IES DOCUMENT WRITER: TAG LOGIC

What is it?

Tag Logic is the syntax in which we describe to Document Writer how to retrieve, calculate and present Values for our Writer Report. This User Manual is the Reference for the Tag Logic syntax.

Hint: We are not expected to necessarily understand or code with the syntax described below. Instead, we may use Tag Templates (they generate the necessary commands automatically) or the Wizard Command (w) to activate the Tag Wizard on a Command Line. The Wizard will write the necessary syntax for us.

HOW TO UNDERSTAND TAG LOGIC

The purpose of Tag Commands is simple: to process an END VALUE !

However, other purposes may be achieved as well, as we will see.

Document Writer execution is based on the concept of an HTML Template that provides a layout, and may include any pre-defined graphics, text and values, any of which may already be pre-set in terms of font, justification, width, etc.

If a Document is executed in the format just described, then it will produce exactly the same results, i.e. content, each time it is executed. However, the purpose of a Report is to produce Results that may be different each time. For instance, if the Document is an Invoice, then it will print a different Invoice each time it is executed, with different content as well. That is where the Tags come into the picture.

Tags are placed in the Template, and are then specified to contain logic that is executed when the Document is executed, thereby producing the desired results, and which may be different for each execution. Instead of printing the Tag, the system will print the Value assigned to each Tag as processed by the Tag Logic.

COMMANDS: - The following commands are available with Tag Logic.

ASSIGN (to Assign Values to Variables based on Operations)

BIND (to concatenate 2 Values)

CALL (to Call a Program)

DIVERT (to provide a point whereto the logic may branch)

FUNCTION (to perform a Function and assign the result)

GETSTRING (to perform a MIDstring function)

ICONV (for Input Conversion Operations)

JUMP (to branch to a 'divert' point)

MAP (to MAP Financial Period Ranges inIES Business)

NOTE (to insert a Note that helps one to understand the logic)

OCONV (for Output Display Conversions)

PLACE (to position and justify a Value on a specified Width)

RESULT (to assign a variable to the current Tag)

SELECT, LOOP, ENDLOOP (to process 1 or more Data Keys, including Commands for each Data Key)

TOTAL (to preserve a raw Number for Totaling)

UNIT (to perform an IES HTML Unit)

W (reserved for entering Wizard)

X (reserved for entering Long Text Editing)

The Commands may include processing of Variables, and we discern the following Variable Types: -

RUNTIME (assign FROM only)

(specify like: R(Fn,Mn) where 'Fn' should be the number of the Field in the Runtime Record, and'Mn' indicates the MultiValue Number to use. If'Mn' is omitted, then'1' is Implied, e.g. R(2) means 1st MultiValue on Field 2 from

the Runtime record.

TAG (assign FROM and TO, assign TO only for current Tag)

(specify like: ME (current Tag Value)

or like: ME(-1,0) for the previous Tag main value

or like: ME(0,1) to address an additional Value on this Tag

[Tags may have up to 7 additional Values other than the

main Value. To address any of those, or any other Tag's

Values, we must use the format ME(n1,n2) where 'n1' identifies the Tag and where '0' is current Tag, -10 is 10 Tags below current Tag Number, +5 is this Tag + 5. And where 'n2' specifies which Value of the Tag being addressed, and where '0' is always the main Value, and there may be

any other value between 1 and 7.

Another way of addressing a Tag Value, ONLY for Assignment FROM (not TO), is as follows: -

T(0001,0)

In the above example, the Tag Number must be a 4 digit Number, and must be followed by the indicator addressing the Tag Value, and where '0' is the main value, and where an indicator from 1 to 7 addresses additional Tag Values ('1' is always the Value preserved by the 'T ME' (i.e. Total format) Command.

SOURCE (assign FROM only)

(specify like: S(#n,Fn,Mn) where '#n' is a Number pointing

to the nth DataSource specified for this Document, 'Fn' points to the Field Number, and 'Mn' indicates the MultiValue

and of course may be omitted to imply 1st Mv)

Note: When a Unit is being executed, then another format is allowed for 'Mn', and it may be specified as 'ITR'. ITR means the current Iteration number for the Unit.

