH [type="type"] [channel_id="channel_id"] [maxrows="rows"]
    [separator="text"] [ds="datasource"]>
    statements...
</FOREACH>
```

Explanation of the different parameters:

type: **Optional** If the *type* attribute is not specified then assumed using a <SQL>...</SQL> query operation or from an eScript query function. functions such as *QueryChannelList(QueryName, -1)*, etc. have already been specified.

Using a *type* will populate an internal query dataset, that removes the need for a "Query..()" function before the <FOREACH> statement. Also if the "channel_id" parameter is omitted, then the current channel is implicitly used.

The following are valid values for "type":

- **channel_start:** Will populate an internal dataset with a list of channel's contained under the "start" channel.
- **channel_start desc:** Works like "channel_start" except the returned list is sorted descending.
- **channel_starth:** Will populate an internal dataset with a list of channel's details including the "Home" channel.
- **channel:** Will populate an internal dataset with a the list of all sub-channels contained under a channel.
- **channel desc:** Will populate an internal dataset with a the list of all sub-channels contained under a channel, sorted in descending order.
- **channelh:** Works the same way as "channel", except the returned list will include the "Home" channel as well.
- **channelp:** Works the same way as "channel", except the returned list will include the parent channel as well.
- **channelhp:** Will populate an internal dataset with a list of channel's details including the "Home" channel and the parent channel.
- **channel prev:** Will populate an internal dataset containing the details of all the channels before the current channel.
- **channel next:** Will populate an internal dataset containing the details of all the channels after the current channel.
- **content:** Will populate an internal dataset containing all the content for a channel, except the index page.
- **content desc:** Will populate an internal dataset containing all the content for a channel, except the index page, sorted in descending order.

- **content_incl:** Will populate an internal dataset containing all the content for a channel, including the index page.
- **content_incl desc:** Will populate an internal dataset containing all the content for a channel, including the index page, sorted in descending order.
- **content prev:** Will populate an internal dataset containing the details of all the content before the current content.
- **content next:** Will populate an internal dataset containing the details of all the content after the current content.
- **content_incl prev:** Will populate an internal dataset containing the details of all the content before the current content, including the index page.
- **content_incl next:** Will populate an internal dataset containing the details of all the content after the current content, including the index page.
- **subcontent:** Will populate an internal list containing all the subcontent items assigned to a content article.
- **subcontent desc:** Will populate an internal list containing all the subcontent items assigned to a content article, sorted in descending order.
- **related_content:** Will populate a list containing all of a content items related content items.
- **related_content desc:** Will populate a list containing all of a content items related content items, sorted in descending order.
- **media:** Will populate a list of global "Image/Media" items.

channel_id: **Optional** Used in conjunction with the "type" parameter.

maxrows: **Optional** This the the maximum number of rows to return.

separator: **Optional** This inserts the text specified between every iteration.

ds: **Required** Here is where you specify the reference datasource name returned from a <SQL> .. </SQL> query operation or from an eScript query function.

Some examples of a FOREACH statements:

```
<FOREACH type="content" maxrows="5" ds="c">
  <a href="[%!c._link%]">[%!c.title%]</a><br>
</FOREACH>
```

In the above example, we use the FOREACH with a type parameter = content. This configures the FOREACH to populate an internal query. The FOREACH then loops through the dataset.

Commonly FOREACH is generally used with a preceding eScript function, which is more versatile, as shown below.

```
<$QueryChannelList('ch', GetChannelIdByName('Home'));$>
<FOREACH separator=" | " ds="ch">
  <a href="{%!ch._chlink%}">{%!ch.NAME%}</a>
</FOREACH>
```

In the above example, the eScript function "QueryChannelList", returns all the channels underneath the "Home" channel, and assigns a dataset "ch" to the result. The FOREACH statement then loops through the dataset result to output each channel name.

```
<$QueryContentList('c', GetChannelIdByName('News'));$>
<FOREACH maxrows="4" ds="c">
  <a href="{%!c._link%}">{%!c.TITLE%}</a><br>
</FOREACH>
```

In the above example, the eScript function "QueryContentList", returns all the content/articles contained in the channel "News", and assigns a dataset "c" to the result. The FOREACH statement then loops through the dataset result to output each channel name, however the "maxrows" parameter limits the loop to 4 executions.

```
<SQL name="user">
  SELECT * FROM USERS
</SQL>

<table>
  <tr>
    <td>Name</td>
    <td>Email</td>
  </tr>
  <FOREACH ds="user">
    <tr>
      <td>{%!user.TITLE%}&nbsp;{%!user.INITIALS%}&nbsp;{%!user.NAME%}</td>
      <td>{%!user.EMAIL%}</td>
    </tr>
  </FOREACH>
</table>
```

In the above example, we use a manual query to load the list of Encore users from the database. We then use the FOREACH to loop through the returned dataset. Note that the name of the query serves as the dataset name in this case.

To summarize, FOREACH statements are loop control structures in Encore that allow you to loop through datasets returned by eScript functions, <SQL> ... </SQL> structures or internal queries.

4.2.1 Using a <FOREACH> to simulate a "between" statement:

Also, FOREACH statements can be used to simulate "between" statements as in the following example:

```
<$var i : integer = 0;$>
<FOREACH type="content">
  <$inc(i, 1);$>
  <$if (i = 1) then begin $>*** <b>before text</b> ***<br><$end;$>
  <$if (i > 1) then begin $>*** <b>between text</b> ***<br><$end;$>
  <$if (i mod 2 = 0) then begin $>*** <b>even text</b> ***<br><$end;$>
  <$if (i mod 2 = 1) then begin $>*** <b>odd text</b> ***<br><$end;$>
  SOME TEXT<br><br>
</FOREACH>
<$if (i > 0) then begin $>*** <b>after text</b> ***<br><$end;$>
<$if (i = 0) then begin $>*** <b>else text</b> ***<br><$end;$>
```

4.3 SQL Structures:

Syntax:

```
<SQL [name="name"]>
  sql query...
</SQL>
```

The <SQL> tag allows for specifying a custom database query to run. The name attribute is optional, and if not specified then assumed specifying the main query (equivalent to using name="").

This tag is the equivalent of using the QueryRun(...) scripting function and internally the query specified within the <SQL>...</SQL> tags does also get converted to use the QueryRun(...) function, but using the <SQL> tag is more convenient and more readable, also the query within these tags is also syntax highlighted according to SQL syntax.

The <SQL> structure can be used in conjunction with a FOREACH statement. In fact most of the eScript functions like "QueryContentList" etc internally translate to a <SQL> statement. The <SQL> statement is more of a "power-user" feature, since it does require some knowledge of the database and table structure within Encore, for this reason, most common functions are wrapped up within eScript functions.

An example of a <SQL> and <FOREACH> statement:

```
<SQL name="list">
  SELECT CHANNEL_ID, NAME, DESCRIPTION
  FROM CHANNELS
  WHERE ((DISABLED_IND IS NULL) OR (DISABLED_IND <> 'Y'))
  ORDER BY NAME, CHANNEL_ID
</SQL>

<FOREACH ds="list">
  Channel: <a href="{%!list._chlink%}">{%!list.NAME%}</a><br>
  {%!list.DESRIPTION%}<br>
</FOREACH>
```

4.4 ANCHOR Tags:

Syntax:

```
<ANCHOR [name="anchor_name" ]>
  anchor text...
</ANCHOR>
```

The most useful purpose is to allow specifying text in templates/components that will automatically get inserted into the `<HEAD>...</HEAD>` or `<BODY...>` sections of a page even though the template or component might not have direct access to the final page where those tags has been defined.

The `<ANCHOR>` tag allows for specifying anchor text which will automatically get inserted at a predefined anchor point in the html after a page is generated. The most useful purpose is to allow specifying text in templates/components that will automatically get inserted into the `<HEAD>...</HEAD>` or `<BODY...>` sections of a page even though the template or component might not have direct access to the final page where those tags has been defined.

Note: You don't need to use the `GetAnchor(...)` function to retrieve the value for the anchor for a `<HEAD>` section, if any anchor text is specified for this tag then it will automatically be inserted just before the closing `</HEAD>` or `</head>` tag. The same applies to an anchor for the `<BODY>` section which will get inserted into the `<BODY>` tag.

The name attribute is optional, and if not specified (or `name=""`) then assumed specifying the `"*HEAD"` anchor. The name can optionally be prefixed with `"+"` or `"*"`, where `"+"` will add the current specified anchor text to any text that might already be specified for an anchor with the name specified, and `"*"` will only add the specified text if it's not already included in the anchor (i.e. it will add the anchor text uniquely!).

This tag is the equivalent of using the `SetAnchor(...)`, `AddToAnchor(...)` or `AddUniqueToAnchor(...)` scripting functions and internally the anchor text specified within the `<ANCHOR>...</ANCHOR>` tags does also get converted to use one of these scripting functions, but using the `<ANCHOR>` tag is more convenient and more readable.

Currently special anchors named `"HEAD"`, `"BODY"`, `"BODY:background"`, `"BODY:onload"` and `"BODY:onunload"` is fully and automatically supported (these are automatically inserted into either the `<HEAD>` section or `<BODY>` section of an html page if any of these anchors are specified), otherwise must use the `<%=GetAnchor(name)%>` scripting function to retrieve an anchor value.

Examples:

```
<HTML>
  <HEAD>
    <TITLE>Test</TITLE>
  </HEAD>
  <BODY>
    ...
    ...
    <ANCHOR>
      <style>
        <!--
          //This style will go into the HEAD section
          .MyStyle { font-color: #FFAABB }
        <!-->
      </style>
    </ANCHOR>
    ...
    ...
  </BODY>
</HTML>
```

Equivalent to above anchor section, if name omitted then equivalent to **"*HEAD"**:

```
<ANCHOR name="*HEAD">
  <style>
    <!--
      //This style will go into the HEAD section
      .MyStyle { font-color: #FFAABB }
    //-->
  </style>
</ANCHOR>
```

Showing how to specify **BODY:onload/BODY:onunload/BODY:background/BODY** anchors:

Any onload anchors are converted to a sequential list (separated by spaces, not newlines) before being inserted into the current BODY tag.

If there is already an onload="..." attribute present it will be inserted after any other onload items, else an onload attribute is added to the end of the **<BODY>** tag.

The same principle applies to the onunload="..." and background="..." properties.

However onunload will also have it's items sequentially but they will be automatically reversed, so that the last item added will be the first to unload (as this is normally recommended behaviour), also this list will be inserted in-front of any other items if an onunload="" attribute is already present.

Since the background="..." attribute can only specify a single image, specifying multiple items via multiple anchors (especially using the '+') syntax will probably cause an error, so ensure that only a single or blank value is returned for the background attribute.

```
<!-- NOTE: Ensure that any javascript function names specified
with ';', this helps to ensure that if multiple
ANCHORS on BODY:onload or BODY:onunload are specified
that each statement will be correctly separated. -->

<ANCHOR name="+BODY:onload">InitFunc1();</ANCHOR>
<ANCHOR name="+BODY:onunload">DoneFunc1();</ANCHOR>
<ANCHOR name="+BODY:onload">InitFunc2();</ANCHOR>
<ANCHOR name="+BODY:onunload">DoneFunc2();</ANCHOR>
```

Overriding the *background* attribute in the current *BODY* tag:

```
<ANCHOR name="BODY:background">images/stars.gif</ANCHOR>
```

Here we are overriding the entire *BODY* tag of the current html page:

```
<ANCHOR name="BODY"><BODY alink="#ABCDEF" bgcolor="#FFFFFF"></ANCHOR>
```

User defined Anchor:

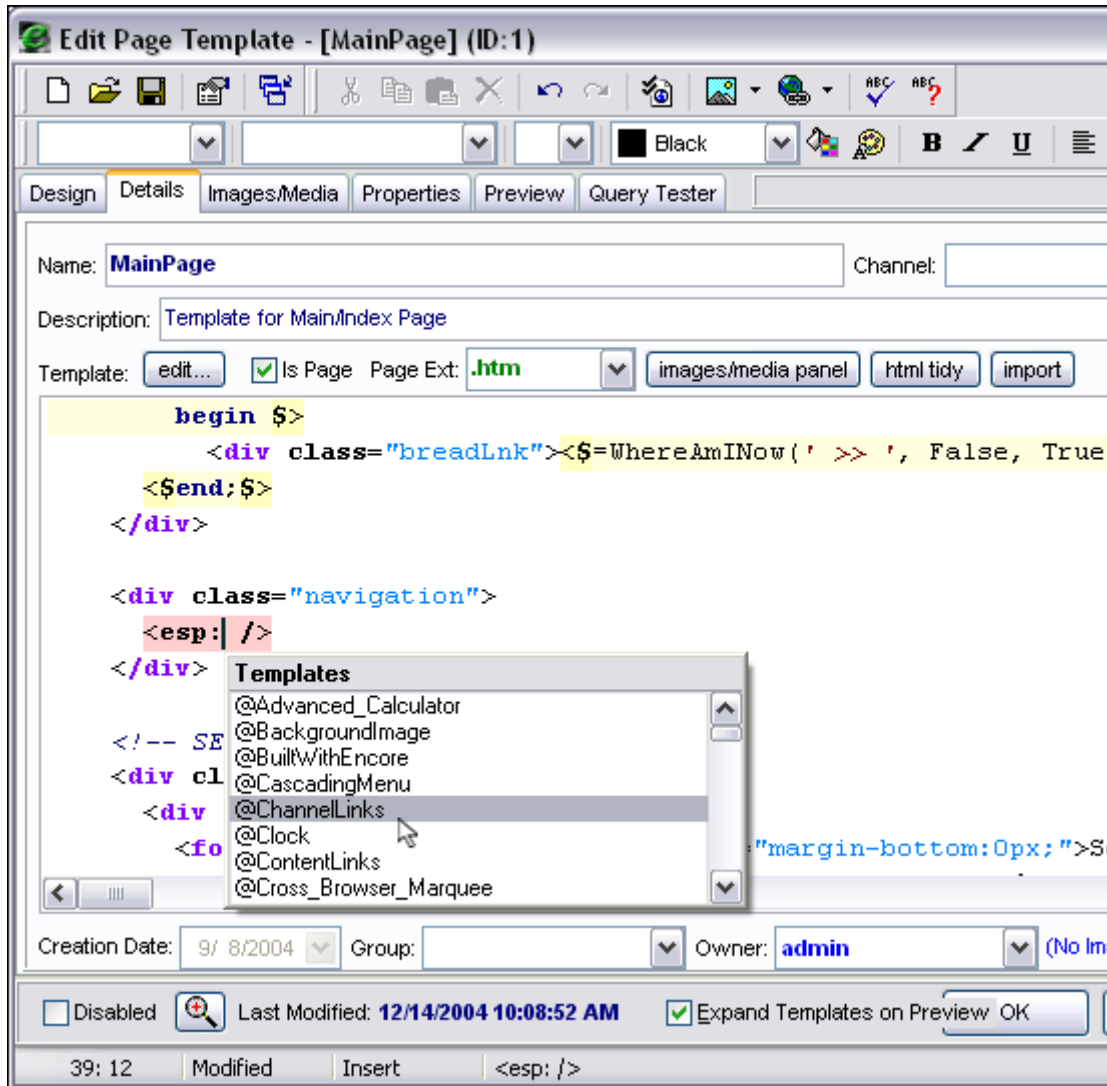
```
<ANCHOR name="+META">
<META name="AUTHOR" content="User One">
</ANCHOR>
...
...
<%=GetAnchor('META')%>
```

4.5 ESP <esp> Tags:

If you create your own components or templates (non-page templates) then you can use the <esp> tag to insert the component/template into the page or another templates.

Often it makes sense to put certain often used code like menus, headers and footers into templates. These templates would then be displayed via <esp> tags.

Simply type <esp:/> with the cursor right after the '<esp:' part and then press <Ctrl+Space> to get a list of all available components & templates.

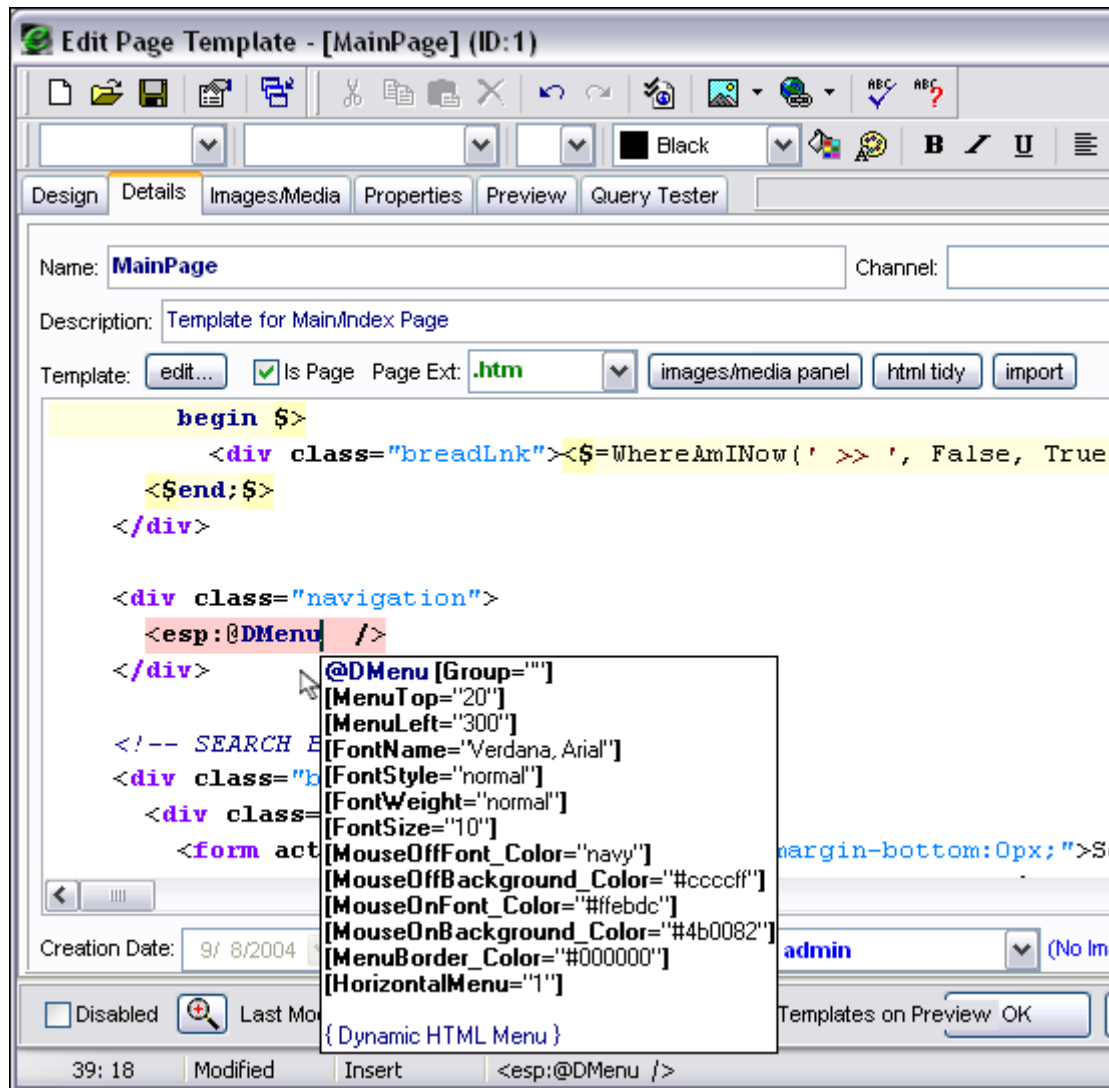


Component names usually start with the '@' character to distinguish them from regular templates. Although components and templates are functionally equivalent, components are usually designed in a project independent manner, and generally without dependencies on other templates/components unless if the other template/component is also defined in the same component package.

Components are designed this way so that you can use them in almost any project without worrying about whether they will work on that particular set of pages & templates.

Also through code completion you can pass parameters to your <esp> tags. For example, if you are calling in a template that contains properties, you can use code completion to display the list of properties.

Place the cursor after the template name in the <esp> tag, and press <Ctrl+Shift+Space>, to obtain the list of properties for the template.



4.6 Macros:

There are a number of macros that can be used in Encore. They get expanded at publish-time to generate meaningful code/values.

Macros can be divided into the following types:

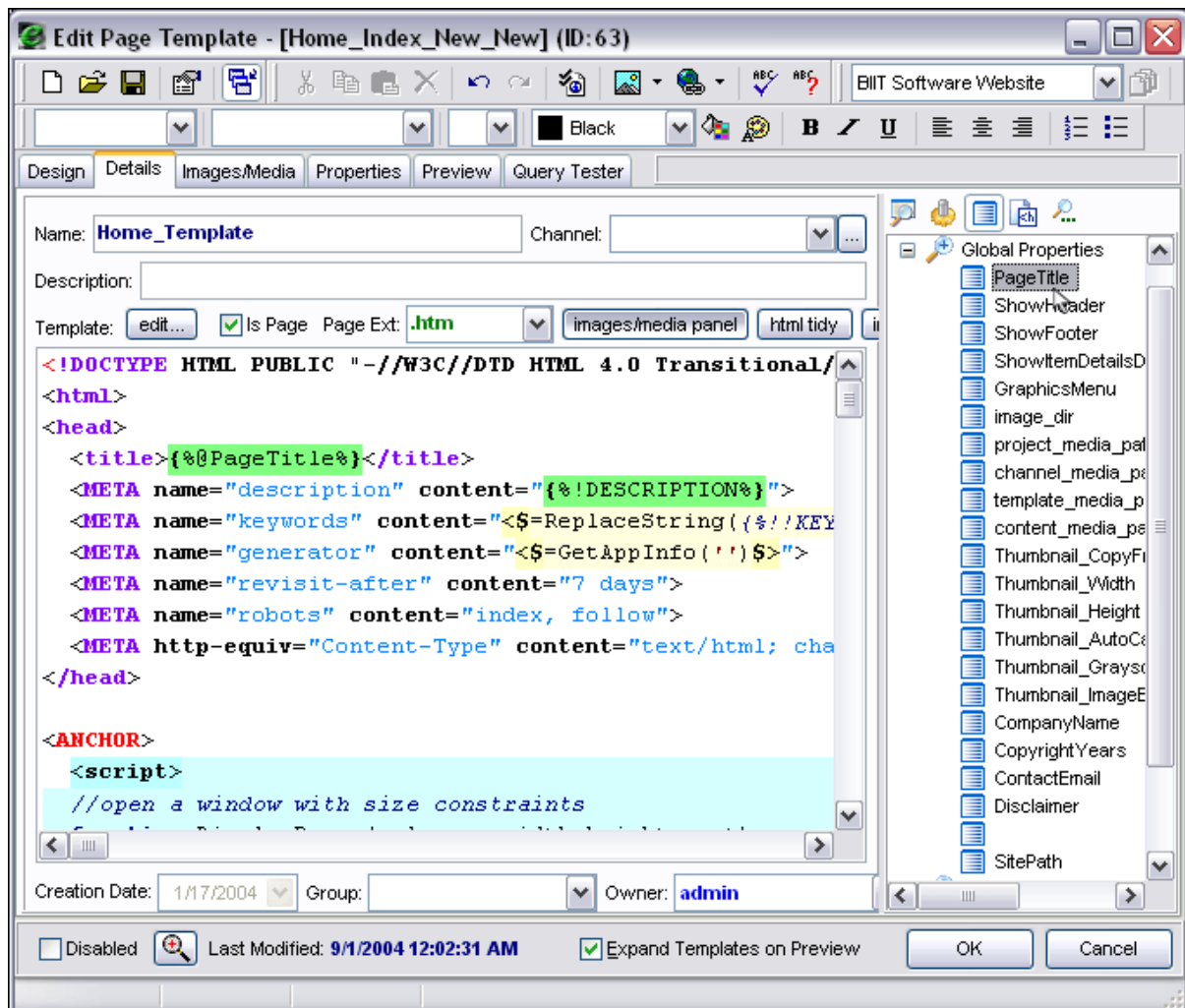
- Global properties
- Global properties (internal)
- Local Properties
- Images/Media
- Content Macros
- Query field macros

4.6.1 Global/Project Properties:

Syntax: **{%@GlobalPropertyName%}**

Global properties are accessible from the images/media panel of any template editor window. They are the list of project properties like "Page Title" etc and are added via the edit project dialog.

To access the list of global properties, click on the "variables/properties" button of the images/media panel and expand the "Global Properties" node. To use any of the properties, simply drag and drop one of them into your code.

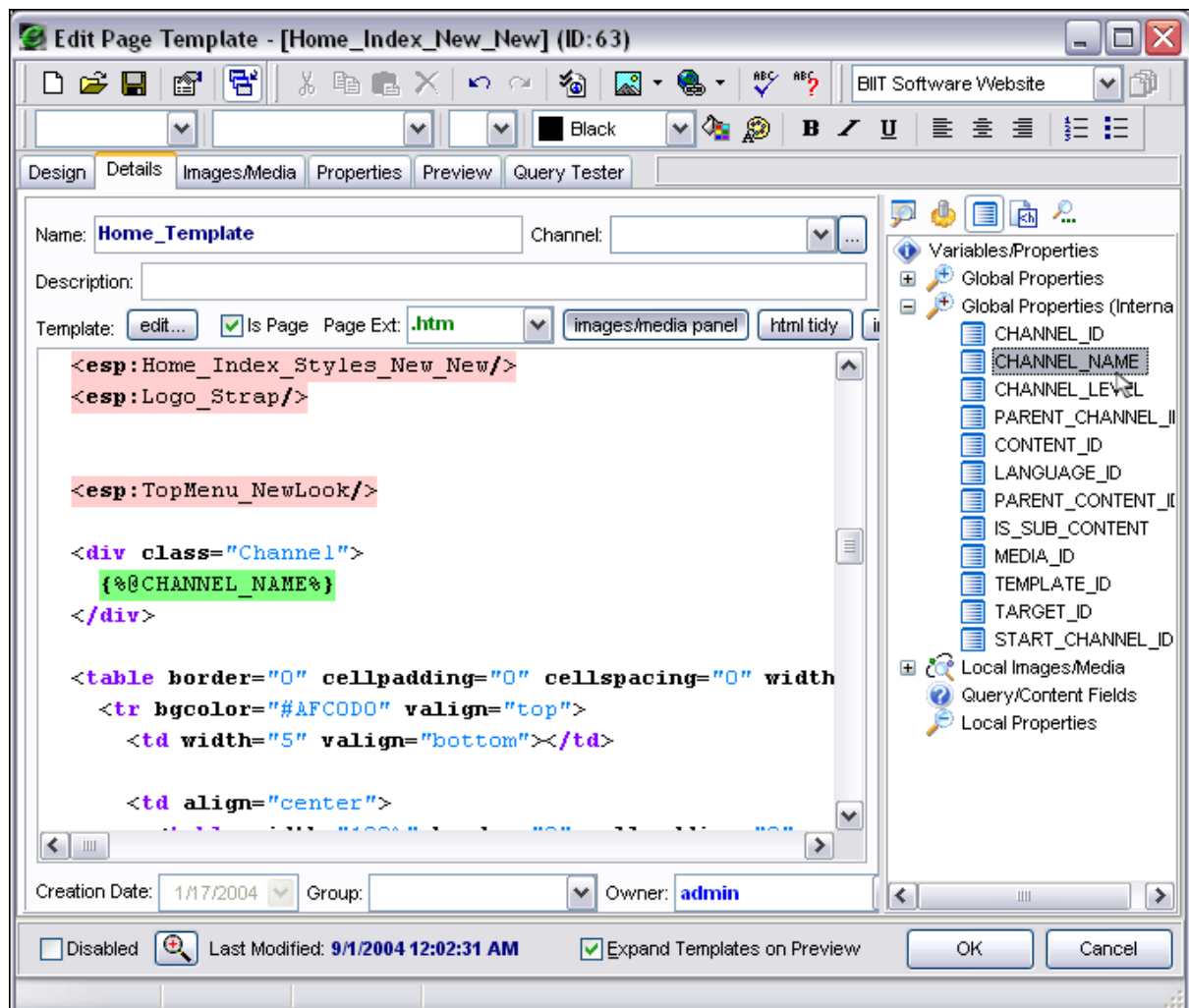


4.6.2 Global Properties (Internal):

Syntax: **{%@PropertyName%}**

These properties and their values are controlled by Encore are made available in the context of what you are creating. For example, if you are creating a template, then Encore will populate this list with the list of valid template property values.

To use any of the properties, simply drag and drop one of them into your code.

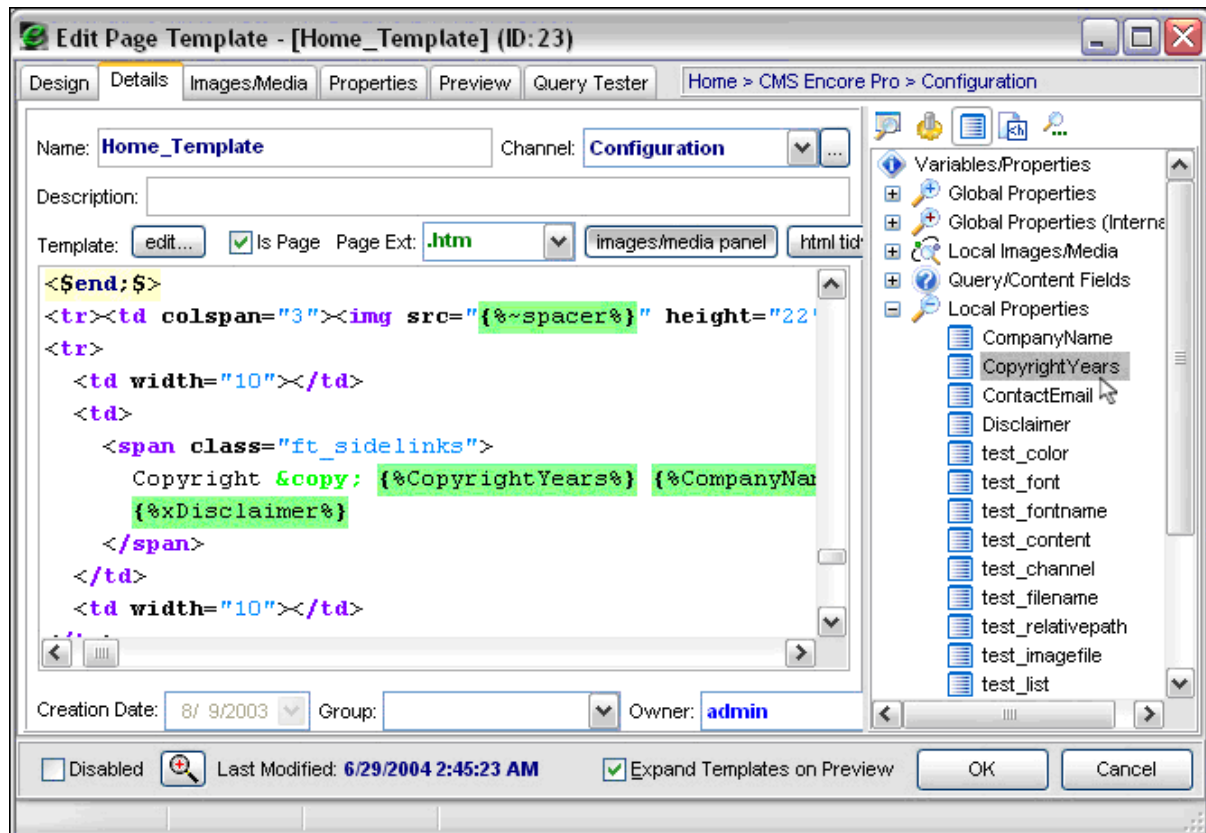


4.6.3 Local Properties:

Syntax: **{%PropertyName%}**

Local properties are the local property values as defined for a particular template/component. The list of local properties can be edited in the "Properties" tab of the template/component.

To use any of the properties, simply drag and drop one of them into your code.



4.6.4: Images/Media:

Syntax:

```

```

The macro will expand to the value of the image name (including the extension)

```
{%-ImageName*%}
```

Note the star character (*) at the end of the image macro name. This macro will expand to a full image tag, including specified alt text, width and height dimensions, etc.

```
{%-ImageName**%}
```

Note the double star (**) at the end of the image macro name. This macro will expand to show the thumbnail version of the image with a mouse over link to show the full size image.

```

```

The macro will expand to the value of the thumbnail image name (including the extension). The thumbnail will only be displayed if defined.

```
{%-ImageName_thumb*%}
```

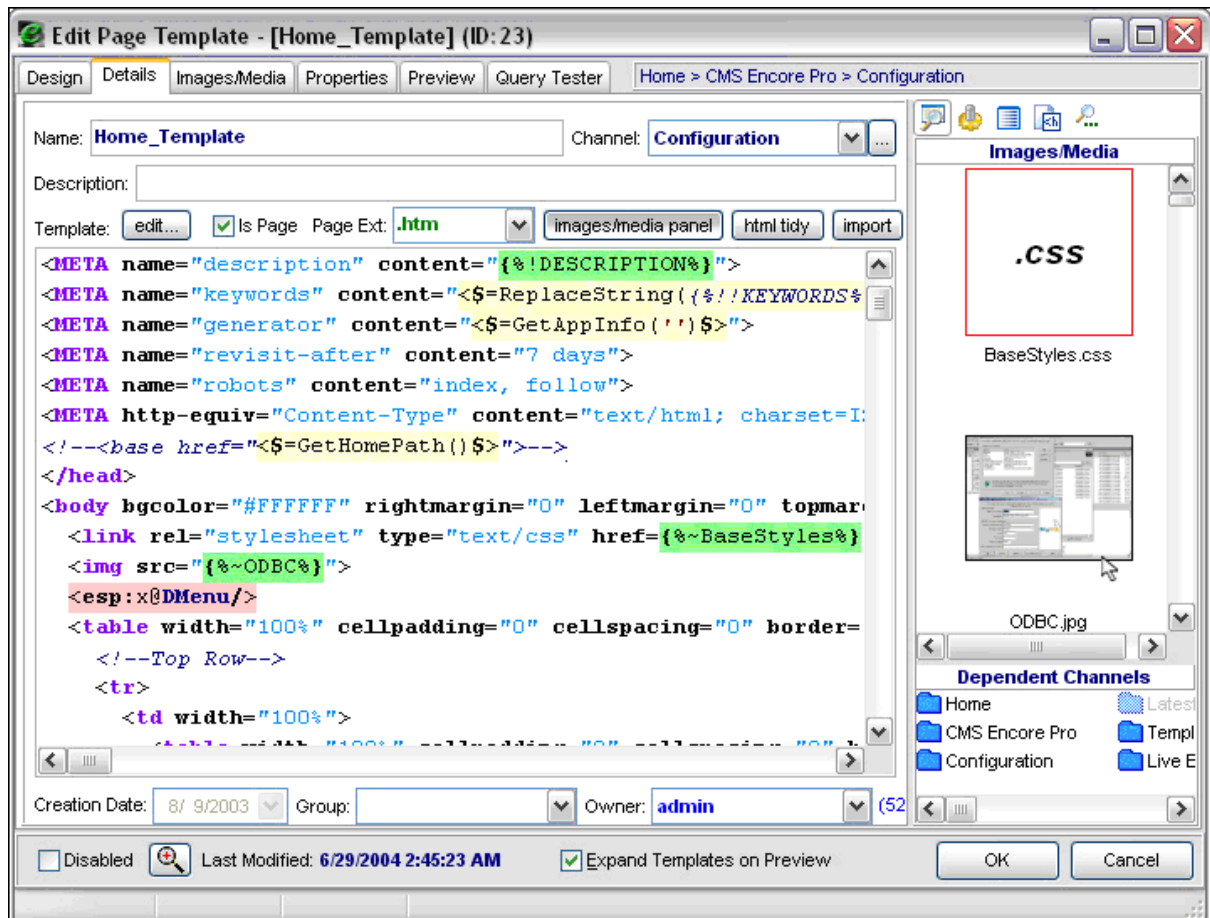
Will display a thumbnail version of the image, if defined.

Note the star character (*) at the end of the image macro name. This macro will expand to a full image tag, including specified alt text, width and height dimensions, etc.

```
<link rel="stylesheet" type="text/css" href="{%-StyleSheetName%}">
```

This macro will expand to link this particular stylesheet.

Images and media, especially stylesheets can also have embedded macros referencing images/media. Also, images/media could be any media content eg: flash object, wav files, video files etc.



4.6.5: Content/Channel Macros:

Syntax: `{%!ContentMacroName%}`

Content macros are macros that extract content related information, eg content title or content text, etc. The values they return are within the context of the page using the template. So these values will reflect the current article using the template.

Content Related Macro's:

CONTENT_ID: The content id.

TITLE: The content title.

CONTENT_TEXT: The content text.

CONTENT_SUMMARY: The content summary.

DESCRIPTION: The content description.

CONTENT_GROUP: The group the content belongs to.

CHAPEU: The content chapeu.

STATUS: The status of the content

KEYWORDS: The keywords for this content.

CREATED_DATE: The date/time the content was created.

LAST_MODIFIED: The date/time the content was last modified.

USER_NAME: The name of the user who created the content.

USER_INITIALS: The initials of the user who created the content.

USER_TITLE: The title of the user who created the content.

_link: The link (url) to this content(article) page.

_abslink: The absolute link (url) to this content(article) page (used with SetAbsLinkPrefix() function).

Channel Related Macro's:

CHANNEL_ID: The channel id.

NAME: The name of the channel.

SHORT_NAME: The short-name of the channel.

DESCRIPTION: The channel description.

LONG_DESCRIPTION: The channel long description.

PARENT_ID: The channel's parent id.

CHANNEL_LEVEL: The hierarchical level of the channel.

HAS_CHILDREN: A value of Y/N indicating if the channel has sub-channels (children).

CHANNEL_GROUP: The group the channel belongs to.

CREATED_DATE: The date/time the channel was created.

LAST_MODIFIED: The date/time the channel was last modified.

_chlink: The link (url) to this channel.

_abschlink: The absolute link (url) to this channel.

_chttitle: The name/title of the channel (multi-lingual aware).

A very important point to bear in mind is that you may need to wrap your content macros in underscore characters if your content contains any eScript code as follows: `{%!_CONTENT_TEXT_%}`

Also note if needing to access a query field within script code, then you should use the QueryFieldAsString() or QueryFieldAsInteger scripting functions.

eg. `<$if (QueryFieldAsString('TITLE') <> ' ') then begin $>...<$end;$>`

Example:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
  <HEAD>
    <TITLE>{%@PageTitle%} | {%!TITLE%}</TITLE>
  </HEAD>
  <BODY>
    <b>{%!TITLE%}</b><br>
    <br>
    {%!CONTENT_TEXT%}
  </BODY>
</HTML>
```

4.6.6 Query Field Macros:

Syntax: `{%!DataSource.FieldName%}`

Query field macros usually go along with any eScript element that returns a datasource.

For example:

```
<$QueryChannelList('ch', GetChannelIdByName('Home'));$>
<FOREACH ds="ch">
  <a href="{%!ch.chlink%}">{%!ch.NAME%}</a><br>
</FOREACH>
```

In this case, the function `QueryChannelList` returned a datasource called "ch". We then use the datasource as part of the query field macro and we specify the fieldname with the datasource.

4.6.7 Using Macro Suffixes to format text:

Encore has a built in set of formatting commands that you can use to format text values. For example, you may wish to convert text to uppercase, lowercase, convert to numeric format or convert to html format, etc. Using macro suffixes you can easily accomplish this.

The Macro Suffix List:

```
^ - Converts to UpperCase.
^^ - Converts to LowerCase.
$ - Converts to single quoted string.
$$ - Converts to double quoted string.
& - Convert text to html format.
# - Converts to numeric value (if applicable).
+ - Expands newline characters in the text.
++ - Expands any newline characters as well as expanding any macros
contained within the text.
```

How to use Macro Suffixes:

Macro names must be suffixed with the required macro suffix as follows:
`{%!MACRO_NAME^$&#%}`

Examples:

`{%!NAME^%}` will convert the macro value to uppercase.

`{%!NAME^^%}` will convert the macro value to lowercase.

`{%!TITLE$%}` will wrap the macro value in single quotes.

`{%!TITLE$$%}` will wrap the macro value in double quotes.

`{%!LONG_DESCRIPTION&%}` will convert the macro value to html format.

`{%!CONTENT_ID#%}` will convert the macro value to a numeric value.

`{%!(ShortBlurb_Memo)+%}` will expand any newline characters in the text.

`{%!(ShortBlurb_Memo)++%}` will expand any newline characters as well as expanding any macros contained within the text.

Note: User defined content fields are surrounded by '(' and ')' characters, as can be seen in the last two examples above.

Section B: The eScript Function Reference

This chapter lists all of the current eScript functions with a short description and examples. The eScript functions are listed by category.

1. Channel(Folder) Related Functions and Procedures:

This group of functions handle channel(folder) related functionality.

GetAbsChannelLink function

Usage: **function** GetAbsChannelLink(ChannelId:integer): **string**;

ChannelId: Integer The channel's Id value

Return Type: String

Description:

Returns the absolute channel link/path + default page name for the channel defined by ChannelId relative to the root path defined by AbsLinkPrefix (default = '').

Example: *This example displays an absolute link to the channel with Id=1.*

Click here to go home <a href="<%=GetAbsChannelLink(1)%>">Home
--

GetChannelIdByName function

Usage: **function** GetChannelIdByName(Name: String): **Integer**;

Name: String A valid channel name

Return type: Integer

Description:

Returns the Id for a channel specified by **Name**. If more than one channel has the same name then the function returns the Id for the first channel with this name. If an invalid name is specified, then the function returns a value of zero.

Example: *This example returns the name Id for the channel called "Home".*

<div class="text"> The Home channel has an Id of: <%=GetChannelIdByName('Home')%> </div>
--

GetChannelIdByPath function

Usage: `function GetChannelIdByPath(ChannelIdStart: Integer;
ChannelPath: String): Integer;`

ChannelIdStart: Integer

The channel to start from, a value of -1, indicates the current channel

ChannelPath: String

The path of the channel to search

Return Type: Integer

Description:

Returns the channel_id when given the channel to start from (a value of -1 indicates current channel) and the channel paths.

NOTE: Using this function is only feasible if each channel has its path filled in, and the path is singular (i.e. does not contain '\' or '/' in the individual channel paths) and that there are not multiple channels assigned to the same path (which is allowed) as will then always only match the channel with first matching path.

Also the path specified is case sensitive and must match the channel's path exactly.

If any part of the specified path does not match then this function returns -1.

Example: *These examples show two different methods of retrieving the channel Id from the channels path. One from the "root" path and the other from the "current" path.*

```
<%=GetChannelIdByPath(-1, '\news\localnews\sport')$> <!--from root path-->  
>  
<%=GetChannelIdByPath(-1, 'localnews\sport')$> <!--from current path-->
```

GetChannelIdByNamePath function

Usage: `function GetChannelIdByPath(ChannelIdStart: Integer; ChannelNamePath, NameSeparator: String): Integer;`

ChannelIdStart: Integer

The channel to start from, a value of -1, indicates the current channel

ChannelNamePath: String

The path of the channel to search

NameSeparator: String

Separator between channel names

Return Type: Integer

Description:

Returns the channel_id when given the channel to start from (a value of -1 indicates current channel) and the channel names separated by character(s) defined by NameSeparator. If NameSeparator is blank (i.e. empty string) then a separator of '\' is assumed.

NOTE: Using this function is only feasible if each channel in a particular branch has a unique name.

Also the path specified is case sensitive and must match the channel's name exactly.

If any part of the specified name does not match then this function returns -1.

Example: *These examples show two different methods of retrieving the channel Id. One from the "root" path and the other from the "current" path.*

```
<%=GetChannelIdByNamePath(-1, '\News\Local News\Sport', '\')$> <!--Root path-->
<%=GetChannelIdByNamePath(-1, 'Local News|Sport', '|')$> <!--From current path-->
```

GetChannelIdByShortName function

Usage: `function GetChannelIdByShortName(ShortName: String): Integer;`

ShortName: String

A valid channel name

Return type: Integer

Description:

Returns the Id for a channel specified by **ShortName**. If more than one channel has the same ShortName then the function returns the Id for the first channel with this ShortName. If an invalid ShortName is specified, then the function returns a value of zero.

Example: *This example returns the Id for the channel called using its's "shortname".*

```
<div class="text">
  The Home channel has an Id of: <%=GetChannelIdByShortName('home')$>
</div>
```

GetChannelImageSrc function

Usage: `function GetChannelImageSrc(SeqNo: Integer; Border: Integer; Thumbnail: Boolean; WithLink: Boolean; OtherAttribs: String): String;`

SeqNo: Integer

The sequence number corresponding to the image to retrieve.

Border: Integer

Specifies the thickness of the border, 0 = no border.

Thumbnail: Boolean

If True will display the image thumbnail

WithLink: Boolean

Provides a link to the original image, if Thumbnail is True

OtherAttribs: String

Allows you to specify extra formatting for example class code etc.

Return type: String;

Description:

Returns the current Channel's associated local image/media source link, or full IMG tag, where SeqNo is the sequence number of the image/media to retrieve. A value of -1 for SeqNo will reference the first image/media.

Example: *This function loops through the images assigned to the channel "Home" and then displays each one of the images on a newline.*

```
<$var ImageCounter: Integer=0;$>
<$QueryChannelMediaList('q', GetChannelIdByName('Home'));$>
<FOREACH ds="q">
  <$Inc(ImageCounter,1);$>
  <%=GetChannelImageSrc(ImageCounter, 0, False, True,
    'class="mystyle"')$>
  <br>
</FOREACH>
```

GetChannelImageSrc2 function

Usage: `function GetChannelImageSrc2(Title: String; OtherAttribs: String): String;`

Title: String

The title(name) of the image

OtherAttribs: String

Allows you to specify extra formatting for example class code etc.

Return Type: String

Description:

Returns the current Channel's associated local image/media source link, or full IMG tag referenced by 'Title'. Append the Title with '*' to return the full IMG tag, or with '**' to automatically create a thumbnail with a link to the full image, else only the source link is returned.

Example: *This function will display a specific image assigned to the current channel, the image is retrieved by it's name.*

```
This is the logo for the <%=GetCurrChannelName$> channel:
<%=GetChannelImageSrc2('logo_top', 'class="mystyle"')$>
```

GetChannelImageSrc2ById function

Usage: `function GetChannelImageSrc2ById(ChannelId: Integer; Title: String; OtherAttribs: String): String;`

<p>ChannelId: Integer The channel Id to search</p> <p>Title: String The title(name) of the image</p> <p>OtherAttribs: String Allows you to specify extra formatting for example class code etc.</p>
--

Return Type: String

Description:

Returns the specified Channel's associated local image/media source link, or full IMG tag referenced by 'Title'. Append the Title with '*' to return the full IMG tag, or with '**' to automatically create a thumbnail with a link to the full image, else only the source link is returned.

Example: *This function will display a specific image assigned to the specified channel, the image is retrieved by it's name.*

<p>This is the logo for the <code><%=GetChannelName(2)%></code> channel: <code><%=GetChannelImageSrc2ById(2, 'logo_top', 'class="mystyle"%></code></p>
--

GetChannelImageSrcById function

Usage: `function GetChannelImageSrcById(ChannelId: Integer; SeqNo: Integer; Border: Integer; Thumbnail: Boolean; WithLink: Boolean; OtherAttribs: String): String;`

<p>ChannelId: Integer The channel's Id value.</p> <p>SeqNo: Integer; The sequence number corresponding to the image to retrieve.</p> <p>Border: Integer; Specifies the thickness of the border, 0 = no border.</p> <p>Thumbnail: Boolean; If True will display the thumbnail version of the image.</p> <p>WithLink: Boolean; Provides a link to the original image, if Thumbnail is True.</p> <p>OtherAttribs: String Allows you to specify extra formatting for example class code, etc.</p>

Return Type: String

Description:

Similar to `GetChannelImageSrc`, except that you can specify the channel via ChannelId (or -1 for current channel) associated local image/media source link, or full IMG tag, where SeqNo is the sequence number of the image/media to retrieve. A value of -1 for SeqNo will reference the first image/media.

Example: *This function will display a specific image assigned to the specified channel.*

<p>This is the logo for the <code>home</code> channel: <code><%=GetChannelImageSrcById(GetChannelIdByName('Home'), 1, 0, False, True, 'class="mystyle"%></code></p>
--

GetChannelLink function

Usage: `function GetChannelLink(ChannelId: Integer): String;`

ChannelId: Integer

An integer Id value for a channel.

Return type: String.

Description:

Returns the full channel link/path + default page name (such as 'index.htm'), defined by ChannelId. If an invalid ChannelId is specified then the output directory is returned.

Example: *This example displays links to the channels "Home" and "News".*

```
<$  
  var HomeChannelLink, NewsChannelLink: String;  
  
  HomeChannelLink := GetChannelLink(GetChannelIdByName('Home'));  
  NewsChannelLink := GetChannelLink(GetChannelIdByName('News'));  
$>  
<a href="<%=HomeChannelLink%>">Home</a> <br>  
<a href="<%=NewsChannelLink%>">News</a>
```

GetChannelListStr function

Usage: `function GetChannelListStr(ChannelId:integer;
IncludeParent:boolean): String;`

ChannelId: Integer

An integer Id value for a channel.

IncludeParent: Boolean

If True, will include the parent channel Id in the returned list

Return Type: String

Description:

Returns a comma delimited list of channel_id's for sub-channels branching from the node specified by ChannelId. If IncludeParent = True then the specified channel_id is also included in the list.

Example: *This example extracts of the channelId's for the subchannels of the channel called "Home". It then displays the sub channel names and indents them according to their level in the tree.*

```
<$  
var lstChannels : TStringList;  
var i : integer;  
var ChannelLevel: Integer;  
lstChannels := TStringList.Create;  
  
function DisplayLevel(Level: Integer): string;  
var k: Integer;  
begin  
  for k := 1 to Level do  
    Result := Result + '++';  
end;  
  
try  
  lstChannels.CommaText := GetChannelListStr(GetChannelIdByName('Home'),  
                                           True);  
  for i := 0 to lstChannels.Count - 1 do  
    begin  
      QueryChannel('ch', StrToInt(lstChannels[i]));  
      ChannelLevel := QueryFieldAsInteger('ch.CHANNEL_LEVEL');  
      SendLn(DisplayLevel(ChannelLevel));  
      SendLn(QueryFieldAsInteger('ch.CHANNEL_LEVEL');$>  
             {'!ch.NAME%'}<br>  
<$end;  
finally  
  lstChannels.Free;  
end; //try-except  
$>
```

GetChannelMTitle function

Usage: **function** GetChannelMTitle(ChannelId: Integer): **String**;

ChannelId: Integer
An integer Id value for a channel.

Return type: String;

Description:

Returns the Channel Name/Title defined by ChannelId. This function is intended to be used when creating multi-lingual sites where channel names need to be returned in different languages. The function depends on the channel having a main/index Content page. If the main/index content page exists then the function will return the title from this content page in the current active language, else it will return the name of the channel. Returns an empty string if the channel Id is invalid.

Example: *This example displays the multilingual name for the channel called "Home" in the current language.*

The name for the "Home" channel in the current language is:
<%=HomeChannelMultiLingualTitle\$>

GetChannelName function

Usage: **function** GetChannelName(ChannelId: Integer): **String**;

ChannelId: Integer
An integer Id value for a channel.

Return type: String

Description:

Returns a channel name based on the ChannelId. Returns an empty string if the channel Id is invalid.

Example: *This example returns the name of the channel with Id = 1*

```
<$var ChannelName: string;  
  ChannelName := GetChannelName(1);$>  
Channel name = <%=Channelname$>
```

GetCurrChannelId function

Usage: `function GetCurrChannelId: Integer;`

Return type: `Integer;`

Description:

Returns the current ChannelId

Example: *This example uses the function "GetCurrChannelId" to display information about the current channel.*

```
<$var CurrentChannelName, CurrentChannelLink: String;  
    CurrentChannelName := GetChannelName(GetCurrChannelId);  
    CurrentChannelLink := GetChannelLink(GetCurrChannelId);$>  
  
<a href="<%=CurrentChannelLink%>"><%=CurrentChannelName%></a>
```

GetCurrChannelLevel function

Usage: `function GetCurrChannelLevel: Integer;`

Return type: `Integer;`

Description:

Returns the level in the main tree of the current channel.

Example: This example displays which level the current channel is in in the **Tree**.

```
<$var CurrChannelLevel: Integer;  
    CurrChannelLevel := GetCurrChannelLevel;$>
```

```
The current channel is at level: <%=CurrChannelLevel%>
```

GetCurrChannelName function

Usage: `function GetCurrChannelName: String;`

Return type: `string;`

Description:

Returns the name of the current channel.

Example: *This example returns the name of the current channel.*

```
The current channel name is: <b><%=GetCurrChannelName%></b>
```

GetCurrChannelParentId function

Usage: `function GetCurrChannelParentId: Integer;`

Return type: Integer;

Description:

Returns the Id of parent of the current channel.

Example: *This example displays the parent of the current channel.*

```
The current channel's parent is:  
<b><%=GetChannelName(GetCurrChannelParentId)%></b>
```

GetFullChannelPagePath function

Usage: `function GetFullChannelPagePath(ChannelId: Integer): String;`

```
ChannelId: Integer  
An integer Id value for a channel
```

Return type: String;

Description:

Returns the full channel link/path + default page name (such as 'index.htm') defined by ChannelId. This function is equivalent to the GetChannelLink() function. If an invalid ChannelId is specified then the output directory is returned.

Example: *This example displays a link to the "Home" channel.*

```
<a href=  
" <%=GetFullChannelPagePath(GetChannelIdByName('Home'))%>">Home  
</a>
```

GetFullChannelPath function

Usage: `function GetFullChannelPath(ChannelId: Integer): String;`

```
ChannelId: Integer  
An integer Id value for a channel
```

Return type: String;

Description:

Returns the full channel link/path (excluding the default page name, such as 'index.htm') defined by ChannelId. Returns an empty string if an invalid ChannelId is specified.

Example: *This example displays a link to the current channel using the channels full path.*

```
<a href=" <%=GetFullChannelPath(GetCurrChannelId)%>">  
 <%=GetCurrChannelName%></a>
```

GetStartChannelId function

Usage: `function GetStartChannelId: Integer;`

Return type: `Integer;`

Description:

Returns the Start ChannelId (i.e. Home Page).

Example: This example displays the name of the channel assigned as the "StartChannel".

```
<a href="<%=GetHomePath%>"><%=GetChannelName(GetStartChannelId)%></a>
```

GetHomePath function

Usage: `function GetHomePath: String;`

Return type: `String;`

Description:

Returns the full channel link/path + default page name (such as 'index.htm') for the Home Page / Start Channel.

Example:

```
<a href="<%=GetHomePath%>"><%=GetChannelName(GetStartChannelId)%></a>
```

GetMainParentChannelId function

Usage: `function GetMainParentChannelId(ChannelId: Integer;
StopAtChannelLevel: Integer; StopAtASiteStart: Boolean): Integer;`

ChannelId: Integer

An integer Id value for a channel

StopAtChannelLevel: Integer

Integer value specifying where in the channel tree level to stop

StopAtASiteStart: Boolean

A Boolean value indicating to stop at the site start channel

Return Type: Integer;

Description:

Returns the root/main parent ChannelId for a specified ChannelId, usually used with StopAtChannelLevel > -1 or StopAtASiteStart = True. Use value of StopAtChannelLevel = -1 if the function should not stop at a specific channel level else set StopAtChannelLevel to the level to stop at.

Example: The following example assumes the following channel structure, note channel level is in parenthesis:

```
Home (1)
  --News (2)
  ----Local (3)
  -----East Coast (4)
  -----West Coast (4)
  ----European (3)
  ----Asian (3)
  ----Rest of the World (3)
```

*In this example we want to return the parent channelId for the main parent channel of the "East Coast" channel, **but** we want to stop at level 2, i.e. the "News" channel.*

The Main Parent ChannelId (2 levels up) for the "East Coast" channel is:
`<%=GetMainParentChannelId(GetChannelIdByName('East Coast'), 2, False)%>`

GetMainParentChannelName function

Usage: function GetMainParentChannelName(ChannelId: Integer;
StopAtChannelLevel: Integer;
StopAtASiteStart: Boolean): **String**;

ChannelId: Integer

An integer Id value for a channel

StopAtChannelLevel: Integer

Integer value specifying where in the channel tree level to stop

StopAtASiteStart: Boolean

A Boolean value indicating to stop at the site start channel

Return Type: String;

Description:

Returns the root/main parent Channel Name for a specified ChannelId, usually used with StopAtChannelLevel > -1 or StopAtASiteStart = True. Use value of StopAtChannelLevel = -1 if the function should not stop at a specific channel level else set StopAtChannelLevel to the level to stop at.

Example: The following example assumes the following channel structure, note channel level is in parenthesis:

```
Home (1)
  --News (2)
  ----Local (3)
  -----East Coast (4)
  -----West Coast (4)
  ----European (3)
  ----Asian (3)
  ----Rest of the World (3)
```

*In this example we want to return the parent channel name for the main parent channel of the "East Coast" channel, **but** we want to stop at level 3, i.e. the "Local" channel.*

The Main Parent Channel Name (1 level up) for the "West Coast" channel is:
<%=GetMainParentChannelName(GetChannelIdByName('West Coast'), 3, False)%>

2. Content (Article) Related Functions/Procedures:

This category of functions deal with content related manipulation.

GetAbsContentLink function

Usage: **function** GetAbsContentLink(ContentId:integer): **String**

ContentId: Integer
An integer Id value for a content item

Return type: String

Description:

Returns the absolute content link/path defined by ContentId relative to the root path defined by AbsLinkPrefix (default = '').

Example:

```
Click <a href="<%=GetAbsContentLink(1)%>"> here </a> to read more about this article
```

GetContent function

Usage: **function** GetContent(ContentId: Integer): **String;**

ContentId: Integer
An integer Id value for a content item

Return type: String

Description:

Returns a content item with an Id equal to "ContentId".

Example: *This example displays the content(article) with an Id of "1".*

```
<div class="content">
  <%=GetContent(1)%>
</div>
```

GetContentEx function

Usage: **function** GetContentEx(ContentId: Integer): **String;**

ContentId: Integer
An integer Id value for a content item

Return type: String

Description:

Returns the main content text defined by ContentId and executes/expands any script/macros contained within the content if applicable.

Example:

```
<div class="content">
  <%=GetContentEx(1)%>
</div>
```

GetContentByTitle function

Usage: **function** GetContentByTitle(ContentTitle: String): **String**;

ContentTitle: String

A string value representing the title of the article

Return Type: String

Description:

Returns a content item with its title field equal to "ContentTitle".

Example: *This example displays the first content(article) with a title of "Latest News".*

```
<%=GetContentByTitle('Latest News')$>
```

GetContentByTitleEx function

Usage: **function** GetContentByTitleEx(ContentTitle: String): **String**;

ContentTitle: String

A string value representing the title of the article

Return Type: String

Description:

Returns the main content text defined by ContentTitle and executes/expands any script/macros contained within the content if applicable. NOTE: If more than one content topic has the same title then will return the content text for the first matching content.

Example:

```
<%=GetContentByTitleEx('Latest News')$>
```

GetContentIdByGlobalRefName function

Usage: **function** GetContentIdByGlobalRefName(GlobalRefName: String): **String**;

GlobalRefName: String

A String value representing a Global Reference Name

Return Type: String

Description:

Returns the actual ContentId when passed a Content's Global Reference Name.

Example: *This example returns the content(article) defined with the global reference name of "ECMS_LINK_12".*

```
<a href="ECMS_LINK_12">  
<%=GetContentIdByGlobalRefName('ECMS_LINK_12')$></a>
```

GetContentImageSrc function

Usage: `function GetContentImageSrc(SeqNo: Integer; Border: Integer; Thumbnail: Boolean; WithLink: Boolean; OtherAttribs: String): String;`

SeqNo: Integer

The sequence number corresponding to the image to retrieve.

Border: Integer

The thickness of the border, 0 = no border

Thumbnail: Boolean

If True, then the function will return the thumbnail version of the image

WithLink: Boolean

Provides a link to the original image, if Thumbnail is True

OtherAttribs: String

Allows you to specify extra formatting for example class code etc

Return Type: String

Description:

Returns an image corresponding to the current content/article item, or full IMG tag, where SeqNo is the sequence number of the image/media to retrieve. A value of -1 for SeqNo will reference the first image/media.

Example: This function loops through the images assigned to the current article and then displays each one of the images on a newline.

```
<$var ImageCounter: Integer=0;$>
<$QueryContentMediaList('q',GetCurrContentId);$>
<FOREACH ds="q">
  <$Inc(ImageCounter,1);$>
  <%=GetContentImageSrc(ImageCounter, 0, False, True,
'class="mystyle"')$>
  <br>
</FOREACH>
```

GetContentImageSrc2 function

Usage: `function GetContentImageSrc2(Title: String; OtherAttribs: String): String;`

Title: String

The title(name) of the image

OtherAttribs: String

Allows you to specify extra formatting for example class code etc.

Return Type: String

Description:

Returns the current content/article's item associated local image/media source link, or full IMG tag referenced by 'Title'. Append the Title with '*' to return the full IMG tag, or with '**' to automatically create a thumbnail with a link to the full image, else only the source link is returned.

Example: *This function displays an image assigned to the current article. The image is retrieved by title.*

```
<%=GetContentImageSrc2('logo_cms', 'class="mystyle"')$>
```

GetContentImageSrc2ById function

Usage: `function GetContentImageSrc2ById(ContentId: Integer; Title: String; OtherAttribs: String): String;`

ContentId: Integer

The content Id of the content to search for.

Title: String

The title(name) of the image.

OtherAttribs: String

Allows you to specify extra formatting for example class code, etc.

Return Type: String

Description:

Returns the specified content/article's item associated local image/media source link, or full IMG tag referenced by 'Title'. Append the Title with '*' to return the full IMG tag, or with '**' to automatically create a thumbnail with a link to the full image, else only the source link is returned.

Example: *This function displays an image assigned to the specified article. The image is retrieved by title.*

```
<%=GetContentImageSrc2ById(10, 'logo_cms', 'class="mystyle"')$>
```

GetContentImageSrcById function

Usage: `function GetContentImageSrcById(ContentId: Integer; SeqNo: Integer; Border: Integer; Thumbnail: Boolean; WithLink: Boolean; OtherAttribs: String): String;`

ContentId: Integer

The content/article Id

SeqNo: Integer;

The sequence number corresponding to the image to retrieve.

Border: Integer;

Specifies the thickness of the border, 0 = no border.

Thumbnail: Boolean;

If True will display the thumbnail version of the image

WithLink: Boolean;

Provides a link to the original image, if Thumbnail is True

OtherAttribs: String

Allows you to specify extra formatting for example class code etc

Return Type: String

Description:

Similar to **GetContentImageSrc**, except that you can specify the content/article via ContentId (or -1 for current content/article item), associated local image/media source link, or full IMG tag, where SeqNo is the sequence number of the image/media to retrieve. A value of -1 for SeqNo will reference the first image/media.

Example: *This function displays the first image assigned to the article with an Id of 17. The image is retrieved by name.*

```
<%=GetContentImageSrcById(17, 1, 0, False, True, 'class="mystyle"')$>
```

GetContentImageSrcLink function

Usage: `function GetContentImageSrcLink(Title: String): String;`

Title: String The title(name) of the image
--

Return Type: String

Description:

Returns the current Content's associated local image/media source link referenced by 'Title'

Example: *This function retrieves the link to an image assigned to the current article. The image is retrieved by name.*

<code><a href="<%=GetContentImageSrcLink('logo_cms')\$%">This is the link to the full size image</code>
--

GetContentLink function

Usage: **function** GetContentLink(ContentId: Integer): **String**;

ContentId: Integer
The content/article Id value

Return Type: String

Description:

Returns the full content link/path defined by ContentId

Example: *This example displays the link to an article.*

```
<a href="<%=GetContentLink(1)%>"><%=GetContentTitle(1)%></a>
```

GetContentLinkAdv function

Usage: **function** GetContentLinkAdv(ContentId: Integer; ChannelId: Integer; FileExt: String): **String**;

ContentId: Integer
The content/article Id

ChannelId: Integer
An integer Id value for a channel

FileExt: String
The extension to assign to the content link

Return Type: String

Description:

An alternate to **GetContentLink**, this function lets you specify the ChannelId and File Extension for the returned link.

Example:

```
<a href="<%=GetContentLinkAdv(1, GetChannelIdByName('Home'), '.asp')%>"><%=GetContentTitle(1)%></a>
```

GetContentPropValue function

Usage: **function** GetContentPropValue(KeyName: String): **String**;

KeyName: String
The name of the property value to retrieve

Return Type: String

Description:

Returns the value of a Content property. These properties are usually specified as user-defined fields on a template, and then filled-in when adding content associated with the template. This function allows to easily retrieve the value given the property name.

Example:

```
<div class="rightsidebar">
  <%=GetContentPropValue('Sidebar_html')%>
</div>
```

GetContentPropValueEx function

Usage: **function** GetContentPropValueEx(KeyName: String): **String**;

KeyName: String
The name of the property value to retrieve

Return Type: String

Description:

Returns the value of a Content property, and expands any macro's/script code contained within the content property. These properties are usually specified as user-defined fields on a template, and then filled-in when adding content associated with the template. This function allows to easily retrieve the value given the property name.

Example:

```
<div class="rightsidebar">
  <%=GetContentPropValueEx('Sidebar_html')%>
</div>
```

GetContentSummary function

Usage: **function** GetContentSummary(ContentId: Integer): **String**;

ContentId: Integer
The content/article Id value

Return Type: String

Description:

This function returns the summary text for a content/article item with Id = ContentId

Example:

```
<div class="Summary">
  <%=GetContentSummary(1)%>
</div>
```

GetContentSummaryEx function

Usage: `function GetContentSummary(ContentId: Integer): String;`

ContentId: Integer The content/article Id value

Return Type: String

Description:

Returns the content summary text defined by ContentId and executes/expands any script/macros contained within the content summary if applicable.

Example:

<pre><div class="Summary"> <%=GetContentSummaryEx(1)%> </div></pre>

GetContentSummaryByTitle function

Usage: **function** GetContentSummaryByTitle(ContentTitle: String): **String**;

ContentTitle: String
The title(name) of the content/article item

Return Type: String

Description:

Works like GetContentSummary, except that the functions returns the summary text for a content/article item with Title = ContentTitle

Example:

```
<div class="Summary">  
  <%=GetContentSummaryByTitle('HomePage_Info')%>  
</div>
```

GetContentSummaryByTitleEx function

Usage: **function** GetContentSummaryByTitleEx(ContentTitle: String): **String**;

ContentTitle: String
The title(name) of the content/article item

Return Type: String

Description:

Returns the content summary text defined by ContentTitle and executes/expands any script/macros contained within the content summary if applicable.

Note: If more than one content topic has the same title then will return the content summary text for the first matching content.

Example:

```
<div class="Summary">  
  <%=GetContentSummaryByTitleEx('HomePage_Info')%>  
</div>
```

GetContentTitle function

Usage: **function** GetContentTitle(ContentId: Integer): **String**;

ContentId: Integer
The content/article Id value

Return Type: String

Description:

Returns the title for a content/article item with Id = ContentId

Example:

```
<div class="Title">  
  <%=GetContentTitle(1)%>  
</div>
```

GetCurrContentId function

Usage: `function GetCurrContentId: Integer;`

Return Type: `Integer;`

Description:

Returns the Content Id for the current page.

Example:

```
<div class="Title">  
  <%=GetContentTitle(GetCurrContentId)%>  
</div>
```

3. Template Related Functions/Procedures:

This group of functions deal with template manipulation and retrieval.

GetTemplate function

Usage: **function** GetTemplate(TemplateId: Integer): **String**;

TemplateId: Integer
The template value Id

Return Type: String

Description:

Returns the text representing a templates code. If the template contains script code, then the template is first executed and the template text is then returned.

Example: *This example retrieves a template called with an Id of "17".*

```
<div class="Menu">  
  <%=GetTemplate(17)%>  
</div>
```

GetTemplateByName function

Usage: **function** GetTemplateByName(TemplateName: String): **String**;

TemplateName: String
A template value specified by Name = TemplateName

Return Type: String

Description:

Functions the same as **GetTemplate**, except that the template name is used instead of the Id.

Example: *This example retrieves a template called: "Menu"*

```
<div class="Menu">  
  <%=GetTemplateByName('Menu')%>  
</div>
```

GetTemplateImageSrc function

Usage: `function GetTemplateImageSrc(SeqNo: Integer; Border: Integer; Thumbnail: Boolean; WithLink: Boolean; OtherAttribs: String): String;`

SeqNo: Integer

The sequence number corresponding to the image to retrieve.

Border: Integer

The thickness of the border, 0 = no border

Thumbnail: Boolean

If True, then the function will return the thumbnail version of the image

WithLink: Boolean

Provides a link to the original image, if Thumbnail is True

OtherAttribs: String

Allows you to specify extra formatting for example class code etc

Return Type: String

Description:

Returns an image corresponding to the current template, or full IMG tag, where SeqNo is the sequence number of the image/media to retrieve. A value of -1 for SeqNo will reference the first image/media.

Example:

```
<%=GetTemplateImageSrc(1, 0, False, True, '')$>
```

GetTemplateImageSrc2 function

Usage: `function GetTemplateImageSrc2(Title: String; OtherAttribs: String): String;`

Title: String

The title(name) of the image

OtherAttribs: String

Allows you to specify extra formatting for example class code etc.

Return Type: String

Description:

Returns the current template image/media item source link, or full IMG tag referenced by 'Title'. Append the Title with '*' to return the full IMG tag, or with '**' to automatically create a thumbnail with a link to the full image, else only the source link is returned.

Example:

```
<%=GetTemplateImageSrc2('arrows', 'class="icl"')$>
```

GetTemplateImageSrc2ById function

Usage: `function GetTemplateImageSrc2ById(TemplateId: Integer; Title: String; OtherAttribs: String): String;`

TemplateId: Integer

The Id of the template to search for.

Title: String

The title(name) of the image.

OtherAttribs: String

Allows you to specify extra formatting for example class code, etc.

Return Type: String

Description:

Returns the specified template image/media item source link, or full IMG tag referenced by 'Title'. Append the Title with '*' to return the full IMG tag, or with '**' to automatically create a thumbnail with a link to the full image, else only the source link is returned.

Example:

```
<%=GetTemplateImageSrc2ById(5, 'arrows', 'class="icl"')$>
```

GetTemplateImageSrcById function

Usage: `function GetTemplateImageSrcById(TemplateId: Integer; SeqNo: Integer; Border: Integer; Thumbnail: Boolean; WithLink: Boolean; OtherAttribs: String): String;`

TemplateId: Integer

The template Id

SeqNo: Integer;

The sequence number corresponding to the image to retrieve.

Border: Integer;

Specifies the thickness of the border, 0 = no border.

Thumbnail: Boolean;

If True will display the thumbnail version of the image

WithLink: Boolean;

Provides a link to the original image, if Thumbnail is True

OtherAttribs: String

Allows you to specify extra formatting for example class code etc

Return Type: String

Description:

Similar to **GetTemplateImageSrc**, except that you can specify the template via TemplateId (or -1 for current template), associated local image/media source link, or full IMG tag, where SeqNo is the sequence number of the image/media to retrieve. A value of -1 for SeqNo will reference the first image/media.

Example:

```
<%=GetTemplateImageSrcById(1, 1, 0, False, True, 'class="mystyle"')$>
```

GetTemplateImageSrcLink function

Usage: `function GetTemplateImageSrcLink(Title: String): String;`

Title: String The title(name) of the image
--

Return Type: String

Description:

Returns the current Template's associated local image/media source link referenced by 'Title'

Example: *This function retrieves the link to an image assigned to the current template. The image is retrieved by name.*

<code><a href="<%=GetTemplateImageSrcLink('logo_cms')%>">This is the link to the full size image</code>

4. Query Related Functions/Procedures:

This group of functions deals querying functions. These are functions that query the database, and populate datasources. They are usually used in conjunction with `<FOREACH>` statements.

QueryChannel function

Usage: `function QueryChannel(QueryName: String; ChannelId: Integer): Boolean;`

<p>QueryName: String A string value that will identify this query result</p> <p>ChannelId: Integer An integer Id value for a channel.</p>

Return type: Boolean

Description:

Queries the channel record specified by ChannelId. Passing a value of -1 for ChannelId will assume the current channel. This function returns False if no record found, else returns True.

Example:

<pre><\$QueryChannel('qryChannel', GetChannelIdByName('Home'));\$> {%!qryChannel.Name%}</pre>

QueryChannelList function

Usage: `function QueryChannelList(QueryName: String; ChannelId: Integer): Boolean;`

<p>QueryName: String A string value that will identify this query result</p> <p>ChannelId: Integer An integer Id value for a channel</p>
--

Return type: Boolean

Description:

Queries for a list of sub-channels for a specified ChannelId. Passing a value of -1 for ChannelId will assume the current channel. Function returns False if no sub-channel records found, else returns True.

Example:

<pre><\$QueryChannelList('qryChannelList', GetChannelIdByName('Home'));\$> <FOREACH ds="qryChannelList"> {%!qryChannelList.Name%}
 </FOREACH></pre>

QueryChannelListAdv function

Usage: `function QueryChannelListAdv(QueryName: String; ChannelId: Integer; IncludeHome, IncludeParent: Boolean; ChannelGroup: String; SortField: String; Descending: Boolean; LocateId: Integer; SkipRows: Integer): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelId: Integer

An integer Id value for a channel

IncludeHome: Boolean

A value of True indicates that the 'Home' channel should be included

IncludeParent: Boolean

A value of True indicates that the 'Parent' channel should be included (ChannelId)

ChannelGroup: String

The function will retrieve all channels contained in this 'Group'

SortField: String

Optionally sort by this field.

Descending: Boolean

If True, then the result will be sorted descending

LocateId: Integer

If specified, will start with the first channel with an Id matching LocateId, else specify 0

SkipRows: Integer

Skips the specified no. of rows from the beginning of the query, else specify 0

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of sub-channels for a specified ChannelId. Similar to the QueryChannelList() function, but this function has a few more advanced options. Passing a value of -1 for ChannelId will assume the current channel.

This function returns False if no sub-channel records found, else returns True.

Example: *This example loops through the project channel list and sorts the results by "NAME".*

```
<$QueryChannelListAdv('ch', GetStartChannelId, True, True, '', 'NAME',  
False, -1, -1);$>  
<FOREACH ds="ch">  
  <div class="nav">  
    <%=QueryFieldAsString('ch.NAME')$><br>  
  </div>  
</FOREACH>
```

QueryChannelListAdv2 function

Usage: `function QueryChannelListAdv2(QueryName: String; ChannelId: Integer; IncludeHome, IncludeParent: Boolean; ChannelGroup: String; SortField: String; Descending: Boolean; LocateId: Integer; SkipRows: Integer; ValidChannelList: String; ReturnFields: String; WhereFilter: String): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelId: Integer

An integer Id value for a channel

IncludeHome: Boolean

A value of True indicates that the 'Home' channel should be included

IncludeParent: Boolean

A value of True indicates that the 'Parent' channel should be included (ChannelId)

ChannelGroup: String

The function will retrieve all channels contained in this 'Group'

SortField: String

Optionally sort by this field.

Descending: Boolean

If True, then the result will be sorted descending

LocateId: Integer

If specified, will start with the first channel with an Id matching LocateId, else specify 0

SkipRows: Integer

Skips the specified no. of rows from the beginning of the query, else specify 0

ValidChannelList: String

List of channels to limit the query to, example '1,3,5' If blank then current ChannelId will be used

ReturnFields: String

The list of fields to return, example: 'CONTENT_ID, CONTENT_GROUP'. If blank then all main fields will be returned

WhereFilter: String

To filter the result, example: '(CREATED_DATE >= "01/01/2003")' If blank, then all records will be returned.

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of sub-channels for a specified ChannelId. Similar to the QueryChannelList() function, but this function has a few more advanced options. Passing a value of -1 for ChannelId will assume the current channel.

SortField is the field to sort on, but can also be an empty string for default sort order. ValidChannelList can be blank, meaning to use channel specified by ChannelId, else process the comma delimited list of channel id's specified by ValidChannelList.

This parameter can also be '*' meaning to process all channels, or prefix with 'group:' and then the channel group to process. ReturnFields lists the fields to return, if blank, indicates that all fields must be returned.

Often query performance can be optimized by just returning the fields needed.

WhereFilter filters the result of the query and lets you restrict the dataset result returned by the query. Once again, this can optimize performance.

This function returns False if no sub-channel records found, else returns True.

Example: This example returns all channels starting with the character "N", and sorts the results by "Name".

```
<${QueryChannelListAdv2('ch', GetStartChannelId, True, True, '', 'NAME',
False, 0, 0, '', '', 'NAME like 'N%' ' ')};>
<FOREACH ds="ch">
  <div class="nav">
    <${QueryFieldAsString('ch.NAME')}><br>
  </div>
</FOREACH>
```

QueryChannelListDesc function

Usage: function QueryChannelListDesc(QueryName: String; ChannelId: Integer): Boolean;

<p>QueryName: String A string value that will identify this query result</p> <p>ChannelId: Integer An integer Id value for a channel</p>
--

Return type: Boolean

Description:

Works exactly like QueryChannelList, except that the result is sorted in descending order. Passing a value of -1 for ChannelId will assume the current channel. Returns False if no sub-channel records found, else returns True.

Example: This example retrieves all of the channels under "Home" and displays them in descending order.

```
<${QueryChannelListDesc('qryChannelList', GetChannelIdByName('Home'))}>
<FOREACH ds="qryChannelList">
  <a href="{!qryChannelList._chlink%}">{!qryChannelList.Name%}</a>
  <br>
</FOREACH>
```

QueryChannelListNext function

Usage: **function** QueryChannelListNext(QueryName: String): Boolean;

QueryName: String

A string value that will identify this query result

Return type: Boolean

Description:

Queries for the next list of sibling channels after the current channel. This function returns False if no next channel records found, else returns True. This function is useful in automatically building previous/next channel links.

Example:

```
<$if (QueryChannelListPrev('p')) then
  begin $>
    Previous Channel: <b>{%!p.NAME%}</b>
<$end;$>
<br><br>
<$if (QueryChannelListNext('n')) then
  begin $>
    Next channel: <b>{%!n.NAME%}</b>
<$end;$>
```

QueryChannelListPrev function

Usage: **function** QueryChannelListPrev(QueryName: String): Boolean;

QueryName: String

A string value that will identify this query result

Return type: Boolean

Description:

Queries for the previous list of sibling channels after the current channel. This function returns False if no next channel records found, else returns True. This function is useful in automatically building previous/next channel links.

Example:

See *previous example*.

QueryChannelMediaList function

Usage: `function QueryChannelMediaList(QueryName: String; ChannelId: Integer): Boolean;`

<p>QueryName: String A string value that will identify this query result</p> <p>ChannelId: Integer An integer Id value for a channel</p>
--

Return Type: Boolean

Description:

A predefined custom query function which queries for the list of media/images associated with the channel defined by ChannelId. Passing a value of -1 for ChannelId will assume the current channel. This function returns False if no channel media records found, else returns True.

Example:

<pre><\$QueryChannelMediaList('c', GetChannelIdByName('Home'));> var I: Integer=0;\$> <FOREACH ds="c"> <\$Inc(I, 1);\$> <%=GetChannelImageSrcById(QueryFieldAsInteger('c.Channel_Id'), I, 0, False, False, '')\$>
 </FOREACH></pre>
--

QueryContent function

Usage: `function QueryContent(QueryName: String; ContentId: Integer): Boolean;`

<p>QueryName: String A string value that will identify this query result</p> <p>ContentId: Integer An integer Id value for a content/article item</p>

Return Type: Boolean

Description:

Queries the database to retrieve a content/article item with an Id specified by ContentId. If no content/article item is found with this Id, then function returns False, True otherwise.

Example:

<pre><div class="text"> <\$QueryContent('c', 1);\$> Article Title: {%!c.TITLE%}

 Article Text: {%!c._CONTENT_TEXT_%}
 </div></pre>
--

QueryContentList function

Usage: `function QueryContentList(QueryName: String; ChannelId: Integer): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelId: Integer

An integer Id value for a channel/folder

Return Type: Boolean

Description:

Queries the database to retrieve a content/article list for a channel with an Id specified by ChannelId. If no channel is found with this Id, then function returns False, True otherwise.

Example:

```
<div class="text">
  <${QueryContentList('c', GetChannelIdByName('Home'))};$>
  <FOREACH ds="c">
    <b>Article Title:</b> {!c.TITLE!}<br><br>
  </FOREACH>
</div>
```

QueryContentListAdv function

Usage: `function QueryContentListAdv(QueryName: String; ChannelOrContentId: Integer; IncludeMainIndexContent: Boolean; DoSubContent: Boolean; ContentGroup: String; SortField: String; Descending: Boolean; LocateId: Integer; SkipRows: Integer): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelOrContentId: Integer

An integer Id value for a channel or content/article item

IncludeMainIndexContent: Boolean

If True, then the query will return the channel's index page as well

DoSubContent: Boolean

A value of True indicates that any subcontent will also be retrieved

ContentGroup: String

The function will retrieve all channels contained in this 'Group'

SortField: String

Optionally sort by this field.

Descending: Boolean

If True, then the result will be sorted descending

LocateId: Integer

If specified, will start with the first channel with an Id matching LocateId, else specify 0

SkipRows: Integer

Skips the specified no. of rows from the beginning of the query, else specify 0

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId. Similar to the QueryContentList() function, but this function has a few more advanced options.

Passing a value of -1 for ChannelId will assume the current channel. The value for IncludeMainIndexContent is generally False, but a value of True will allow to also return the main/index content record linked to the channel (normally this content is not displayed in lists, but this parameter allows to override that behaviour).

SortField is the field to sort on, but can also be an empty string for default sort order.

This function returns False if no content records found, else returns True.

Example: *This example loops through and displays all of the content(articles) for the "News" channel. The articles are then sorted by "TITLE".*

```
<$QueryContentListAdv('co', GetChannelIdByName('News'), False, False, '', 'TITLE', False, -1, -1);$>
<FOREACH ds="co">
  <div class="articles">
    <a href="<%=GetContentLink(QueryFieldAsInteger('co.CONTENT_ID'))$%">
      <%=QueryFieldAsString('co.TITLE')$%>
    </a><br>
  </div>
</FOREACH>
```

QueryContentListAdv2 function

Usage: `function QueryContentListAdv2(QueryName: String; ChannelOrContentId: Integer; IncludeMainIndexContent: Boolean; DoSubContent: Boolean; ContentGroup: String; SortField: String; Descending: Boolean; LocateId: Integer; SkipRows: Integer; ValidChannelList: String; ReturnFields: String; WhereFilter: String): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelOrContentId: Integer

An integer Id value for a channel or content/article item

IncludeMainIndexContent: Boolean

If True, then the query will return the channel's index page as well

DoSubContent: Boolean

A value of True indicates that any subcontent will also be retrieved

ContentGroup: String

The function will retrieve all channels contained in this 'Group'

SortField: String

Optionally sort by this field.

Descending: Boolean

If True, then the result will be sorted descending

LocateId: Integer

If specified, will start with the first channel with an Id matching LocateId, else specify 0

SkipRows: Integer

Skips the specified no. of rows from the beginning of the query, else specify 0

ValidChannelList: String

List of channels to limit the query to, example '1,3,5' If blank then current ChannelId will be used

ReturnFields: String

The list of fields to return, example: 'CONTENT_ID, CONTENT_GROUP'. If blank then all main fields will be returned

WhereFilter: String

To filter the result, example: '(CREATED_DATE >= "01/01/2003")' If blank, then all records will be returned.

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId. Similar to the QueryContentList() function, but this function has a few more advanced options.

Passing a value of -1 for ChannelId will assume the current channel. The value for IncludeMainIndexContent is generally False, but a value of True will allow to also return the main/index content record linked to the channel (normally this content is not displayed in lists, but this parameter allows to override that behaviour).

SortField is the field to sort on, but can also be an empty string for default sort order. ValidChannelList can be blank, meaning to use channel specified by ChannelOrContentId, else process the comma delimited list of channel id's specified by ValidChannelList.

This parameter can also be '*' meaning to process all channels, or prefix with 'group:' and then the channel group to process.

ReturnFields lists the fields to return, if blank, indicates that all fields must be returned.

Often query performance can be optimized by just returning the fields needed.

WhereFilter filters the result of the query and lets you restrict the dataset result returned by the query. Once again, this can optimize performance.

This function returns False if no content records found, else returns True.

Example: This example loops through and displays all of the content(articles) for the "News" channel. The articles are then sorted by "TITLE". Also only article starting with the character "B" will be returned. Note you might need to prefix the field name with 'c.', as in 'c.TITLE like 'B%' ' when using a "like" statement in the "Where" clause.

```
<$QueryContentListAdv2('co', GetChannelIdByName('News'), False, False,
'', 'TITLE', False, 0, 0, '', '', 'c.TITLE like 'B%' ');$>
<FOREACH ds="co">
  <div class="articles">
    <a href="<%=GetContentLink(QueryFieldAsInteger('co.CONTENT_ID'))$>">
      <%=QueryFieldAsString('co.TITLE')$>
    </a><br>
  </div>
</FOREACH>
```

QueryContentListDesc function

Usage: `function QueryContentListDesc(QueryName: String; ChannelId: Integer): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelId: Integer

An integer Id value for a channel/folder

Return Type: Boolean

Description:

This function works exactly like `QueryContentList`, except that the content list is sorted descending.

Example: This example will return all of the content(articles) for the "News" channel and display the results in descending order.

```
<div class="text">
  <$QueryContentListDesc('c', GetChannelIdByName('News'));$>
  <FOREACH ds="c">
    <b>Article Title:</b> {%!c.TITLE%!}<br><br>
  </FOREACH>
</div>
```

QueryContentListIncl function

Usage: **function** QueryContentListIncl(QueryName: String; ChannelId: Integer): **Boolean**;

QueryName: String

A string value that will identify this query result

ChannelId: Integer

An integer Id value for a channel/folder

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId and also includes main/index content record of channel (if assigned). Passing a value of -1 for ChannelId will assume the current channel. This function returns False if no content records found, else returns True.

Example: *This example displays all of the content(articles) for the "News" Channel, including the "Index" page.*

```
<div class="text">
  <${QueryContentListIncl('c', GetChannelIdByName('News'))};$>
  <FOREACH ds="c">
    Article Title: <a href="{!c._link%}">{!c.TITLE%}</a>
    <br><br>
  </FOREACH>
</div>
```

QueryContentListInclDesc function

Usage: **function** QueryContentListInclDesc(QueryName:string; ChannelId:integer):**boolean**;

QueryName: String

A string value that will identify this query result

ChannelId: Integer

An integer Id value for a channel/folder

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId in descending order and also includes main/index content record of channel (if assigned). Passing a value of -1 for ChannelId will assume the current channel. This function returns False if no content records found, else returns True.

Example: *This example displays all of the content(articles) for the "News" Channel, including the "Index" page, and sorts the results in descending order.*

```
<div class="text">
  <${QueryContentListInclDesc('c', GetChannelIdByName('News'))};$>
  <FOREACH ds="c">
    Article Title: <a href="{!c._link%}">{!c.TITLE%}</a>
    <br><br>
  </FOREACH>
</div>
```

QueryContentListInclNext function

Usage: `function QueryContentListInclNext(QueryName:string):Boolean;`

QueryName: String

A string value that will identify this query result

Return Type: Boolean

Description:

A predefined custom query function which queries for the next list of sibling content after the current content (assumes current channel) and also includes main/index content record of channel (if assigned). This function returns False if no next content records found, else returns True. This function is useful in automatically building previous/next content links.

Example: *This example allows you to scroll between the previous and next articles in relation to the current article. The index page for the channel will also be included.*

```
<$if (QueryContentListInclPrev('p')) then
begin $>
Previous Article:&nbsp;
<a href="{!p._link%}">
  {!p.TITLE%}
</a>
<br>
<$end;$>

<$if (QueryContentListInclNext('n')) then
begin $>
Next Article:&nbsp;
<a href="{!n._link%}">
  {!n.TITLE%}
</a>
<br>
<$end;$>
```

QueryContentListInclPrev function

Usage: `function QueryContentListInclPrev(QueryName:string):Boolean;`

QueryName: String

A string value that will identify this query result

Return Type: Boolean

Description:

A predefined custom query function which queries for the previous list of sibling content before the current content (assumes current channel) and also includes main/index content record of channel (if assigned). This function returns False if no previous content records found, else returns True. This function is useful in automatically building previous/next content links.

Example: *This example allows you to scroll between the previous and next articles in relation to the current article. The index page for the channel will also be included.*

```
<$if (QueryContentListInclPrev('p')) then
begin $>
Previous Article:&nbsp;
<a href="{!p._link%}">
  {!p.TITLE%}
</a>
<br>
<$end;$>

<$if (QueryContentListInclNext('n')) then
begin $>
Next Article:&nbsp;
<a href="{!n._link%}">
  {!n.TITLE%}
</a>
<br>
<$end;$>
```

QueryContentListLastModified function

Usage: `function QueryContentListLastModified(QueryName: String; ChannelId: Integer): Boolean;`

<p>QueryName: String A string value that will identify this query result</p> <p>ChannelId: Integer An integer Id value for a channel/folder</p>

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId ordered by the date/time the content was last modified. Passing a value of -1 for ChannelId will assume the current channel. This function returns False if no content records found, else returns True.

Example:

```
<$QueryContentListLastModified('c', GetChannelIdByName('News'));$>
<FOREACH ds="c">
  <a href="{!c._link%}">{!c.TITLE%}</a>&nbsp;&nbsp;&nbsp;
  Published on: <${=FormatDateTime('dd mmmm yyyy hh:mm',
    QueryFieldAsDateTime('c.LAST_MODIFIED'))}$>
  <br>
</FOREACH>
```

QueryContentListLastModifiedGlobal function

Usage: `function QueryContentListLastModifiedGlobal(QueryName: String; ChannelIdList: String; ChannelGroup: String): Boolean;`

<p>QueryName: String A string value that will identify this query result</p> <p>ChannelIdList: String A comma delimited string list of channel Id's</p> <p>ChannelGroup: String A string value representing a channel group</p>
--

Return type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified list of ChannelId's (comma delimited) or by channels matching the specified ChannelGroup. Output is ordered by date/time the content was last modified.

Passing a value of blank for ChannelIdList will check all channels. This function returns False if no content records found, else returns True.

Example:

```
<$QueryContentListLastModifiedGlobal('c', '2,4,6', '');$>
<FOREACH ds="c">
  <a href="{!c._link%}">{!c.TITLE%}</a>&nbsp;&nbsp;&nbsp;
  Published on: <${=FormatDateTime('dd mmmm yyyy hh:mm',
    QueryFieldAsDateTime('c.LAST_MODIFIED'))}$>
  <br>
</FOREACH>
```

QueryContentListLatest function

Usage: `function QueryContentListLatest(QueryName: String; ChannelId: Integer): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelId: Integer

An integer Id value for a channel/folder

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId ordered by sequence that the content was added. Passing a value of -1 for ChannelId will assume the current channel. This function returns False if no content records found, else returns True.

Example: *This example displays the title plus the link to the latest article for the channel called "News".*

```
<div class="Latest">
  <${QueryContentListLatest('c', GetChannelIdByName('News'))};$>
  The latest article for this channel:
  <a href="{!c._link%}">{!c.TITLE%}</a>
</div>
```

QueryContentListLatestGlobal function

Usage: `function QueryContentListLatestGlobal(QueryName: String; ChannelIdList: String; ChannelGroup: String): Boolean;`

QueryName: String

A string value that will identify this query result

ChannelIdList: String

A comma delimited string list of channel Id's

ChannelGroup: String

A string value representing a channel group

Return type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified list of ChannelId's (comma delimited) or by channels matching the specified ChannelGroup. Output is sequence that the content was added. Passing a value of blank for ChannelIdList will check all channels. This function returns False if no content records found, else returns True.

Example: *This example displays the title plus the link to the latest article for the entire site. If we wanted to check for only certain channels, then we would fill in a comma delimited list of channelId's in the "ChannelIdList" property.*

```
<div class="Latest">
  <${QueryContentListLatestGlobal('c', '', '')};$>
  The latest article on the site is:
  <a href="{!c._link%}">{!c.TITLE%}</a>
</div>
```

QueryContentListNext function

Usage: `function QueryContentListNext(QueryName: String): Boolean;`

QueryName: String

A string value that will identify this query result

Return Type: Boolean

Description:

A predefined custom query function which queries for the next list of sibling content after the current content (assumes current channel). This function returns False if no next content records found, else returns True. This function is useful in automatically building previous/next content links.

Example: *This example allows you to scroll between the previous and next articles in relation to the current article.*

```
<$if (QueryContentListPrev('p')) then
begin $>
Previous Article:&nbsp;
<a href="{%!p._link%}">
  {%!p.TITLE%}
</a>
<br>
<$end;$>

<$if (QueryContentListNext('n')) then
begin $>
Next Article:&nbsp;
<a href="{%!n._link%}">
  {%!n.TITLE%}
</a>
<br>
<$end;$>
```

QueryContentListPrev function

Usage: `function QueryContentListPrev(QueryName: String): Boolean;`

QueryName: String

A string value that will identify this query result

Return Type: Boolean

Description:

A predefined custom query function which queries for the previous list of sibling content before the current content (assumes current channel). This function returns False if no previous content records found, else returns True. This function is useful in automatically building previous/next content links.

Example: *This example allows you to scroll between the previous and next articles in relation to the current article.*

```
<$if (QueryContentListPrev('p')) then
begin $>
Previous Article:&nbsp;
<a href="{%!p._link%}">
  {%!p.TITLE%}
</a>
<br>
<$end;$>

<$if (QueryContentListNext('n')) then
begin $>
Next Article:&nbsp;
<a href="{%!n._link%}">
  {%!n.TITLE%}
</a>
<br>
<$end;$>
```

QueryContentListSorted function

Usage: `function QueryContentListSorted(QueryName:string; ChannelId:integer; IncludeMainIndexContent:boolean; SortField:string):boolean;`

QueryName: string

A string value that will identify this query result

ChannelId: integer

An integer Id value for a channel/folder

IncludeMainIndexContent :Boolean

If True, will include the main/index page for the channel

SortField :string

A String value which indicates which field to sort on

WhereFilter: string

To filter the result, example: '(CREATED_DATE >= "01/01/2003")' If blank, then all records will be returned.

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId and sorts results on field specified by SortField. SortField can also be an empty string for default sort order. Set IncludeMainIndexContent to True to include main/index content record assigned to channel, else set to False. Passing a value of -1 for ChannelId will assume the current channel.

WhereFilter filters the result of the query and lets you restrict the dataset result returned by the query. Once again, this can optimize performance. This function returns False if no content records found, else returns True.

Example: *This example will return all the articles for the current channel and sort the results by "Title". Also, because of the "WhereFilter" only articles assigned to the language with LanguageID = 1 (i.e. the primary language.)*

```
<div class="text">
  <${QueryContentListSorted('c', -1, True, 'Title', 'Language_Id=1')};>
  <FOREACH ds="c">
    <b>Article Title:</b> {%!c.TITLE%}<br><br>
  </FOREACH>
</div>
```

QueryContentListSortedDesc function

Usage: `function QueryContentListSortedDesc(QueryName:string;
ChannelId:integer; IncludeMainIndexContent:boolean;
SortField:string):boolean;`

QueryName: string

A string value that will identify this query result

ChannelId: integer

An integer Id value for a channel/folder

IncludeMainIndexContent :Boolean

If True, will include the main/index page for the channel

SortField :string

A String value which indicates which field to sort on

WhereFilter: string

To filter the result, example: '(CREATED_DATE >= "01/01/2003")' If blank, then all records will be returned.

Return Type: Boolean

Description:

A predefined custom query function which queries for a list of content for a specified ChannelId and sorts results on field specified by SortField in descending order. SortField can also be an empty string for default sort order. Set IncludeMainIndexContent to True to include main/index content record assigned to channel, else set to False. Passing a value of -1 for ChannelId will assume the current channel.

WhereFilter filters the result of the query and lets you restrict the dataset result returned by the query. Once again, this can optimize performance. This function returns False if no content records found, else returns True.

This function returns False if no content records found, else returns True.

Example:

```
<div class="text">  
  <${QueryContentListSortedDesc('c', -1, True, 'Title', '')};$>  
  <FOREACH ds="c">  
    <b>Article Title:</b> {%!c.TITLE%}<br><br>  
  </FOREACH>  
</div>
```

QueryContentListSortedNext function

Usage: `function QueryContentListSortedNext(QueryName:string;
IncludeMainIndexContent:boolean; SortField:string):boolean;`

QueryName: string

A string value that will identify this query result

IncludeMainIndexContent :Boolean

If True, will include the main/index page for the channel

SortField :string

A String value which indicates which field to sort on

WhereFilter: string

To filter the result, example: '(CREATED_DATE >= "01/01/2003")' If blank, then all records will be returned.

Return Type: Boolean

Description:

A predefined custom query function which queries for the next list of sibling content after the current content (assumes current channel) with ability to specify SortField and whether to include main/index content record assigned to channel. This function returns False if no next content records found, else returns True.

WhereFilter filters the result of the query and lets you restrict the dataset result returned by the query. Once again, this can optimize performance. This function returns False if no content records found, else returns True.

This function is useful in automatically building previous/next content links.

Example:

```
<div class="text">  
  <${QueryContentListSortedNext('c', True, 'Title', 'Audience_Id=1');$}>  
  <FOREACH ds="c">  
    <b>Article Title:</b> {%!c.TITLE%}<br><br>  
  </FOREACH>  
</div>
```

QueryContentListSortedPrev function

Usage: `function QueryContentListSortedPrev(QueryName:string;
IncludeMainIndexContent:boolean; SortField:string):boolean;`

QueryName: string

A string value that will identify this query result

IncludeMainIndexContent :Boolean

If True, will include the main/index page for the channel

SortField :string

A String value which indicates which field to sort on

WhereFilter: string

To filter the result, example: '(CREATED_DATE >= "01/01/2003")' If blank, then all records will be returned.

Return Type: Boolean

Description:

A predefined custom query function which queries for the previous list of sibling content before the current content (assumes current channel) with ability to specify SortField and whether to include main/index content record assigned to channel. This function returns False if no previous content records found, else returns True. This function is useful in automatically building previous/next content links.

Example:

```
<div class="text">  
  <${QueryContentListSortedPrev('c', True, 'Title', '')};>  
  <FOREACH ds="c">  
    <b>Article Title:</b> {!c.TITLE%}<br><br>  
  </FOREACH>  
</div>
```

QueryContentMediaList function

Usage: `function QueryContentMediaList(QueryName: String; ContentId:
Integer): Boolean;`

QueryName: String

A string value that will identify this query result

ContentId: Integer

A Integer representing a ContentId value

Return Type: Boolean

Description:

A predefined custom query function which queries for the list of media/images associated with the content defined by ContentId. Passing a value of -1 for ContentId will assume the current content. This function returns False if no content media records found, else returns True.

Example:

```
<${QueryContentMediaList('c', 1);  
  var I: Integer=0;}>  
<FOREACH ds="c">  
  <${Inc(I, 1)};>  
  <${GetContentImageSrcById(QueryFieldAsInteger('c.Content_Id'), I, 0,  
    False, False, '')}>  
  <br>  
</FOREACH>
```

QueryFieldAsDateTime function

Usage: `function QueryFieldAsDateTime(Fieldname: String): DateTime;`

FieldName: String

The name of the field to return as DateTime

Return Type: DateTime

Description:

This function will take a field value from query and convert it to a DateTime

Example:

```
<$QueryContentList('c', GetChannelIdByName('News'));$>
<FOREACH ds="c">
  <$if QueryFieldAsDateTime('c.LAST_MODIFIED') <
    StrToDateTime('12/31/2004') then begin $>
    <a href="{%!c._link%}" >
      {%!c.LAST_MODIFIED%}
    </a>
    <br>
  <$end;$>
</FOREACH>
```

QueryFieldAsFloat function

Usage: `function QueryFieldAsFloat(Fieldname: String): Float;`

FieldName: String

The name of the field to return as Float

Return Type: Float

Description:

Returns a query field as a numeric floating point value

Example: *This example displays a list of articles, then outputs the "ContentId" field as a float value.*

```
<$QueryContentList('c', GetChannelIdByName('News'));$>
<FOREACH ds="c">
  {%!c.TITLE%} has contentID of: <%=QueryFieldAsFloat('c.CONTENT_ID')$>
  as a <b>float</b> value.<br>
</FOREACH>
```

QueryFieldAsInteger function

Usage: **function** QueryFieldAsInteger(FieldName: String): **Integer**;

FieldName: String

The name of the field to return as Integer

Return Type: Integer

Description:

Returns a query field as an integer value.

Example: *This example displays a list of articles, then outputs the "ContentId" field as an integer value.*

```
<${QueryContentList('c', GetChannelIdByName('News'))};$>
<FOREACH ds="c">
  {!c.TITLE!} has contentID of: <${QueryFieldAsInteger('c.CONTENT_ID')}$>
  as an <b>integer</b> value.<br>
</FOREACH>
```

QueryFieldAsString function

Usage: **function** QueryFieldAsString(FieldName: String): **String**;

FieldName: String

The name of the field to return as String

Return Type: String

Description:

Returns a query field as a string value.

Example:

```
<${QueryContentList('c', GetChannelIdByName('News'))};$>
<FOREACH ds="c">
  <a href="{!c.link!}">
    <${QueryFieldAsString('c.TITLE')}$>
  </a>
  <br>
</FOREACH>
```

QueryFieldByName function

Usage: **function** QueryFieldByName(FieldName: String): Variant;

FieldName: String The name of the field to return

Return Type: Variant

Description:

Returns a query field as a numeric or string value depending on where it is used. Only use this function if the field type for FieldName is not known else rather use one of the QueryFieldAs...() functions.

Example: This example displays a list of articles, then outputs the "ContentId" field as a variant value.

<pre><\${QueryContentList('c', GetChannelIdByName('News'))};\$> <FOREACH ds="c"> {%!c.TITLE%} has contentID of: <\${QueryFieldByName('c.CONTENT_ID')}\$> as a variant value.
 </FOREACH></pre>
--

QueryFieldIsNull function

Usage: **function** QueryFieldIsNull(FieldName: String): Boolean;

FieldName: String The name of the field to test

Return Type: Boolean

Description:

Returns True if a query field has a null value (i.e. value is unspecified! this is different from a blank or empty string value).

Example: This example returns all the article for the "News" channel, but only the articles that do not have an expiry date!

<pre><\${QueryContentList('c', GetChannelIdByName('News'))};\$> <FOREACH ds="c"> <\${if QueryFieldIsNull('c.EXPIRY_DATE') = True then begin \$> <\${QueryFieldAsString('c.TITLE')}\$>
 <\${end};\$> </FOREACH></pre>
--

QueryMediaList function

Usage: **function** QueryMediaList(QueryName: String; ChannelId: Integer): Boolean;

QueryName: String

A string value that will identify this query result

ChannelId: integer

An integer Id value for a channel/folder

Return Type: Boolean

Description:

A predefined custom query function which queries for the list of global media/images in the database for the specified ChannelId. Passing a value of -1 for ChannelId will assume the current channel, and value of -2 for ChannelId will return all media/images across all channels. This function returns False if no media records found, else returns True.

Example:

```
<$QueryMediaList('c', -2);$>
<FOREACH ds="c">
  <$if UpperCase(QueryFieldAsString('c.MEDIA_TYPE'))='IMAGE' then
    begin $>
      
<$ end;$>
  <br>
</FOREACH>
```

QueryIsStartChannel function

Usage: **function** QueryIsStartChannel(QueryName: String): Boolean;

QueryName: String

A string value that will identify this query result

Return Type: Boolean;

Description:

Returns True if the query specified by name has it's ChannelId field equal to the start(home) channel.

Example: *This example lists all of the sub channels for "News" channel and the parent channel of "News". It then tests if each channel is the site start channel.*

```
<$QueryChannelListAdv('ch', GetChannelIdByName('News'), True, True, '',
'', False, -1, -1);$>
<FOREACH ds="ch">
  <$if QueryIsStartChannel('ch') then
    begin $>
      {%!ch.NAME%} <-- this is the home channel <br>
    <$end
  else
    begin $>
      {%!ch.NAME%} <br>
    <$end;$>
  </FOREACH>
```

QueryIsParentChannel function

Usage: **function** QueryIsParentChannel(QueryName: String): **Boolean**;

QueryName: String

A string value that will identify this query result

Return Type: Boolean;

Description:

Returns True if the query specified by name has it's ChannelId field equal to the current channel's parent.

Example: *This example lists all of the sub channels for "News" channel and the parent channel of "News". It then tests if each channel is the parent channel.*

```
<$QueryChannelListAdv('ch', GetChannelIdByName('News'), True, True, '',
'', False, -1, -1);$>
<FOREACH ds="ch">
  <$if QueryIsParentChannel('ch') then
  begin $>
    {%!ch.NAME%} <-- this is the parent channel <br>
  <$end
  else
  begin $>
    {%!ch.NAME%} <br>
  <$end;$>
</FOREACH>
```

QueryIsCurrChannel function

Usage: **function** QueryIsCurrChannel(QueryName: String): **Boolean**;

QueryName: String

A string value that will identify this query result

Return Type: Boolean;

Description:

Returns True if the query specified by name has it's ChannelId field equal to the current channel.

Example: *This example lists all of the sub channels for "News" channel and the parent channel of "News". It then tests if each channel is the current channel.*

```
<$QueryChannelListAdv('ch', GetChannelIdByName('News'), True, True, '',
'', False, -1, -1);$>
<FOREACH ds="ch">
  <$if QueryIsCurrChannel('ch') then
  begin $>
    {%!ch.NAME%} <-- this is the current channel <br>
  <$end
  else
  begin $>
    {%!ch.NAME%} <br>
  <$end;$>
</FOREACH>
```

QueryIsCurrContent function

Usage: **function** QueryIsCurrContent(QueryName: String): Boolean;

QueryName: String

A string value that will identify this query result

Return Type: Boolean;

Description:

Returns True if the query specified by name has its ContentId field equal to the current content.

Example: This example lists all of the articles for "News" channel, it then tests if each article is the current article.

```
<$QueryContentList('c', GetChannelIdByName('News'));$>
<FOREACH ds="c">
  <$if QueryIsCurrContent('c') then
    begin $>
      {%!c.TITLE%} <-- this is the current article <br>
    <$end
  else
    begin $>
      {%!c.TITLE%} <br>
    <$end;$>
</FOREACH>
```

QueryIsCurrMedia function

Usage: **function** QueryIsCurrMedia(QueryName: String): Boolean;

QueryName: String

A string value that will identify this query result

Return Type: Boolean;

Description:

Returns True if the query specified by name has its MediaId field that's the same as the current media.

Example: This example displays the full list of items in the global images/media list.

```
<div class="img">
  <$QueryMediaList('m', -2);$>
  <FOREACH ds="m">
    <$if QueryIsCurrMedia('m') then
      begin $>
        This is the current image/media item:<br>
      <$end;$>
      <br>
    </FOREACH>
  </div>
```

QueryMedia function

Usage: **function** QueryMedia(QueryName: String; MediaId: Integer): **Boolean**;

QueryName: String

A string value that will identify this query result

MediaId: Integer;

An Integer value identifying a media item

Return Type: Boolean;

Description:

A predefined custom query function which queries the global image/media list for the image/media item specified by MediaId. Passing a value of -1 for MediaId will assume the current media. This function returns False if no record found, else returns True.

Example: This example queries the globalimage/media list for the first image/media item.

```
<div class="img">
  <$QueryMedia('m', 1);$>
  <FOREACH ds="m">
    <br>
  </FOREACH>
</div>
```

QueryParamAsInteger function

Usage: **procedure** QueryParamAsInteger(Fieldname: String; Value: Integer);

Fieldname: String;

The name of the field in the dataset to query

Value: Integer;

The value of the field to search for

Description:

Sets an integer parameter value specified by FieldName to be retrieved in a dataset. Prefix the query name in the FieldName if wanting to refer a field in a named query (i.e. not the default query). NOTE: The query should be closed when assigning parameters. Parameters are usually defined by ':' symbol followed by the parameter/field name in the query sql. This procedure is only useful when defining your own sql queries in the templates.

[See example on page 98](#)

QueryParamAsFloat function

Usage: `procedure QueryParamAsFloat(FieldName: String; Value: Float);`

<p>FieldName: String; The name of the field in the dataset to query</p> <p>Value: Float; The value of the field to search for</p>

Description:

Sets a floating point parameter value specified by FieldName to be retrieved in a dataset. Prefix the query name in the FieldName if wanting to refer a field in a named query (i.e. not the default query). NOTE: The query should be closed when assigning parameters. Parameters are usually defined by ':' symbol followed by the parameter/field name in the query sql. This procedure is only useful when defining your own sql queries in the templates.

[See example on page 103](#)

QueryParamAsString function

Usage: `procedure QueryParamAsString(FieldName: String; Value: String);`

<p>FieldName: String; The name of the field in the dataset to query</p> <p>Value: String; The value of the field to search for</p>
--

Description:

Sets a string parameter value specified by FieldName to be retrieved in a dataset. Prefix the query name in the FieldName if wanting to refer a field in a named query (i.e. not the default query). NOTE: The query should be closed when assigning parameters. Parameters are usually defined by ':' symbol followed by the parameter/field name in the query sql. This procedure is only useful when defining your own sql queries in the templates.

[See example on page 103](#)

QueryParamAsDateTime function

Usage: `procedure QueryParamAsDateTime(FieldName: String; Value: DateTime);`

<p>FieldName: String; The name of the field in the dataset to query</p> <p>Value: DateTime; The value of the field to search for</p>
--

Description:

Sets a DateTime parameter value specified by FieldName to be retrieved in a dataset. Prefix the query name in the FieldName if wanting to refer a field in a named query (i.e. not the default query). NOTE: The query should be closed when assigning parameters. Parameters are usually defined by ':' symbol followed by the parameter/field name in the query sql. This procedure is only useful when defining your own sql queries in the templates.

[See example on page 103](#)

QueryParamByName function

Usage: **procedure** `QueryParamAsByName`(FieldName: String; Value: Variant);

FieldName: String; The name of the field in the dataset to query
Value: Variant; The value of the field to search for

Description:

Sets a numeric or string parameter value specified by FieldName depending on the type of value passed. It's usually preferable to use the `QueryParamAs...()` procedures to avoid any possible type mismatch errors.

[See example on page 103](#)

QueryParamClear function

Usage: **procedure** `QueryParamClear`(FieldName: String);

FieldName: String; The name of the field in the dataset to query
--

Description:

Sets a numeric or string parameter value specified by FieldName depending on the type of value passed. It's usually preferable to use the `QueryParamAs...()` procedures to avoid any possible type mismatch errors.

[See example on page 103](#)

QueryOpen function

Usage: **procedure** `QueryOpen`(QueryName: String);

QueryName: String; The name of the field in the dataset to query
--

Description:

Opens the query defined by QueryName. If the query is already open it will first be closed then reopened.

[See example on page 103](#)

QueryNext function

Usage: **procedure** `QueryNext`(QueryName: String);

QueryName: String; The name of the field in the dataset to query
--

Description:

Moves the query pointer defined by QueryName to the next record.

[See example on page 103](#)

QueryFirst function

Usage: **procedure** QueryFirst(QueryName: String);

QueryName: String; The name of the field in the dataset to query
--

Description:

Moves the query pointer defined by QueryName to the first record.

[See example on page 103](#)

QueryBOF function

Usage: **function** QueryBOF(QueryName: String): Boolean;

QueryName: String; The name of the field in the dataset to query
--

Description:

Returns True if the query defined by QueryName is at the Beginning Of File (BOF) (i.e. the first record) else this function returns False.

[See example on page 103](#)

QueryEOF function

Usage: **function** QueryEOF(QueryName: String): Boolean;

QueryName: String; The name of the field in the dataset to query
--

Description:

Returns True if the query defined by QueryName is at the End Of File (EOF) (i.e. the last record) else this function returns False.

[See example on page 103](#)

QueryClose function

Usage: **procedure** QueryClose(QueryName: String);

QueryName: String;
The name of the field in the dataset to query

Description:

Closes the query defined by QueryName.

Example: This example builds a custom query to select all content created before the current date/time. It uses a query parameter (CREATED_DATE) and the parameter gets passed to the query to retrieve the relevant results.

```
<SQL name="*s">
  select content_id, title
  from Content
  where (CREATED_DATE <= :CREATED_DATE)
  order by TITLE
</SQL>

<$QueryParamAsDateTime('s.CREATED_DATE', Now);
  QueryOpen('s');
  while not QueryEOF('s') do
  begin $>
    <a href="{!s._link%}">{!s.TITLE%}</a><br>
  <$ QueryNext('s');
  end;
  QueryClose('s');$>
```

QueryProjectMediaList function

Usage: **function** QueryProjectMediaList(QueryName: String; ProjectId: Integer): Boolean;

QueryName: String
A string value that will identify this query result

ProjectId: Integer;
An Integer value identifying the project

Return Type: Boolean;

Description:

A predefined custom query function which queries for the list of media/images associated with the project defined by ProjectId. Currently only a single project is allowed per database so ProjectId should be 1. This function returns False if no project media records found, else returns True.

Example: This example displays the list of images assigned to the project.

```
<div class="img">
  <$QueryProjectMediaList('m', 1);$>
  <FOREACH ds="m">
    <br>
  </FOREACH>
</div>
```

QueryRecordCount function

Usage: **function** QueryRecordCount(QueryName: String): **Integer**;

QueryName: String

A string value that will identify this query result

Return Type: Integer;

Description:

Returns an integer value indicating the total number of records returned by the query defined by QueryName.

Example:

```
<$QueryContentList('c', GetChannelIdByName('News'));$>
```

There is currently: <\$=QueryRecordCount('c')\$> article(s) in the
News channel.

Note: Not all database systems return an accurate RecordCount, especially certain relational database types. To ensure that the RecordCount is always correct, first call function QueryLast() and then QueryFirst() before calling the QueryRecordCount() function. However this method is more database intensive so should only be performed if needed.

e.g.

```
<$QueryLast('c');
```

```
QueryFirst('c');$>
```

The record count is: <\$=QueryRecordCount('c')\$>

QueryRelatedContentList function

Usage: `function QueryRelatedContentList(QueryName: String; ContentId: Integer; Descending: Boolean): Boolean;`

QueryName: String

A string value that will identify this query result

ContentId: Integer;

An Integer value identifying the content

Descending: Boolean

If True, then the query will be sorted descending

Return Type: Boolean;

Description:

A predefined custom query function which queries the related content list for a specific content. Passing a value of -1 for ContentId will assume the current content.

Example:

```
<$QueryRelatedContentList('c', 1, False);
  if QueryRecordCount('c') > 0 then
  begin $>
    <b>Related Content:</b>
    <div class="related_content">
      <FOREACH ds="c">
        {%!c.TITLE%}
        <br>
      </FOREACH>
    </div>
  <$end;$>
```

5. Date/Time Related Functions/Procedures:

These functions deal with the formatting of date/time variables. They can be useful when wanting to display the date/time an article was published or last modified, etc.

Date function

Usage: **function** Date: DateTime;

Return Type: DateTime

Description:

Returns the current system date

Example:

```
<div class="date">
  The date now is: <%=Date%>
</div>
```

DateTimeToStr function

Usage: **function** DateTimeToStr(dt: DateTime): String;

```
dt: DateTime
A date/time variable value
```

Return Type: String

Description:

DateTimeToStr converts the DateTime value given by dt using the format given by the ShortDateFormat global variable, followed by the time using the format given by the LongTimeFormat global variable. The time is not displayed if the fractional part of the DateTime value is zero.

DateToStr function

Usage: **function** DateToStr(dt: DateTime): String;

```
dt: DateTime
A date/time variable value
```

Return Type: String

Description:

This function works like **DateTimeToStr**, except only the date portion is returned, not the time portion

DayOfWeek function

Usage: **function** DayOfWeek(dt: DateTime): **Integer**;

dt: DateTime
A date/time variable value

Return Type: Integer

Description:

DayOfWeek returns the day of the week of the specified date as an integer between 1 and 7, where Sunday is the first day of the week and Saturday is the seventh.

Example:

Today is the `<%=DayOfWeek(Now)%>` day of the week.

DecodeDate procedure

Usage: **procedure** DecodeDate(dt: DateTime; var y: Integer; var m: Integer; var d: Integer);

dt: DateTime
A date/time variable value

var y: Integer;
The returned value specifying the year component of the date

var m: Integer;
The returned value specifying the month component of the date

var d: Integer
The returned value specifying the day component of the date

Return Type: N/A

Description:

The DecodeDate procedure breaks the value specified as the DateTime parameter into Year, Month, and Day values. If the given DateTime value is less than or equal to zero, then year, month, and day return parameters are all set to zero.

Example: *This example displays the current date and time.*

```
<%=var Day, Month, Year: Integer;
var Hour, Min, Sec, MSec: Integer;

DecodeDate(Now, Year, Month, Day);
DecodeTime(Now, Hour, Min, Sec, MSec);%>

<div class="DateTime">
  Today is the <%=Day%> of <%=Month%>, and the year is <%=Year%><br>
  The time is now: <%=Hour%>:<%=Min%>:<%=Sec%>
</div>
```

DecodeTime procedure

Usage: `procedure DecodeTime(dt: DateTime; var h: Integer; var m: Integer; var s: Integer; var ms: Integer);`

dt: DateTime

A date/time variable value

var h: Integer;

The returned value specifying the hour component of the time

var m: Integer;

The returned value specifying the minute component of the time

var s: Integer

The returned value specifying the seconds component of the time

var ms: Integer

The returned value specifying the milli-seconds component of the time

Return Type: N/A

Description:

The DecodeTime procedure breaks the value specified as the DateTime parameter into Hour, Minute, Seconds and Milliseconds values. If the given DateTime value is less than or equal to zero, then hour, minute, seconds and milliseconds return parameters are all set to zero.

Example: *This example displays the current date and time.*

```
<$var Day, Month, Year: Integer;
  var Hour, Min, Sec, MSec: Integer;

  DecodeDate(Now, Year, Month, Day);
  DecodeTime(Now, Hour, Min, Sec, MSec);$>

<div class="DateTime">
  Today is the <%=Day%> of <%=Month%>, and the year is <%=Year%><br>
  The time is now: <%=Hour%>:<%=Min%>:<%=Sec%>
</div>
```

EncodeDate function

Usage: `function EncodeDate(y: Integer; m: Integer; d: Integer): DateTime;`

```
var y: Integer;  
The value specifying the year component of the date  
  
var m: Integer;  
The value specifying the month component of the date  
  
var d: Integer;  
The value specifying the day component of the date
```

Return Type: DateTime

Description:

Returns a DateTime value from the values specified as the Year, Month, and Day parameters. The year must be between 1 and 9999. Valid Month values are 1 through 12. Valid Day values are 1 through 28, 29, 30, or 31, depending on the Month value.

For example, the possible Day values for month 2 (February) are 1 through 28 or 1 through 29, depending on whether or not the Year value specifies a leap year.

Example: *This example retrieves the articles for the "News" channel but only displays the articles if they were published after 2004/12/31.*

```
<$QueryContentList('c', GetChannelIdByName('News'));$>  
<FOREACH ds="c">  
  <$if GetPublishDateTime > EncodeDate(2004, 12, 31) then  
    begin $>  
      <a href="{!c._link%}">{!c.TITLE%}</a><br>  
    <$end;$>  
</FOREACH>
```

EncodeTime function

Usage: `function EncodeTime(h: Integer; m: Integer; s: Integer; ms: Integer): DateTime;`

```
var h: Integer;  
The value specifying the hour component of the time  
  
var m: Integer;  
The value specifying the minute component of the time  
  
var s: Integer;  
The value specifying the seconds component of the time  
  
var ms: Integer;  
The value specifying the milli-seconds component of the time
```

Return Type: DateTime

Description:

EncodeTime encodes the given hour, minute, second, and millisecond into a DateTime value. Valid Hour values are 0 through 23. Valid Min and Sec values are 0 through 59. Valid MSec values are 0 through 999.

Example: *This example retrieves the articles for the "News" channel but only displays the articles if they were published after 11:30 PM.*

```
<${QueryContentList('c', GetChannelIdByName('News'))};$>  
<FOREACH ds="c">  
  <${if GetPublishDateTime > EncodeTime(23, 30, 0, 0) then  
    begin $>  
      <a href="{!c._link%}">{!c.TITLE%}</a><br>  
    <${end};$>  
  </FOREACH>
```

FormatDateTime function

Usage: **function** FormatDateTime(frm: String; dt: DateTime): **String**;

frm: String The desired output format for the date/time value
dt: DateTime A date/time variable value

Return Type: String

Description:

FormatDateTime formats the DateTime value given by **dt** using the format given by **frm**. See Date-Time format strings below for more information. If the string specified by the **frm** parameter is empty, the DateTime value is formatted as if a 'c' format specifier had been given.

DateTime format strings:

Specifier	Displays
c	Displays the date using the format given by the ShortDateFormat global variable, followed by the time using the format given by the LongTimeFormat global variable. The time is not displayed if the fractional part of the DateTime value is zero.
d	Displays the day as a number without a leading zero (1-31)
dd	Displays the day as a number with a leading zero (01-31)
ddd	Displays the day as an abbreviation (Sun-Sat)
dddd	Displays the day as a full name (Sunday-Saturday)
dddddd	Displays the date using the format given by the System ShortDateFormat
ddddddd	Displays the date using the format given by the System LongDateFormat
m	Displays the month as a number without a leading zero (1-12). If the m specifier immediately follows an h or hh specifier, the minute rather than the month is displayed
mm	Displays the month as a number with a leading zero (01-12). If the mm specifier immediately follows an h or hh specifier, the minute rather than the month is displayed
mmm	Displays the month as an abbreviation (Jan-Dec)
mmmmm	Displays the month as a full name (January-December)
yy	Displays the year as a two-digit number (00-99)
yyyy	Displays the year as a four-digit number (0000-9999)
h	Displays the hour without a leading zero
hh	Displays the hour with a leading zero
n	Displays the minute without a leading zero
nn	Displays the minute with a leading zero
s	Displays the second without a leading zero
ss	Displays the second with a leading zero
z	Displays the millisecond without a leading zero
zzz	Displays the millisecond with a leading zero
t	Displays the time using the format given by the system ShortTimeFormat
tt	Displays the time using the format given by the system LongTimeFormat
am/pm	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'am' for any hour before noon, and 'pm' for any hour after noon. The am/pm specifier can use lower, upper, or mixed case, and the result is displayed accordingly
a/p	Uses the 12-hour clock for the preceding h or hh specifier, and displays 'a' for any hour before noon, and 'p' for any hour after noon. The a/p specifier can use lower, upper, or mixed case, and the result is displayed accordingly
ampm	Uses the 12-hour clock for the preceding h or hh specifier, and displays the contents of the system TimeAMString for any

	hour before noon, and the contents of the TimePMString for any hour after noon
/	Displays the date separator character given by the system DateSeparator
:	Displays the time separator character given by the system TimeSeparator
'xx'/"xx"	Characters enclosed in single or double quotes are displayed as-is, and do not affect formatting

Example:

```
<div class="date">
  The date now is: <b><%=FormatDateTime('dddd dd mmmm yyyy
                        hh:mm', Now)></b>.
</div>
```

IsLeapYear function

Usage: **function** IsLeapYear(year: Integer): Boolean;

```
year: Integer
An integer value > 0
```

Return Type: Boolean;

Description:

This function returns TRUE if the year specified is a leap year, FALSE otherwise.

Example:

```
<$var Year, Month, Day: Integer;
  DecodeDate(Now, Year, Month, Day);
  if IsLeapYear(Year) then
  begin
    SendLn(IntToStr(Year) + ' is a leap year!');
  end
  else
  begin
    SendLn(IntToStr(Year) + ' is not a leap year!');
  end; $>
```

Now function

Usage: **function** Now: DateTime;

Return Type: DateTime

Description:

This function returns the current system date/time.

Example: *This example displays the current(system) date and time.*

```
<div class="date">
  The current date/time is: <%=DateTimeToStr(Now)>
</div>
```

StrToDateTime function

Usage: **function** StrToDateTime(str: String): DateTime;

str: String
A string which will be converted to a DateTime

Return Type: Boolean

Description:

Converts a string to a TDateTime value. If str does not contain a valid date, StrToDateTime raises an error.

The str parameter must use the current locale's date/time format. In the US, this is commonly MM/DD/YY HH:MM:SS format. Specifying AM or PM as part of the time is optional, as are the seconds. Use 24-hour time (7:45 PM is entered as 19:45, for example) if AM or PM is not specified.

Example: This example displays the current(system) date/time.

```
<div class="date">  
  The current date/time is: <%=StrToDateTime('12/31/2004 23:59')%>  
</div>
```

Time function

Usage: **function** Time: DateTime;

Return Type: DateTime

Description:

Returns the current system date

Example: This example returns the current(system) time.

```
<div class="time">  
  The time now is: <%=TimeToStr(Time)%>  
</div>
```

TimeToStr function

Usage: **function** TimeToStr(dt: DateTime): String;

dt: DateTime
A date/time variable value

Return Type: String

Description:

This function works like **DateTimeToStr**, except only the time portion is returned, not the date portion.

Example: This example returns the current(system) time.

```
<div class="time">  
  The time now is: <%=TimeToStr(Time)%>  
</div>
```

6. String Related Functions and Procedures:

This group of functions are string formatting functions and procedures.

AnsiLowerCase function

Usage: **function** AnsiLowerCase(str: String): String;

```
str: String
A String value to be converted to lowercase
```

Return Type: String

Description:

AnsiLowerCase returns a string that is a copy of the given string converted to lower case. The conversion uses the current locale. This function supports multi-byte character sets (MBCS).

Example: *This example will result in an output of: "the rain in spain falls mainly in the plain."*

```
<div>
  <%=AnsiLowerCase('The rain in Spain falls mainly in the Plain.')==>
</div>
```

AnsiQuotedStr function

Usage: **function** AnsiQuotedStr(S: String; Quote: String): String;

```
S: String
A string value

Quote: String
The quote value to use
```

Return Type: String

Description:

Returns the quoted form of string 'S' using the Quote character specified.

Example: *This example will result in an output of: " Color="#ABCDEF"*

```
<div>
  Color=<%=AnsiQuotedStr('#ABCDEF', '"')==>
</div>
```

AnsiUpperCase function

Usage: **function** AnsiUpperCase(str: String): String;

```
str: String
A String value to be converted to uppercase
```

Return Type: String

Description:

AnsiUpperCase returns a string that is a copy of the given string converted to upper case. The conversion uses the current locale. This function supports multi-byte character sets (MBCS).

Example: *This example will result in an output of: "THE RAIN IN SPAIN FALLS MAINLY IN THE PLAIN."*

```
<div>
  <%=AnsiUpperCase('The rain in Spain falls mainly in the Plain.')==>
</div>
```

Chr function

Usage: **function** Chr(x: Integer): String;

```
x: Integer
An Integer value
```

Return Type: String

Description:

Returns the character for a specified ASCII value

Example: *This example displays the alphabet by displaying each character from its keycode value.*

```
<div class="alphabet">
  <%=var I: Integer;%=>
  <%=for I := 65 to 122 do
    SendLn(Chr(I));%=>
</div>
```

ColorToHTMLColorString function

Usage: **function** ColorToHTMLColorString(Color: Integer): String;

```
Color: Integer
An Integer value representing a color value
```

Return Type: String

Description:

Converts a numeric integer color code, to the HTML equivalent

Example: *This example converts the numeric value to its html color equivalent.*

```
<%=ColorToHTMLColorString(254)==>
```

CompactNewLine function

Usage: **function** CompactNewLine(Text: String): **String**;

Text: String
The string value to compact

Return Type: String

Description:

Returns the compacted form of Text with all #13#10 NewLine chars replaced with #1 chars. This function is useful to compact multi-line values (which use #13#10 for separate multiple lines) so that they can be stored as property values (which require multi-line text to be separated with #1 chars.)

This function is equivalent to `<%=ReplaceString(Text, #13#10, #1)%>`

Example: *This example expands the new line characters in the text:*

```
<%=CompactNewLine('Some multiline text'#13#10'line 2'#13#10'line 3')%>
```

Copy function

Usage: **function** Copy(str: String; Index: Integer; Len: Integer): **String**;

str: String
A String value from which to copy

Index: Integer
An Integer representing the index from where to start copying

Len: Integer
An Integer that represents how many characters to copy

Return Type: String

Description:

Returns a substring of a string. Str is an expression of a string, Index and Count are integer-type expressions. Copy returns a substring containing Count characters or elements starting at str[Index]. If Index is larger than the length of str, Copy returns an empty string. If Count specifies more characters than are available, only the characters or elements from str[Index] to the end of S are returned.

Example: *This example will result in an output of: "Spain".*

```
<%=Copy('The rain in Spain falls mainly in the plain', 13, 5)%>
```

Delete procedure

Usage: `procedure Delete(var S: String; Index: Integer; Len: Integer);`

```
var S: String  
The string value from which to delete  
  
Index: Integer  
An Integer representing the index from where to start deleting  
  
Len: Integer  
An Integer that represents how many characters to delete
```

Return Type: n/a

Description:

Removes a substring of Count characters from string S starting with S[Index]. S is a string-type variable. Index and Count are integer-type expressions.

If index is larger than the length of the string or less than 1, no characters are deleted.

If count specifies more characters than remain starting at the index, Delete removes the rest of the string. If count is less than or equal to 0, no characters are deleted.

Example: *This example will result in an output of: "The rain in Spain".*

```
<$  
var Str: String = 'The rain in Spain falls mainly in the plain';  
Delete(Str, 18, 26);  
SendLn(Str);  
$>
```

DoubleQuotedStr function

Usage: `function DoubleQuotedStr(S: String): String;`

```
S: String  
A string value
```

Return Type: String

Description:

Returns the quoted form of string 'S' using a doubled quote character.

Example: *This example will result in an output of: "The rain in spain" (incl. the "" characters).*

```
<$=DoubleQuotedStr('The rain in spain')$>
```

ExpandNewLine function

Usage: **function** ExpandNewLine(Text: String): String;

Text: String
The string value to compact

Return Type: String

Description:

Returns the expanded form of Text with all #1 chars replaced with #13#10 (NewLine chars.) This function is useful to expand compacted multi-line property values (which are separated internally with #1 chars) to the proper multi-line equivalent which require multiple lines to be separated with #13#10.

This function is equivalent to `<%=ReplaceString(Text, #1, #13#10)%>`.

Example: *This example expands the new line characters in the text:*

```
<%=ExpandNewLine(GetContentPropValue('ShortBlurb_memo'))%>
```

FloatToStr function

Usage: **function** FloatToStr(f: Float): String;

f: Float
A floating point value to be converted to a string

Return type: String

Description:

converts the floating-point value given by Value to its string representation.

Example: *This example displays the decimal portion of the float value "1.23".*

```
<%=Copy(FloatToStr(1.23), 3, 2)%>
```

HTMLToText function

Usage: **function** HTMLToText(HTML: String): String;

HTML: String
A string value representing html code

Return Type: String

Description:

Converts HTML formatted text to it's plain text equivalent, so that special tags such as 'j', etc. gets converted to '&', etc.

Example: *This example results in the following output: "5 & 4 = 9"*

```
<%=HTMLToText('<b>5 &#106; 4 &#106; = 9</b>')%>
```

FormatFloat function

Usage: **function** FormatFloat(Format: string; Value: Float): **String**

Format: String

A floating point value to be converted to a string. The following format specifiers are supported in the format string:

- 0 Digit place holder. If the value being formatted has a digit in the position where the '0' appears in the format string, then that digit is copied to the output string. Otherwise, a '0' is stored in that position in the output string.
- # Digit placeholder. If the value being formatted has a digit in the position where the '#' appears in the format string, then that digit is copied to the output string. Otherwise, nothing is stored in that position in the output string.
- . Decimal point. The first '.' character in the format string determines the location of the decimal separator in the formatted value; any additional '.' characters are ignored.
- , Thousand separator. If the format string contains one or more ',' characters, the output will have thousand separators inserted between each group of three digits to the left of the decimal point. The placement and number of ',' characters in the format string does not affect the output, except to indicate that thousand separators are wanted.
- E+ Scientific notation. If any of the strings 'E+', 'E-', 'e+', or 'e-' are contained in the format string, the number is formatted using scientific notation. A group of up to four '0' characters can immediately follow the 'E+', 'E-', 'e+', or 'e-' to determine the minimum number of digits in the exponent. The 'E+' and 'e+' formats cause a plus sign to be output for positive exponents and a minus sign to be output for negative exponents. The 'E-' and 'e-' formats output a sign character only for negative exponents.
- 'xx'/"xx" Characters enclosed in single or double quotes are output as-is, and do not affect formatting.
- ; Separates sections for positive, negative, and zero numbers in the format string.

Value: Float

A floating point value to be formatted

Return Type: String

Description:

FormatFloat formats the floating-point value given by Value using the format string given by Format.

Example: The following examples displays formatted float values.

```
<$=FormatFloat('$#,##0.00', 1234567)$> = $1,234,567.00
<$=FormatFloat('#, ##0.00;-#, ##;Zero', 1234)$> = 1,234.00
<$=FormatFloat('#, ##0.00;-# ,##;Zero', -1234)$> = -1,234
<$=FormatFloat('#, ##0.00;-#, ##;Zero', 0)$> = Zero
```

Insert procedure

Usage: `procedure Insert(src: String; var S: String; Index: Integer);`

```
src: String  
The substring to insert  
  
var S: String  
The string into which to insert  
  
Index: Integer  
The position where to insert
```

Return type: n/a

Description:

Insert merges src into S at the position S[Index]. Src is a string-type expression. S is a string-type variable of any length. Index is an integer-type expression. It is a character index and not a byte index. If Index is less than 1, it is mapped to a 1. If it is past the end of the string, it is set to the length of the string, turning the operation into an append.

If the Source parameter is an empty string, Insert does nothing.

Example: *This example results in the following output: "The rain in Spain falls mainly in the plain".*

```
<$  
var Str: String = 'The rain in the plain';  
Insert(' Spain falls mainly in ', Str, 12);  
SendLn(Str);  
$>
```

IntToStr function

Usage: `function IntToStr(I: Integer): String;`

```
I: Integer  
An Integer value to convert to a String
```

Return Type: String

Description:

IntToStr converts an integer into a string containing the decimal representation of that number.

Example: *This example results in the following output: "34"*

```
<$=Copy(IntToStr(123456), 3, 2)$>
```

LeftStr function

Usage: **function** LeftStr(AText: String; ACount: Integer): **String**;

AText: String A string value
ACount: Integer An Integer value representing the number of characters to copy

Return Type: String

Description:

Returns the leftmost characters of AText up to a length of ACount characters

Example: This example results in the following output: "The rain in Spain"

```
<%=LeftStr('The rain in Spain falls mainly in the plain', 17)%>
```

Length function

Usage: **function** Length(str: String): **Integer**;

str: String A String value which will have it's length determined

Return Type: Integer;

Description:

Returns the number of characters in a string.

Example: This example tests if the content property "SideBarText" is blank, by testing its length. If it is not blank (i.e. length > 0), then it displays the content property value.

```
<$if length(GetContentPropValue('SideBarText_html')) > 0 then  
begin  
  GetContentPropValue('SideBarHeader');$><br>  
  <%=GetContentPropValue('SideBarText_html')%>  
<$end;$>
```

LoadStringFromFile function

Usage: **function** LoadStringFromFile(fileName: String): **String**;

filename: String The name of the file from which to extract.
--

Return Type: String

Description:

Loads the contents of a file on disk into a string.

Example: This example loads and displays the text from the file "c:\test.txt"

```
<%=LoadStringFromFile('c:\test.txt')%>
```

LowerCase function

Usage: **function** LowerCase(str: String): String;

str: String
A String value to be converted to lowercase

Return Type: String

Description:

LowerCase returns a string with the same text as the string passed in S, but with all letters converted to lowercase. The conversion affects only 7-bit ASCII characters between 'A' and 'Z'. To convert 8-bit international characters, use AnsiLowerCase.

Example: *This example results in the following output: "the rain in spain falls mainly in the plain".*

```
<%=LowerCase('The Rain in SpaIn falls maINly in the plain')$>
```

MidStr function

Usage: **function** MidStr(AText: String; AStart: Integer; ACount: Integer): String;

AText: String
A string value

AStart: Integer
An Integer value specifying the start point

ACount: Integer
An Integer value representing the number of charcters to copy

Return Type: String

Description:

Returns a substring with ACount characters starting at AStart in the string AText

Example: *This example results in the following output: "rain in Spain".*

```
<%=MidStr('The rain in Spain falls mainly in the plain', 5, 13)$>
```

Ord function

Usage: **function** Ord(s: String): Integer;

s: String
A string value

Return Type: Integer

Description:

Returns the ordinal value of an ordinal-type expression.

Example: *This example displays the ordinal value for the charcter "A".*

```
The ascii code for "A" = <%=Ord('A')$>
```

Pos function

Usage: `function Pos(subStr: String; str: String): Integer;`

<p>subStr: String The substring text to locate</p> <p>str: String The string in which to search</p>

Return Type: Integer;

Description:

Pos searches for a substring, subSstr, in a string, str. Substr and str are string-type expressions. Pos searches for subSstr within str and returns an integer value that is the index of the first character of subSstr within str. Pos is case-sensitive. If Substr is not found, Pos returns zero.

Example: *This example results in the following output: "13". i.e. the position of the word "Spain" in the sentence below.*

```
<%=Pos('Spain', 'The rain in Spain falls mainly in the plain')%>
```

PosEx function

Usage: `function PosEx(SubStr: String; S: String; Offset: Integer): Integer;`

<p>SubStr: String A string value representing the substring value to search for</p> <p>S: String A string value, the original string</p> <p>Offset: Integer An Integer value specifying where to start the search</p>
--

Return Type: Integer

Description:

Returns the index of SubStr in S, beginning the search at Offset. If Offset is 1 (default), PosEx is equivalent to Pos

Example: *This example results in the following output: "31". i.e. the position of the word " in " in the sentence below from the 12th position.*

```
<%=PosEx(' in ', 'The rain in Spain falls mainly in the plain', 12)%>
```

RemoveHTMLTags function

Usage: **function** RemoveHTMLTags(HTML: String): **String**;

HTML: String

A string value representing html code

Return Type: String

Description:

Returns only the plain text (excluding all the HTML tags) from the specified 'HTML' string

Example: *This example results in the following output: "The rain in Spain falls mainly in the plain", i.e. all of the html tags formatting removed.*

```
<%=RemoveHTMLTags('<b>The rain in <i>Spain</i> falls mainly in the  
<i>plain</i></b>')$>
```

RepeatStr function

Usage: **function** RepeatStr(Text: String; Count: Integer): **String**;

Text: String

A string value to be repeated.

Count: Integer

An integer value specifying how many times to repeat the text.

Return Type: String

Description:

Returns a string value where the "Text" parameter has been repeated "Count" times.

Example: *This example repeats the "-*" 5 times, and results in the following output: "-*-*-*-*-"*

```
<%=RepeatStr('-*', 5)$>
```

ReplaceString function

Usage: `function ReplaceString(SourceStr: String; OldStr: String; NewStr: String): String;`

```
SourceStr: String
The source string

OldStr: String
The substring to replace

NewStr: String
The value to replace OldStr with
```

Return Type: String

Description:

Replaces occurrences of the substring specified by OldStr with the substring specified by NewStr in SourceStr.

Example: *This example results in the following output: "The rain in England falls mainly in the plain".*

```
<%=ReplaceString('The rain in Spain falls mainly in the plain', 'Spain',
'England')%>
```

RightStr function

Usage: `function RightStr(AText: String; ACount: Integer): String;`

```
AText: String
A string value

ACount: Integer
An Integer value representing the number of characters to copy
```

Return Type: String

Description:

Returns the rightmost characters of AText up to a length of ACount characters

Example: *This results in the following output: "mainly in the plain".*

```
<%=RightStr('The rain in Spain falls mainly in the plain', 20)%>
```

SendLn procedure

Usage: `procedure SendLn(s: Variant);`

```
s: Variant
The value to render in the page
```

Return Type: n/a

Description:

This procedure takes a variant value and renders to output in the page as text

Example: *This example displays the text below on the screen.*

```
<%=SendLn('This text will be displayed.')%>
```

StrToInt function

Usage: `function StrToInt(str: String): Integer;`

str: String

A String value to be converted to an Integer

Return Type: Integer

Description:

Converts a string that represents an integer (decimal or hex notation) to a number.

Example: *This function results in output of: "3".*

```
<%=StrToInt('1') + StrToInt('2')%>
```

StrToIntDef function

Usage: **function** StrToIntDef(str: String; def: Integer): **Integer**;

<p>str: String A String value to be converted to an Integer</p> <p>def: Integer A default Integer value</p>

Return Type: Integer

Description:

Converts a given string to an Integer value with error default. If the conversion fails for any reason, then the default value is returned.

Example: *This example adds the content property value "Number_Entries" to 100. If the content property value does not convert to an integer, then the value "0" is used.*

```
<%=StrToIntDef(GetContentPropValue('Number_Entries'), 0) + 100$>
```

StrToFloat function

Usage: **function** StrToFloat(str: String): **Float**;

<p>str: String A string value to be converted to a floating point value</p>
--

Return type: Float

Description:

Converts a given string to a floating-point value.

Example: *This function results in output of: "4".*

```
<%=StrToFloat('5.5') - StrToFloat('1.5')$>
```

StrToFloatDef function

Usage: **function** StrToFloatDef(str: String; def: Float): **Float**;

<p>str: String A string value to be converted to a floating point value</p> <p>def: Float A default float value.</p>
--

Return Type: Float

Description:

Converts a given string to a floating-point value with error default. If the conversion fails for any reason, then the default value is returned.

Example: *This example divides property value "Degres_Celcius" by 100. If the content property value does not convert to a float, then the value "0" is used.*

```
<%=StrToFloatDef(GetContentPropValue('Degrees_Celcius'), 0)/100%>
```

TextToHTML function

Usage: **function** TextToHTML(Text: String): **String**;

<p>Text: String The string value to convert to HTML</p>
--

Return Type: String

Description:

Converts plain text to it's HTML formatted equivalent, so that special symbols such as '&', etc. gets converted to '&', etc.

Example: *This example results in output of: "5 & "*

```
<%=TextToHTML('&lt;b&gt;5 & &lt;/b&gt;')%>
```

Trim function

Usage: **function** Trim(str: String): **String**;

<p>str: String The String value to be trimmed</p>
--

Return Type: String

Description:

Trims leading and trailing spaces and control characters from a string.

Example: *This example results in output of: "The rain in Spain falls mainly in the plain".*

```
<%=Trim('    The rain in Spain falls mainly in the plain    ')%>
```

TrimLeft function

Usage: `function TrimLeft(str: String): String;`

```
str: String
The String value to be trimmed
```

Return Type: String

Description:

Trims leading spaces and control characters from a string.

Example: *This example results in output of: "The rain in Spain falls mainly in the plain".*

```
<%=Trim('    The rain in Spain falls mainly in the plain')$>
```

TrimRight function

Usage: `function TrimRight(str: String): String;`

```
str: String
The String value to be trimmed
```

Return Type: String

Description:

Trims trailing spaces and control characters from a string.

Example: *This example results in output of: "The rain in Spain falls mainly in the plain".*

```
<%=Trim('The rain in Spain falls mainly in the plain    ')$>
```

UpperCase function

Usage: `function UpperCase(str: String): String;`

```
str: String
A String value to be converted to uppercase
```

Return Type: String

Description:

UpperCase returns a string with the same text as the string passed in S, but with all letters converted to uppercase. The conversion affects only 7-bit ASCII characters between 'A' and 'Z'. To convert 8-bit international characters, use AnsiUpperCase.

Example: *This example will result in output of: "THE RAIN IN SPAIN FALLS MAINLY IN THE PLAIN".*

```
<%=UpperCase('The rain in Spain falls mainly in the plain')$>
```

7. System Related Functions/Procedures:

These functions and procedures retrieve system specific information about Encore or the current project.

GetAbsLinkPrefix function

Usage: **function** GetAbsLinkPrefix: String;

Return Type: String

Description:

Returns the current value of the AbsLinkPrefix (used with GetAbsChannelLink(), GetAbsContentLink() & GetAbsMediaLink() functions)

Example: This example assigns the global absolute link prefix, then uses it to prefix a link url.

```
<${SetAbsLinkPrefix('http://www.mywebsite.com/')};>  
<a href="!_abslink%">!TITLE% </a><br>  
The current absolute prefix is: <${GetAbsLinkPrefix()}>
```

SetAbsLinkPrefix procedure

Usage: **procedure** GetAbsLinkPrefix(AbsLinkPrefix: String): String;

AbsLinkPrefix: String

A String value to assign to the the system global var: 'INIT_ABS_LINK_PREFIX'

Return Type: N/A

Description:

Set the global var. 'INIT_ABS_LINK_PREFIX' to the initial abs. link prefix. If not specified then assumes ''.
Example values for AbsLinkPrefix: '', '/', 'http://www.mywebsite.com/', etc.

Example: This example assigns the global absolute link prefix, then uses it to prefix a link url.

```
<${SetAbsLinkPrefix('http://www.mywebsite.com/')};>  
<a href="!_abslink%">!TITLE% </a>
```

GetAppInfo function

Usage: **function** GetAppInfo(InfoType: String): **String**;

InfoType: String

A String value indicating what kind of info is being requested

Return Type: String

Description:

Returns application specific information.

'InfoType' can have one of the following values:

' ' - A blank value returns the default value of App. Name + Version + Edition.

'name' - App. Name, i.e. 'CMS Encore Pro'.

'edition' - App. Edition, example: 'Enterprise'.

'version' - App. Version, example: '1.23'.

'major_version' - Major Version number of App. Version, example: '1'

'minor_version' - Minor Version number of App. Version, example: '23'.

'builddate' - App. Build Date in 'yyyy/mm/dd' format.

'company' - Company Name, i.e. 'BIIT Software Systems'.

'website' - Company Website, i.e. '<http://www.biitsoft.com>'.

Example: This example shows some specific information about your current version of Encore. In our case the output returned:

"Built with: **CMS Encore Pro**, Enterprise

Build date: 2005/01/30"

```
<head>
  <META name="generator" content="<%=GetAppInfo('')%>">
</head>

<div class="info">
  Built with: <b><%=GetAppInfo('name')%></b>,
  <%=GetAppInfo('edition')%><br>
  Build date: <%=GetAppInfo('builddate')%>
</div>
```

GetBaseFormatType function

Usage: **function** GetBaseFormatType: **String**;

Return Type: String

Description:

Returns the base format type of the output being generated.

Possible return values are: **HTML**, **RTF**, **Text**, **Other** or blank.

NOTE: RTF is the base format type for many output types including .rtf, .doc, .pdf, etc.

Example: This example displays the current output type for the current project.

The current output type is: <%=GetBaseFormatType%>

GetPublishDateTime function

Usage: `function GetPublishDateTime: DateTime;`

Return Type: DateTime

Description:

Returns the DateTime that the content was published.

Example: *This example returns the list of articles for the "News" channel, and displays the publish date for each article.*

```
<${QueryContentList('c', GetChannelIdByName('News'));}$>
<FOREACH ds="c">
  <a href="{!c._link!}">{!c._title!}</a>, published on:
  <${FormatDateTime('dddd mm yyyy mm:hh', GetPublishDateTime)}$> <br>
</FOREACH>
```

WhereAmI function

Usage: `function WhereAmI(ChannelId: Integer; Sep: String; ShowPath: Boolean; WithLink: Boolean; OtherLinkAttribs: String): String;`

ChannelId: Integer

An Integer value for a channel Id

Sep: String

String value that will be act as the separator in the bread crumb link

ShowPath: Boolean

If True will display the full path to the link

WithLink: Boolean

If True will make the breadcrumb linkable

Return Type: String

Description:

A useful function which allows to automatically create bread crumb links for a page. Pass a value of -1 for ChannelId if wishing to use the current channel id.

Example: *This example shows a link of where you currently are in the channel structure starting at channel "News", for example: "HOME > NEWS"*

```
<${WhereAmI(GetChannelIdByName('News'), ' &gt; ', False, True, '')}$>
```

WhereAmINow function

Usage: `function WhereAmINow(Sep: String; ShowPath: Boolean; WithLink: Boolean; OtherLinkAttribs: String): String;`

Sep: String

String value that will be act as the separator in the bread crumblink

ShowPath: Boolean

If True will display the full path to the link

WithLink: Boolean

If True will make the breadcrumb linkable

Return Type: String

Description:

Similar to the **WhereAmI** function, but automatically assumes to create BreadCrumb links for the current channel.

Example: *This example shows a link of where you currently are in the channel structure and also displays the name of the current article, for example: "HOME >> NEWS >> index.htm"*

```
<%=WhereAmINow(' >> ', False, True, '')$> >>  
<%=GetContentTitle(GetCurrContentId)$>
```

GetPropValue function

Usage: `function GetPropValue (PropString: String; KeyName: String): String;`

PropString: String

The String value representing the property name

KeyName: String

The name of the property to retrieve

Return Type: String

Description:

This function retrieves the values for any properties associated with templates or content, most notably user defined fields.

Example: *This example displays the property values for the associated content properties.*

```
<%=GetPropValue('Value=12'#13#10'Country=Spain', 'Country')$>  
<%=GetPropValue(QueryFieldAsString('CONTENT_PROPERTIES'), 'Value')$>
```

GV function

Usage: **function** GV(VarName: String): String;

VarName: String The name of a global variable

Return Type: String

Description:

Returns the value of a Global Variable (project property). This is the short form of the ReadGlobalVar() function.

Example: *This example displays some of the global variables in the project.*

<pre><div class="contact"> Please contact us here: <a href="<%=GV('ContactEmail')%>">info.
 Copyright <%=GV('CompanyName')%>, <%=GV('CopyrightYears')%> </div></pre>

Note: `<%=GV('PropertyName')%>` is equivalent to `{%@PropertyName%}`,

e.g. `<%=GV('CompanyName')%>` = `{%@CompanyName%}`

IsDesigning function

Usage: **function** IsDesigning: Boolean;

Return Type: Boolean

Description:

Returns True if currently busy designing a template, i.e. it's True if within the Template Editor IDE.

Example: *This example displays a different bread crumb link if the template containing this code is in design or preview mode.*

<pre><div class="bread_crumb"> <%=If (IsDesigning) or (IsPreviewing) then begin %> Bread crumb Link >> Page <%=end else begin %> <%=WhereAmINow(' >> ', False, True, '')%> >> <%=GetContentTitle(GetCurrContentId)%> <%=end;%> </div></pre>
--

IsHelpFile function

Usage: `function IsHelpFile: Boolean;`

Return Type: Boolean

Description:

Returns True if currently set to generate a Windows HTML Help File (.CHM).

Example: *The message dialog will only popup if Encore is set to create a windows help file!*

```
<$if IsHelpFile then ShowMessage('Encore is currently building the help file.');
```

IsLiveEdit function

Usage: `function IsLiveEdit: Boolean;`

Return Type: Boolean

Description:

Returns True if the current template is in LiveEdit mode

Example: *The message dialog will only popup if the current template is in LiveEdit mode.*

```
<$if IsLiveEdit then ShowMessage('Template currently in LiveEdit mode.');
```

IsLocalFileOutput function

Usage: `function IsLocalFileOutput: Boolean;`

Return Type: Boolean

Description:

Returns True if Encore's current publish location set for local file output.

Example: *The message dialog will only popup if the Encore is set to publish locally.*

```
<$if IsLocalFileOutput then ShowMessage('Encore is publishing locally.');
```

IsManual function

Usage: `function IsManual: Boolean;`

Return Type: Boolean

Description:

Returns True if Encore currently set to generate a single file Manual such as .rtf, .doc, .pdf, etc.

Example: *The message dialog will only popup if the Encore is set to publish a manual.*

```
<$if IsManual then ShowMessage('Publishing a manual file.');
```

IsPreviewing function

Usage: `function IsPreviewing: Boolean;`

Return Type: Boolean

Description:

This function will return True, if the current template is in Preview mode

Example: *This example displays a different bread crumb link if the template containing this code is in design or preview mode.*

```
<div class="bread_crumb">
  <$if (IsDesigning) or (IsPreviewing) then
    begin $>
      <a href="#">Bread crumb Link </a> >> Page
    <$end
  else
    begin $>
      <%=WhereAmINow(' >> ', False, True, '')$> >>
      <%=GetContentTitle(GetCurrContentId)$>
    <$end;$>
</div>
```

IsPublishing function

Usage: `function IsPublishing: Boolean;`

Return Type: Boolean

Description:

Returns True if Encore is currently publishing.

Example: *The message dialog will only popup if Encore is currently publishing.*

```
<$if IsPublishing then ShowMessage('Encore is currently Publishing.');
```

IsRelativeLocalPaths function

Usage: **function** IsRelativeLocalPaths: Boolean;

Return Type: Boolean

Description:

Returns True if current publish location is set to use relative path names for file output.

Example: *The message dialog will only popup if Encore is currently configured to use relative local paths.*

```
<$if IsPublishing then ShowMessage('Encore is currently using relative local paths.');
```

ReplaceMacros function

Usage: **function** ReplaceMacros(Text: String): String;

Text: String

The text to expand any macros contained within.

Return type: String

Description:

Replaces any macros (such as expanding global properties, etc.) within the specified Text string. However this does not run any script code that the Text string might contain.

Example: *This example will expand any macros, such as `{%@PageTitle%}`, etc. contained in the query field "Title".*

```
<%=ReplaceMacros(QueryFieldAsString('TITLE'))$>
```

ReplaceMacroEx function

Usage: **function** ReplaceMacroEx(Text: String): String;

Text: String

The text to expand any macros contained within.

Return type: String

Description:

Replaces any macros (such as expanding global properties, etc.) within the specified Text string, as well as running any script code that might be contained within the Text.

Example: *This example will expand any macros, such as `{%@PageTitle%}`, as well as eScript code, etc. contained in the query field "LONG_DESCRIPTION".*

```
<%=ReplaceMacroEx(QueryFieldAsString('LONG_DESCRIPTION'))$>
```

ShowMessage procedure

Usage: **procedure** ShowMessage(Text: String);

Text: String The String value to display
--

Return type: n/a

Description:

This procedure displays a modal dialog popup containing the value for Text. The user has to press Enter, or click "OK" to make the dialog close. This is useful for showing debug messages as you write your code.

Example: *This example displays a dialog displaying the message below.*

<code><\$ShowMessage('Press Enter to close this dialog!');\$></code>
--

SiteFilePath function

Usage: **function** SiteFilePath(Path: String): String;

Path: String The String value representing the website path

Return Type: String

Description:

Converts a website path to a local path if the website path is equivalent to that defined in the current project. If however publishing the site instead of only a local publish to file, then this function returns the Path unmodified.

Example:

<code><\$=SiteFilePath('http://www.biitsoft.com/abc.jpg')\$></code>

SiteFilePath2 function

Usage: **function** SiteFilePath2(Path: String): String;

Path: String The String value representing the website path

Return Type: String

Description:

Converts a website path to a local path if the website path is equivalent to that defined in the current project. If however publishing the site instead of only a local publish to file, then this function returns the Path unmodified.

Similar to the **SiteFilePath**, but this also evaluates the page name, such as 'index.htm'.

Example:

<code><\$=SiteFilePath2('http://www.biitsoft.com/index.htm')\$></code>
--

CreateOleObject function

Usage: `function CreateOleObject(ClassName: String): ComVariant;`

ClassName: String

String representing the OLE class to instantiate

Return Type: ComVariant

Description:

Allows you to automate MSWord, Excel, PowerPoint, etc. and any application that supports COM automation.

You can for example use eScript to automatically create an MSWord document containing all the articles/content in the project.

Example 1: This example creates a MS Word document containing a list of articles.

```
<$
{CMS Encore Pro - COM Example - Output to MSWord.

This is a simple example which first displays some Word system
information in a table, and then traverses through the content/
articles for the current channel and then outputs the title &
content to a Word.

To use this example, first create any new file in CMS Encore Pro,
such as in the main tree, preferably on a Channel with many articles
in it, then right-click and then choose 'New|New HTML File (.htm)'.
This could also be added via the local image/media panel to a
template/content.

Then press 'Edit' button in the Edit Media dialog, and then paste
the entire code into the HTML Editor, and then save.
Also remember to set the media type to 'Text/Exec' to ensure the
script is executed at publish time (by clicking the 'Text Contains
Executable Links/Script' checkbox).
Now when publishing when this media item is generated/published
will run the associated script here and invoke Word.

Word 9.0 (2000) or higher should be installed on your system.

To extend this example open "Visual Basic for Applications"
(Word-Macro editor) and press [F2] (object catalog) You'll find
all necessary information there.
}

var msWord: ComVariant;
var doc, sel, tab: ComVariant;

// Create a instance of Word 9.0
msWord := CreateOleObject('Word.Application');

// Display Window
msWord.Visible := True;

// Add a new empty document
doc := msWord.Documents.Add();
// Get the current selection (cursor position)
sel := msWord.Selection;
```

```

// Add some text
sel.TypeText('Some System Parameters:');
sel.HomeKey(5 {wdLine}, 1 {wdExtend});

// Some simple formatting
sel.Font.Bold := True;

sel.EndKey(5 {wdLine}, 0 {wdMove});
sel.Font.Bold := False;
sel.TypeParagraph();

// Create a table
tab := doc.Tables.Add(sel.Range, 3, 2);

// Add text to the table cells
sel.TypeText('Operating System');
sel.MoveRight(12 {wdCell});
sel.TypeText(msWord.System.OperatingSystem);
sel.MoveLeft(12 {wdCell});
sel.MoveDown(5 {wdLine});
sel.TypeText('Processor');
sel.MoveRight(12 {wdCell});
sel.TypeText(msWord.System.ProcessorType);
sel.MoveLeft(12 {wdCell});
sel.MoveDown(5 {wdLine});

sel.TypeText('Word Version');
sel.MoveRight(12 {wdCell});
sel.TypeText(MsWord.Version);
sel.MoveLeft(12 {wdCell});
sel.MoveDown(5 {wdLine});

//--- Here we start the output of our content ---
sel := msWord.Selection;
sel.TypeParagraph();

var strTitle, strContent : string;$>
<FOREACH type="content" ds="c">
  <$
    strTitle := QueryFieldAsString('c.TITLE');
    strContent := RemoveHTMLTags(QueryFieldAsString('c._CONTENT_TEXT_'));

    sel.TypeText(strTitle);
    sel.HomeKey(5 {wdLine}, 1 {wdExtend});

    // Some simple formatting
    sel.Font.Bold := True;
    sel.EndKey(5 {wdLine}, 0 {wdMove});
    sel.Font.Bold := False;
    sel.TypeParagraph();
    sel.TypeParagraph();
    sel.TypeText(strContent);
    sel.TypeParagraph();
  $>
</FOREACH>

```

Example 2: This example traverses an access database ("DBDemos.mdb") and then displays some data from the database. This sort of functionality can be used to insert articles into Encore from an external database source.

```
<${
CMS Encore Pro - COM Example - ADO.

This is a simple example which connects to an ADO dataset, and
then traverses through the dataset returning some field values.

To use this example simply paste this script anywhere inside
a template, when the template is used will then run this script.

This script can also be used in local or global image/media files,
but remember to set the media type to 'Text/Exec' to ensure the
script is executed at publish time (by clicking the 'Text Contains
Executable Links/Script' checkbox).
}

var conn: ComVariant;
var rs: ComVariant;

try
  conn := CreateOleObject('ADODB.Connection');
  conn.ConnectionString := 'Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=DbDemos.mdb;Persist Security Info=False';
  conn.Open();

  rs := conn.Execute('select * from Customer');
  rs.MoveFirst();

  while not (rs.EOF = True) do
  begin
    SendLn(rs.Fields('Company').Value + '<br>');
    rs.MoveNext();
  end;

except
  var x, cnt: Integer;
  cnt := conn.Errors.Count;
  for x := 0 to cnt - 1 do
    SendLn(conn.Errors.Item(0).Description + '<br>');
  end;
$>
```

CleanupGlobalVars procedure

Usage: `procedure CleanupGlobalVars;`

Return Type: n/a

Description:

This procedure resets all of the internal internal variables.

Example: *This example resets the internal variables.*

```
Caution, all internal variables will now be reset:  
<${CleanupGlobalVars;}>
```

NB: It is not recommended to use this procedure because it will also reset the internal variables used by Encore, and therefore could have adverse effects on your project.

ReadGlobalVar function

Usage: `function ReadGlobalVar(n: String): Variant;`

```
n: String  
The name of a global variable
```

Return Type: String

Description:

Returns the value of a Global Variable (project property).

Example: *This example displays some global variables.*

```
<div class="contact">  
  Please contact us here:  
  <a href="mailto:<${ReadGlobalVar('ContactEmail')}$>">info</a>.<br>  
  Copyright <${ReadGlobalVar('CompanyName')}$>,  
    <${ReadGlobalVar('CopyrightYears')}$>  
</div>
```

Note: The "ReadGlobalVar" function is equivalent to using the "GV" function.

ReadGlobalVarDef function

Usage: `function ReadGlobalVarDef(n: String; d: Variant): Variant;`

<p>n: String The name of a global variable</p> <p>d: Variant The default value to assign to the global variable</p>

Return Type: Variant

Description:

Similar to the "ReadGlobalVar" procedure. Returns the value of a Global Variable (project property), if the global variable is not defined, the function returns the default value specified by "d"

Example: *This example displays some global variables, and displays the default values if the global variables are not defined. In the example below, we have deliberately misspelled the global variables name (by adding an "X" at the end of the name), so that the default values would be used.*

<pre><div class="contact"> Please contact us here: <a href="mailto:<%=ReadGlobalVarDef('ContactEmailX', 'robert@bobsautoworld.com')%>">info.
 Copyright <%=ReadGlobalVarDef('CompanyNameX', 'Roberts Auto World')%>, <%=ReadGlobalVarDef('CopyrightYearsX', '1999-2005')%> </div></pre>
--

WriteGlobalVar procedure

Usage: `procedure WriteGlobalVar(n: String; v: Variant);`

<p>n: String The name of a global variable</p> <p>v: Variant The value to assign to the global variable</p>

Return Type: n/a

Description:

This function assigns a value to a global variable. The value of the global variable gets reset the next time you load your project.

Example: *This examples assigns the value "contact@bobsautoworld.com" to the global variable "ContactEmail".*

<pre><%=WriteGlobalVar('ContactEmail', 'contact@bobsautoworld.com');%></pre>
--

8. Anchor Related Functions and Procedures:

The <ANCHOR> tag allows for specifying anchor text which will automatically get inserted at a predefined anchor point in the html after a page is generated. The <ANCHOR> tags are wrappers around the functions AddToAnchor(), AddUniqueToAnchor(), etc. as described below.

The most useful purpose of anchors is to allow specifying text in templates/components that will automatically get inserted into the <HEAD>...</HEAD> or <BODY...> sections of a page even though the template or component might not have direct access to the final page where those tags has been defined.

This is very useful when writing javascript or DHTML code.

GetAnchor function

Usage: **function** GetAnchor(Name: String): **String**;

Name: String The name of the anchor to retrieve.
--

Return Type: String

Description:

Returns the current value of an Anchor defined by 'Name'.

AddToAnchor function

Usage: **function** AddToAnchor(Name: String; Value: String): **Boolean**;

Name: String The name of the anchor to use.
Value: String The value to add to the anchor.

Return Type: Boolean

Description:

Adds a value to the Anchor defined by 'Name', and returns True if the anchor exists, else returns False.

AddUniqueToAnchor function

Usage: **function** AddUniqueToAnchor(Name: String; Value: String): **Boolean**;

Name: String The name of the anchor to use.
Value: String The value to add to the anchor.

Return Type: Boolean

Description:

Adds a unique value to the Anchor defined by 'Name', and returns True if the anchor exists, else returns False.

AnchorExists function

Usage: **function** AnchorExists(Name: String): Boolean;

Name: String
The name of the anchor to search for.

Return Type: Boolean

Description:

Returns True if an Anchor defined by 'Name' already exists, else returns False.

ClearAllAnchors procedure

Usage: **procedure** ClearAllAnchors;

Return Type: n/a

Description:

Clears all defined anchors.

ClearAnchor function

Usage: **function** ClearAnchor(Name: String): Boolean;

Name: String
The name of the anchor to clear.

Return Type: Boolean

Description:

Clears the anchor defined by 'Name', and returns True if the anchor exists, else returns False.

SetAnchor function

Usage: **function** SetAnchor(Name: String; Value: String): Boolean;

Name: String
The name of the anchor to use.

Value: String
The value to add to the anchor.

Return Type: Boolean

Description:

Sets the value of an anchor defined by 'Name', and returns True if the anchor exists, else returns False.

Example:

```
<$AddToAnchor('+META', '<META name="AUTHOR" content="User One">');$>
...
...
<$=GetAnchor('META')$>
```

9. Logical (Conditional) Related Functions and Procedures:

IIF function

Usage: `function IIF(Condition: Boolean; TrueValue: String; FalseValue: String): String;`

Condition: Boolean

A Boolean condition to test for

TrueValue: String

If Condition evaluates to True, then return this String value

FalseValue: String

If Condition evaluates to False, then return this String value

Return Type: String

Description:

If condition is True then returns value associated with TrueValue, else returns value associated with FalseValue. Condition can be any complex Boolean statement.

Example: *This example evaluates the expression below.*

```
<${SendLn(IIF(5 > 2, '5 is greater than 2', '5 is not greater than 2'))}$>
```

10. Glossary/WordScan Related Functions and Procedures:

Encore supports the ability to replace words in an article with a user defined structure.

For example, you may wish to replace all occurrences of a certain word with a url, or acronym, etc.

The "GlossaryWordScan" and "GlossaryWordScanAdv" functions allow you to do this.

Using replace filers and a word list, you can modify the list of words you specify to anything you wish.

For an example, see the "Hints/Tips" section on the BIITSoft Forum

GlossaryWordScan function

Usage: `function GlossaryWordScan(SourceText: String; ReplaceFilter: String; WordList: String; ExcludedWordList: String; FindFirstOnly: Boolean): String;`

SourceText: String

The that should be parsed

ReplaceFilter: String

Are the "structures" that tell Encore what to do when it finds a word that it should replace.

WordList: String

The list of words to replace

ExcludedWordList: String

The list of words to exclude

FindFirstOnly: Boolean

If True, then only the first instance of a word will be replaced

Return Type: String

Description:

Takes the text you wish to check and searches for words in the text that match words in a specified word list. It then replaces the words, using a replace filter.

Additionally you can specify whether to replace only the first occurrence of the word.

Example:

```
<%=GlossaryWordScan(QueryFieldAsString('_CONTENT_TEXT_'),
'<a href="##value##">##text##</a>',
'Microsoft=http://www.microsoft.com'+#13#10+
'BIITSoft=http://www.biitsoft.com',
'', False)%>
```

This example will replace the words "Microsoft" and "BIITSoft" with "[Microsoft](http://www.microsoft.com)" and "[BIITSoft](http://www.biitsoft.com)" in an article respectively.

GlossaryWordScanAdv function

Usage: `function GlossaryWordScanAdv(SourceText: String; ReplaceFilter: String; WordList: String; ExcludedWordList: String; FindFirstOnly: Boolean; MatchCase: Boolean; SkipTags: Boolean; MatchFilter: String; var MatchesFound: String): String;`

<p>SourceText: String The that should be parsed</p> <p>ReplaceFilter: String Are the "structures" that tell Encore what to do when it finds a word that it should replace.</p> <p>WordList: String The list of words to replace</p> <p>ExcludedWordList: String The list of words to exclude</p> <p>FindFirstOnly: Boolean If True, then only the first instance of a word will be replaced</p> <p>MatchCase: Boolean If True, then only case matching words will be replaced</p> <p>SkipTags: Boolean If True then all words within HTML tags will be skipped</p> <p>MatchFilter: String Similar to ReplaceFilter, but it displays a summary of the words that have been replaced</p> <p>MatchesFound: String Returns the total number of matches found</p>

Return Type: String

Example:

```
<$var strMatches : string;$>
<${GlossaryWordScanAdv(QueryFieldAsString('_CONTENT_TEXT_'),
  '<acronym title="##value##">##text##</acronym>',
  ' eScript=Encore Scripting Language'+#13#10+
  'DWS=Delphi Web Script'+#13#10+
  'OOP=Object Oriented Programming',
  '',
  False, False, False,
  '##index## - <b>##keyword##</b> - ##value##: ##descrip##
  (##count## match(es))##descrip2##<br>',
  strMatches)$>

Total Matches=<${strMatches$>
```

NOTE: Currently the keyword can only consist of single words, i.e. 'Windows' is valid, but 'Microsoft Windows' is invalid - it won't give an error, but it won't find the keyword if it's made up of multiple words.

Supported filter symbols:

- ##value##** - value associated with keyword.
- ##descrip##** - first description associated with keyword (separated by '|' char).
- ##descrip2##** - second description associated with keyword (separated by '|' char).
- ##text##** - same as keyword but contains text as it appears in the source.
- ##keyword##** - keyword as stored in word list.
- ##index##** - index of item in matched list.
- ##count##** - current count of matched keyword.
- ##total##** - current total number of matched keywords.

11. LiveEdit Related Functions and Procedures:

The LiveEdit capabilities of Encore allow you to edit content/content links/channel links in your templates visually. The "LiveEdit" links indicate areas of the template that are editable. Users view the template in the WYSIWYG preview, and then by clicking the LiveEdit links they can then edit that part of the template (page).

LiveEditContent function

Usage: `function LiveEditContent(Action: String; ContentOrChannelId: Integer; Message: String): String;`

Action: String

A string representing action to be taken by Encore when the user clicks the Link. Action can have the following values:

- 'add' - Adds new content to the channel specified by ContentOrChannelId. It's LiveEdit marker is displayed as (+).
- 'edit' - Edits the content specified by ContentOrChannelId. It's LiveEdit marker is displayed as (#).
- 'delete' - Deletes the content specified by ContentOrChannelId. It's LiveEdit marker is displayed as (-).

ContentOrChannelId: Integer

An integer representing the ChannelId of the Channel to edit or the ContentId of the content to update

Message: String

A string value to display with the LiveEdit link

Return Type: String

Description:

Performs a LiveEditContent action when the associated link is clicked in LiveEdit mode. The 'Message' parameter is an optional message that you want to display with the action link when LiveEdit is active.

Example: *The examples below allow you to edit and delete the property "TopStory_Content" using LiveEdit.*

```
<%=LiveEditContent('edit', {%.TopStory_Content.}##}, 'Change')$>  
<%=LiveEditContent('delete', {%.TopStory_Content.}##}, '')$>  
<%=GetImageSrc({%.TopStory_Content.}##}, -1, 0, False, False, '')$>
```

LiveEditProp function

Usage: **function** LiveEditProp(Property: String; Message: String): **String**;

Property: String

The name of the property to edit

Message: String

A string value to display with the LiveEdit link

Return Type: String

Description:

Allows editing a channel published property which has the syntax `'(.property.)'` when the associated link is clicked in LiveEdit mode. The 'Message' parameter is an optional message that you want to display with the link below when LiveEdit is active.

It's LiveEdit marker is displayed as `(*)`.

Example:

```
<%=LiveEditProp('(.FirstStory_Content.)', '')$>  
<esp:Summary_Story Story="%(.FirstStory_Content.)%" />
```

LiveEditPropT function

Usage: **function** LiveEditPropT(Property: String; Message: String): **String**;

Property: String

The name of the property to edit

Message: String

A string value to display with the LiveEdit link

Return Type: String

Description:

Allows editing any template property when the associated link is clicked in LiveEdit mode in the Template Editor IDE or using LiveEdit on a Template rather than a channel. The 'Message' parameter is an optional message that you want to display with the link below when LiveEdit is active.

It's LiveEdit marker is displayed as `(*)`.

Example:

```
<%=LiveEditPropT('Headline_Color', '<small>[Change Color]</small>')$>
```

12. Image/Media Functions:

This set of functions deal with images/media and their retrieval.

GetCurrMediaId function

Usage: `function GetCurrMediaId: Integer;`

Return Type: Integer

Description:

Returns the current Media Id for the current media.

Example: *This example displays the Id of the current image/media item.*

```
<${SendLn(GetCurrMediaId);$>
```

GetImageSrc function

Usage: `function GetImageSrc(ContentId: Integer; SeqNo: Integer; Border: Integer; Thumbnail: Boolean; WithLink: Boolean; OtherAttribs: String): String;`

ContentId: Integer

The Id of the content item in which to retrieve the image

SeqNo: Integer

The sequence number corresponding to the image to retrieve.

Border: Integer

The thickness of the border, 0 = no border

Thumbnail: Boolean

If True, then the function will return the thumbnail version of the image

WithLink: Boolean

Provides a link to the original image, if Thumbnail is True

OtherAttribs: String

Allows you to specify extra formatting for example class code etc

Return Type: String

Description:

Returns an image corresponding to the content/article item specified by ContentId, or full IMG tag, where SeqNo is the sequence number of the image/media to retrieve. A value of -1 for SeqNo will reference the first image/media.

Example:

```
<!-- This function displays the 12th image for an article with ContentId = 25. -->
```

```
<${GetImageSrc(25, 12, 0, False, True, 'class="mystyle"')}$>
```

GetMediaLink function

Usage: `function GetMediaLink(MediaId: Integer): String;`

MediaId: Integer

The Id of the media item to link to.

Return Type: String

Description:

Returns the full media link/path defined by MediaId.

Example:

Click here to view `<a href="<%=GetMediaLink(1)%>">Readme.pdf`

GetAbsMediaLink function

Usage: `function GetAbsMediaLink(MediaId: Integer): String;`

MediaId: Integer

The Id of the media item to link to.

Return Type: String

Description:

Returns the absolute media link/path defined by MediaId relative to the root path defined by AbsLinkPrefix (default = '').

Example:

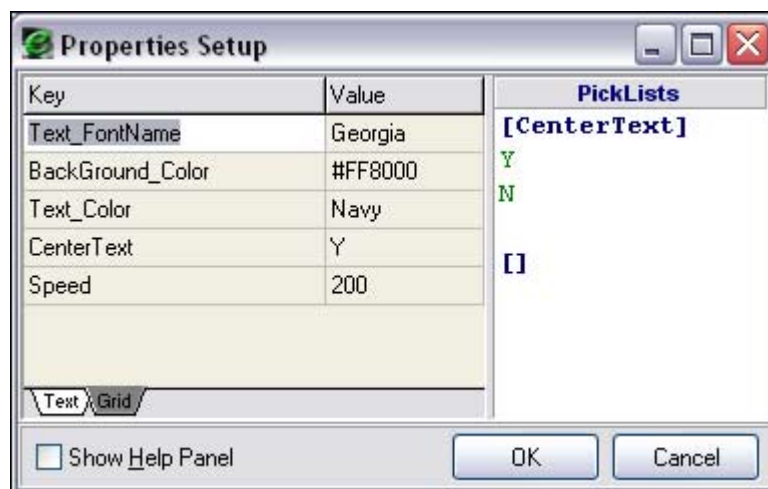
Click here to view `<a href="<%=GetAbsMediaLink(1)%>">Readme.pdf`

13. Imaging Template Functions (only valid on Image Templates):

These functions/procedures deal with images and image manipulation. Encore supports image manipulation via image templates, allowing you to create sophisticated image effects like animated gif's etc. These functions may be expanded upon in a separate document.

Note: There is a detailed example function at the end of this section which references most of these functions. There you will find a practical application for some of these functions.

```
procedure ImageAssign;
procedure ImageClear;
procedure ImageClearBackground(BgColor: Integer);
procedure ImageEffects(Effects: String);
procedure ImageEllipse(X1, Y1, X2, Y2, PenColor, PenWidth, BgColor:
Integer);
function ImageGetHeight: Integer;
function ImageGetPixel(X: Integer; Y: Integer): Integer;
function ImageGetTextHeight(Text: String): Integer;
function ImageGetTextWidth(Text: String): Integer;
function ImageGetWidth: Integer;
procedure ImageGifAnimateAddFrame(DelayMs: Integer);
procedure ImageGifAnimateBegin;
procedure ImageGifAnimateEnd;
procedure ImageGradientBackground(Color1, Color2: Integer; IsHorizontal:
Boolean);
procedure ImageInit(Width, Height, BgColor: Integer);
function ImageIsAllowed: Boolean;
procedure ImageLine(X1, Y1, X2, Y2, PenColor, PenWidth: Integer);
procedure ImageLineTo(X, Y, PenColor, PenWidth: Integer);
procedure ImageLoadFromFile(Filename: String);
procedure ImageMoveTo(X, Y: Integer);
procedure ImageRectangle(X1, Y1, X2, Y2, PenColor, PenWidth, BgColor:
Integer);
procedure ImageSaveToFile(Filename: String);
procedure ImageSetFont(FontName: String; FontSize: Integer; FontColor:
Integer; FontStyle: String);
procedure ImageSetHeight(Height: Integer);
procedure ImageSetPixel(X: Integer; Y: Integer; Color: Integer);
procedure ImageSetWidth(Width: Integer);
procedure ImageSetWidthHeight(Width: Integer; Height: Integer);
procedure ImageTextOutXY(X: Integer; Y: Integer; Text: String);
procedure ImageTextOutXYAdv(X: Integer; Y: Integer; Text: String;
ShadowColor: Integer; Opacity: Integer);
```



This image template example creates an animated gif button from a channel name.

Note for the example to work you must set up properties like in the screenshot above.

```
<$//Creates an animated gif banner from current channel name
var strChannelId : string = IntToStr(ReadGlobalVar('CHANNEL_ID'));

WriteGlobalVar('NEW_MEDIA_MAKE_BUTTON', 'N');
WriteGlobalVar('NEW_MEDIA_TITLE', strChannelId + 'ani_banner');
WriteGlobalVar('NEW_MEDIA_TYPE', 'Image');
WriteGlobalVar('NEW_MEDIA_DESCRIPTION', '');
WriteGlobalVar('NEW_MEDIA_EXT', '.gif');
WriteGlobalVar('NEW_MEDIA_DISPLAY_ORDER', '');
WriteGlobalVar('NEW_MEDIA_ALT_TEXT', '');
WriteGlobalVar('NEW_MEDIA_LONG_DESCRIPTION', '');
WriteGlobalVar('NEW_MEDIA_GROUP', '');
WriteGlobalVar('NEW_MEDIA_WIDTH', '');
WriteGlobalVar('NEW_MEDIA_HEIGHT', '');
WriteGlobalVar('NEW_MEDIA_SIZE', '');

var strChannelName : string = ReadGlobalVar('CHANNEL_NAME');

ImageInit(468, 60, HTMLColorStringToColor({%BackGround_Color%}));
//ImageLoadFromFile('*test');
ImageSetFont({%Text_FontName%}, 20, $ABCDEF, 'BU');
var s : string = strChannelName;
var i, cx, cy : integer;

ImageGifAnimateBegin;
try
  for i := 1 to 10 do
    begin
      ImageGradientBackground(HTMLColorStringToColor({%BackGround_Color%}),
        $FFFFFF, True);
      ImageSetFont({%Text_FontName%}, 4 + (i * 4), $FFFFFF, 'B');

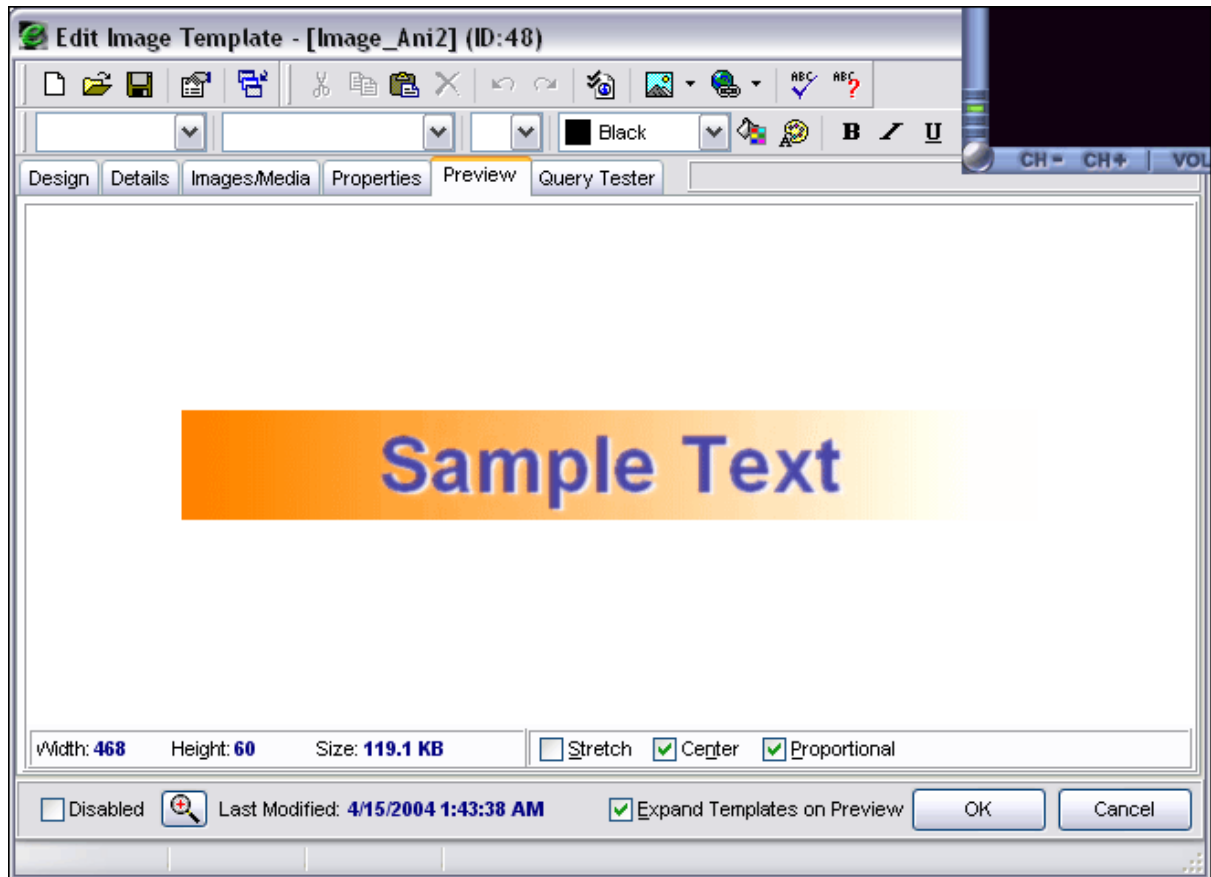
      if ({%CenterText%} = 'Y') then
        begin
          //Calculate centers
          cx := (ImageGetWidth div 2) - (ImageGetTextWidth(s) div 2);
          cy := (ImageGetHeight div 2) - (ImageGetTextHeight(s) div 2);
        end
      else
        begin
          cx := i * 20;
          cy := i * 3;
        end;

      ImageTextOutXY(cx + 2, cy + 1, s);
      ImageSetFont({%Text_FontName%}, 4 + (i * 4),
HTMLColorStringToColor({%Text_Color%}), 'B');
      ImageTextOutXYAdv(cx, cy, s, -1, i * 10);

      ImageEffects('antialias=25');
      //ImageEffects('sharpen=25');
      ImageEffects('contrast=' + IntToStr(i * 5));

      ImageGifAnimateAddFrame({%Speed%});
    end;
  finally
    ImageGifAnimateEnd;
    //ImageSaveToFile('c:\test_.gif');
  end;
$>
```

This results in the following animated gif.



14. File - I/O Related Functions/Procedures:

These functions deal with files and folder manipulation.

AppendStringToFile procedure

Usage: **procedure** AppendStringToFile(fileName: String; data: String);

<p>fileName: String The name of the file to operate on</p> <p>data: String The data to append to the file</p>

Return type: n/a

Description:

This function will append the string data to the end of the file specified by fileName.

ChangeFileExt function

Usage: **function** ChangeFileExt(fName: String; ext: String): **String**;

<p>fName: String The name of the file to operate on</p> <p>ext: String The new extension to assign to the file</p>
--

Return type: String

Description:

Takes the file name passed in FileName and changes the extension of the file name to the extension passed in Extension. Extension specifies the new extension, including the initial dot character.

ChDir procedure

Usage: **procedure** ChDir(s: String);

<p>s: String The name of the directory to change to</p>
--

Return type: n/a

Description:

ChDir changes the current directory to the path specified by S.

CreateDir function

Usage: **function** CreateDir(dir: String): **Boolean**;

dir: String The name of the directory to create

Return type: Boolean

Description:

CreateDir creates a new directory. The return value is True if a new directory was successfully created, or False if an error occurred.

DeleteFile function

Usage: **function** DeleteFile(fileName: String): **Boolean**;

fileName: String The name of the file to delete

Return type: Boolean

Description:

Deletes the file named by FileName from the disk. If the file cannot be deleted or does not exist, the function returns False.

ExtractFileDir function

Usage: **function** ExtractFileDir(fName: String): **String**;

fName: String The name of the file from which to extract the directory name

Return type: String

Description:

Extracts the drive and directory parts from FileName. The result is empty if FileName contains no drive and directory parts.

ExtractFileDrive function

Usage: **function** ExtractFileDrive(fName: String): **String**;

fName: String The name of the file from which to extract the file drive name
--

Return type: String

Description:

Returns the drive portion of a file name.

ExtractFileExt function

Usage: **function** ExtractFileExt(fName: String): **String**;

fName: String

The name of the file from which to extract the file extension

Return type: String

Description:

Returns the extension portions of a file name, the result is empty if the given file name has no extension.

ExtractFileName function

Usage: **function** ExtractFileName(fName: String): **String**;

fName: String

The name of the file from which to extract the file extension

Return type: String

Description:

Extracts the name and extension parts of a file name.

ExtractFilePath function

Usage: **function** ExtractFilePath(fName: String): **String**;

fName: String

The name of the file from which to extract the file path

Return type: String

Description:

Returns the drive and directory portions of a file name.

FileExists function

Usage: **function** FileExists(fileName: String): **Boolean**;

fileName: String

The name of the file to check

Return type: Boolean

Description:

FileExists returns True if the file specified by FileName exists. If the file does not exist, then returns False.

FileSearch function

Usage: **function** FileSearch(name: String; dirList: String): **String**;

<p>name: String The name of the file to search for</p> <p>dirList: String A semicolon delimited list of directories</p>

Return type: String

Description:

Searches through the directories passed in DirList for a file named Name. DirList is a list of path names delimited by semicolons. If FileSearch locates a file matching Name, it returns a string specifying a path name for that file. If no matching file exists, FileSearch returns an empty string.

GetCurrentDir function

Usage: **function** GetCurrentDir: **String**;

Return type: String

Description:

Returns the name of the current directory.

RemoveDir function

Usage: **function** RemoveDir(dir: String): **Boolean**;

<p>dir: String The name of the directory to remove</p>

Return type: Boolean

Description:

Removes the directory specified by the Dir parameter. The return value is True if a new directory was successfully deleted, False if an error occurred. The directory must be empty before it can be successfully deleted.

RenameFile function

Usage: **function** RenameFile(oldName: String; newName: String): **Boolean**;

<p>oldName: String The name of the file to rename</p> <p>newName: String The new name to assign to the file</p>

Return type: Boolean

Description:

Attempts to change the name of the file specified by OldFile to NewFile. If the operation succeeds, RenameFile returns True. If RenameFile cannot rename the file (for example, if the application does not have permission to modify the file), it returns False.

SaveStringToFile procedure

Usage: **procedure** SaveStringToFile(fileName: String; data: String);

fileName: String The name of the file to save
data: String The data to save in the file

Return type: n/a

Description:

Saves the contents of data, to a file specified by filename.

SetCurrentDir function

Usage: **function** SetCurrentDir(dir: String): Boolean;

dir: String The name of the directory to change to
--

Return type: Boolean

Description:

Sets the current directory. The return value is True if the current directory was successfully changed, or False if an error occurred.

Example: *This is a minimal File - I/O example using some of the functions from this section. This example extracts the articles from the "News" channel and outputs it to a file.*

```
<$
QueryContentList('c', GetChannelIdByName('News'));
var ArticleList: TStringList;
ArticleList := TStringList.Create;
try $>
  <FOREACH ds="c">
    <$ArticleList.Add(QueryFieldAsString('c.CONTENT_TEXT'));
    ArticleList.Add('-----');$>
  </FOREACH>
  <$ArticleList.Add('<br><br>');
  var str : string = '';
  if FileExists('c:\temp.txt') then
    str := LoadStringFromFile('c:\temp.txt');
  str := str + ArticleList.Text;
  SaveStringToFile('c:\temp.txt', str);
  //AppendStringToFile('c:\temp.txt', ArticleList.Text);
  //DeleteFile('c:\temp.txt');
finally
  ArticleList.Free;
end;$>
```

Tip: This example could also be extended to create a .rss feed file.

15. Mathematical/Statistical Functions/Procedures:

This group of functions are for mathematical/statistical purposes. They are listed briefly.

Abs function

Usage: `function Abs(X: Float): Float;`

X: Float
A Float value.

Return Type: Float

Description:

Returns the absolute value of the argument.

ArcCos function

Usage: `function ArcCos(v: Float): Float;`

V: Float
A Float value.

Return Type: Float

Description:

Returns the inverse cosine of V. V must be between -1 and 1. The return value is in the range $[0..Pi]$, in radians

ArcCosh function

Usage: `function ArcCosh(v: Float): Float;`

V: Float
A Float value.

Return Type: Float

Description:

Returns the inverse hyperbolic cosine of V. The value of V must be greater than or equal to 1.

ArcSin function

Usage: `function ArcSin(v: Float): Float;`

V: Float
A Float value.

Return Type: Float

Description:

Returns the inverse sine of V. V must be between -1 and 1. The return value will be in the range $[-Pi/2..Pi/2]$, in radians.

ArcSinh function

Usage: `function ArcSinh(v: Float): Float;`

V: Float
A Float value.

Return Type: Float

Description:

Returns the inverse hyperbolic sine of V.

ArcTan function

Usage: `function ArcTan(v: Float): Float;`

V: Float
A float representing an angle in Radians

Return Type: Float

Description:

Returns the arctangent of V. V is an expression that represents an angle in radians.

ArcTanh function

Usage `function ArcTanh(v: Float): Float;`

V: Float
A Float value.

Return Type: Float

Description:

Returns the inverse hyperbolic tangent of V. The value of V must be between -1 and 1.

Cos function

Usage: `function Cos(a: Float): Float;`

A: Float
A float representing an angle in Radians

Return Type: Float

Description:

Returns the cosine of the angle A. A is an expression that represents an angle in radians.

Cosh function

Usage: **function** Cosh(a: Float): Float;

A: Float
A Float value.

Return Type: Float

Description:

Returns the hyperbolic cosine of A.

Cotan function

Usage: **function** Cotan(a: Float): Float;

A: Float
A Float value.

Return Type: Float

Description:

Returns the cotangent of A. The cotangent is calculated using the formula $1 / \tan(A)$

Note: Do not call Cotan with $X = 0$.

DegToRad function

Usage: **function** DegToRad(a: Float): Float;

A: Float
A float representing an angle in Degrees

Return Type: Float

Description:

Convert angles expressed in degrees to the corresponding value in radians, where radians = degrees(pi/180).

Note: That most of the angle functions require their angles to be in radians.

Exp function

Usage: **function** Exp(v: Float): Float;

V: Float
A Float value.

Return Type: Float

Description:

Returns the value of e raised to the power of V, where e is the base of the natural logarithms.

Frac function

Usage: **function** Frac(v: Float): Float;

V: Float
A Float value.

Return Type: Float

Description:

Returns the fractional part of the argument V. The result is the fractional part of X; that is, $\text{Frac}(X) = X - \text{Int}(X)$.

Hypot function

Usage: **function** Hypot(x: Float; y: Float): Float;

X: Float
A Float value.

Y: Float
A Float value.

Return Type: Float

Description:

Returns the length of the hypotenuse of a right angle triangle. Specify the lengths of the sides adjacent to the right angle in X and Y. Hypot uses the formula $\text{Sqrt}(X^2 + Y^2)$

Inc function

Usage: **function** Inc(var a: Integer; b: Integer): Integer;

A: Integer
An Integer value.

B: Integer
An Integer value to add to A

Return Type: Integer

Description:

Increments the value of A, by B.

Int function

Usage: **function** Int(v: Float): Float;

V: Float
A Float value.

Return Type: Float

Description:

Returns the integer part of X; that is, X rounded toward zero.

IntToHex function

Usage: `function IntToHex(v: Integer; digits: Integer): String;`

V: Integer

An Integer value to convert.

Digits: Integer

The number of hexadecimal digits to return

Return Type: String

Description:

Converts a number into a string containing the number's hexadecimal (base 16) representation. V is the number to convert. Digits indicates the minimum number of hexadecimal digits to return.

Ln function

Usage: `function Ln(v: Float): Float;`

V: Float

A Float value.

Return Type: Float

Description:

Returns the natural logarithm ($\ln(e) = 1$) of the expression V.

Log10 function

Usage: `function Log10(v: Float): Float;`

V: Float

A Float value.

Return Type: Float

Description:

Returns the log base 10 of V.

Log2 function

Usage: `function Log2(v: Float): Float;`

V: Float

A Float value.

Return Type: Float

Description:

Returns the log base 2 of V.

LogN function

Usage: **function** LogN(n: Float; x: Float): **Float**;

N: Float
The base value

X: Float
A float value

Return Type: Float

Description:

Returns the log of X for a specified base.

Max function

Usage: **function** Max(v1: Float; v2: Float): **Float**;

V1: Float
A float value

V2: Float
A float value

Return Type: Float

Description:

Returns the greater value of V1 and V2.

Min function

Usage: **function** Min(v1: Float; v2: Float): **Float**;

V1: Float
A float value

V2: Float
A float value

Return Type: Float

Description:

Returns the smaller value of V1 and V2.

Pi function

Usage: **function** Pi: **Float**;

Return Type: Float

Description:

Use Pi in to represent the mathematical value **pi**, the ratio of a circle's circumference to its diameter. Pi is approximated as 3.1415926535897932385.

Power function

Usage: **function** Power(base: Float; exponent: Float): **Float**;

Base: Float

A number to raise

Exponent: Float

The value to raise to

Return Type: Float

Description:

Power raises Base to any power.

RadToDeg function

Usage: **function** RadToDeg(a: Float): **Float**;

A: Float

A float value representing a radian

Return Type: Float

Description:

Convert angles measured in radians to degrees, where degrees = radians(180/pi).

RandG function

Usage: **function** RandG(mean: Float; stdDev: Float): **Float**;

Mean: Float

A float value representing the median of a distribution

StdDev: Float

A float representing the standard deviation

Return Type: Float

Description:

Produces random numbers with Gaussian distribution about the Mean. This is useful for simulating data with sampling errors and expected deviations from the Mean.

Random function

Usage: **function** Random: **Float**;

Return Type: Float

Description:

Returns a random number within the range $0 \leq X < 1$.

RandomInt function

Usage: `function RandomInt(range: Integer): Integer;`

Range: Integer

An Integer value specifying the range

Return Type: Integer

Description:

Returns a random number within the range $0 \leq X < \text{Range}$.

Randomize procedure

Usage: `procedure Randomize;`

Return Type: n/a

Description:

Initializes the built-in random number generator with a random value (obtained from the system clock). The random number generator should be initialized by making a call to Randomize, or by assigning a value to RandSeed.

Do not combine the call to Randomize in a loop with calls to the Random function. Typically, Randomize is called only once, before all calls to Random.

RandSeed function

Usage: `function RandSeed: Integer;`

Return Type: Integer;

Description:

By assigning a specific value to RandSeed, you can use the Random function to repetitively generate a specific sequence of random numbers.

Round function

Usage: `function Round(v: Float): Integer;`

V: Float

A float value to round

Return Type: Float

Description:

Returns the value of X rounded to the nearest whole number.

SetRandSeed procedure

Usage: procedure **SetRandSeed**(seed: Integer);

seed: Integer

The seed of the random number generator.

Return Type: n/a

Description:

Sets the seed of the random number generator.

Sin function

Usage: **function Sin**(a: Float): **Float**;

A: Float

A float representing an angle in Radians

Return Type: Float

Description:

Returns the sine of the angle in radians.

Sinh function

Usage: **function Sinh**(a: Float): **Float**;

A: Float

A float value

Return Type: Float

Description:

Returns the hyperbolic sine of an angle.

Sqr function

Usage: **function Sqr**(v: Float): **Float**;

V: Float

A float value

Return Type: Float

Description:

Returns the square of a V, where $\text{sqr}(V) = V \cdot V$

Sqrt function

Usage: **function Sqrt**(v: Float): **Float**;

V: Float

A float value

Return Type: Float

Description:

Returns the square-root of a V.

Tan function

Usage: `function Tan(a: Float): Float;`

A: Float
A float representing an angle in Radians

Return Type: Float

Description:

Returns the Tangent of the angle A, where A is in radians

Tanh function

Usage: `function Tanh(a: Float): Float;`

A: Float
A float value

Return Type: Float

Description:

Returns the hyperbolic tangent of X.

Trunc function

Usage: `function Trunc(v: Float): Integer;`

V: Float
A float value to truncate

Return Type: Integer

Description:

Truncates a float to an integer, i.e. rounds V to zero.

Example:

```
<$var RandomNum, i: Integer;  
  //Generate a random number between 1 and 10  
  RandomNum := RandomInt(10) + 1;$>  
  x = <%=RandomNum%>, y = 1..10 <br>  
  <table border="1" width="80%" align="center">  
    <tr bgcolor="#abcdef">  
      <th>y</th>  
      <th>x<sup>y</sup></th>  
      <th>Ln<sub>x</sub>(y)</th>  
      <th>Hex(x)</th>  
    </tr>  
    <$for i := 1 to 10 do  
      begin $>  
        <tr>  
          <td><%=i%></td>  
          <td><%=RandomNum%><sup><%=i%></sup>=  
            <%=Power(RandomNum, i)%></td>  
          <td>Ln<sub><%=i%></sub>(<%=RandomNum%>)=  
            <%=LogN(RandomNum, i)%></td>  
          <td align="right"><%=IntToHex(RandomNum, i)%></td>  
        </tr>  
      <$end;$>  
    </table>
```

end.