WORK (assign FROM and TO)

(specify like: W(n) where ‘n’ is a number > 0 and < 1000, i.e. which WORK variable)

LITERAL (assign FROM only)

(specify like: L(text) where ‘text’ is the Literal Value)

COMMON (assign FROM only)

(specify like: C(D) for Date, C(T) for Time, C(U) for current UserCode)

NAME (assign FROM only)

(specify like: N(name) where name is selected as a Valid Execution Name (Dict Object) for the Current Key being processed on the CurrentFileName)

KEY (specify like: K where ‘K’ is the current Record Key being

Processed during the Loop; this Variable is only valid inside a Loop construct;)

Of course, the Commands allow us to perform OPERATIONS on the Variables, and we discern the following OPERATIONS: -

+ (plus / add)

- (minus / subtract)

* (multiplication)

/ (division)

Rounding

% (percentage)

Int (Integer)

>, =, <, #, >=, <= (these are ‘test’ operations)

Tag Logic is based on how we string together the elements listed above, and there are rules as to how we may do so. Therefore, let us have a look at the examples of how we may use the various Commands, Variables and Operations.

COMMAND: ASSIGN

The ASSIGN Command is used to assign the result of an Operation to a specified Variable.

A ME = VAR + VAR (ADD Operation)

A W(7) = VAR – VAR (MINUS)

A W(3) = VAR * VAR (MULTIPLICATION)

A ME(0,5) = VAR / VAR (DIVISION)

A ME = ME R 0 (if ME = 1.5 it becomes 2, if 1.4 it becomes 1)

A ME = ME R 1 (if ME = 1.55 it becomes 1.6, if 1.54 it becomes 1.5)

A ME = ME R 2 (rounds to 2 decimals)

A ME = W(5) R 3 (rounds to 3 decimals)

A W(1) = W(2) % W(3) (will assign to W(1) what w(2) is as % of W(3))

A ME = ME I (if ME = 1.5 then ME becomes = 1, i.e. performs Integer Function,

rather than rounding)

The ASSIGN Command can also be used to create test conditions for the JUMP Command, as follows: -

A W(10) = VAR # VAR

The Command is specified like a normal assignment statement, except that a test operation is used instead of a normal operation. The test operations are >, >=, <, <=, = , #. In the example above, W(10) is assigned '1' (i.e. TRUE) if the 1 st VAR is not equal to the 2 nd VAR, and '0' (i.e. FALSE) if it is. W(10) is then ready for use with a JUMP Command (the JUMP will execute if the condition tests FALSE).

COMMAND: BIND

The BIND Command is used to concatenate 2 Values, like –

B VAR1 = VAR2 VAR3

B (Command)

VAR1 (Final Variable, which must be a Variable that may be assigned)

= (Equal Symbol)

VAR2 (Value which will be 1 st part of Concatenation)

VAR3 (Value which will be 2 nd part of Concatenation)

COMMAND: CALL

This Command is used to perform a 'ByValue' Subroutine Call, for example: -

C name_of_program W(1) W(2)

Where the program will be called with an Argument List of 2 values, and when control returns, the current Values of those Variables are re-assigned to them, if changed. Therefore, only WORK type Variables may be used.

A minimum of 1 and a maximum of 5 Argument Variables may be used for a CALL.

COMMAND: DIVERT

The DIVERT Command is used to state a LABEL, which is the 1 st string following the DIVERT. A LABEL may be used by a JUMP Command to branch execution of the logic to the LABEL, based on the testing of a variable which either is equal to '1' or not.

D here

Caution (1): A Divert LABEL may not be inside a Loop Construct. During execution, if a Command attempts to branch to a Label that is inside a Loop Construct, then the Command Execution will terminate.

Caution (2): During execution, if a Command attempts to branch to a Label which is not found, then the Command Execution will terminate.

COMMAND: FUNCTION

The FUNCTION Command includes a number of variations.

#1: Simple Assign

The format for a Simple Assign is: F VAR = VAR

The VAR on the left of the '=' symbol must be a variable that may be assigned to.

#2: Make Raw Number

'Make Raw' turns a number that is already formatted by a 'mr' type Display Conversion back into a raw number, for example '1,200.00' is converted to '120000'.

The format for 'Make Raw' is: F VAR R

VAR is a variable that may be assigned to, and the 'raw' function is performed on itself.

#3: Count

The Count Function is used to (a) count the number of Fields in a Record, (b) the number of Values in a Field, or (c) the number of sub values in a Value. The Function is always executed on one of the listed Data Sources or the Runtime Record for this Document. The following Formats are supported: -

F C n F

F C n V n

F C n S n n

#4: Bold (This function is used with Print Writer only, not with Document Writer.)

F (B)

This will apply BOLD printing to the tag result.

#5: Underline (This function is used with Print Writer only, not with Document Writer.)

F (U)

This will apply an UNDERLINE to the tag result. If the Tag is iterative, i.e. as in a Print Unit, then the UNDERLINE will only be applied to the last value in the list.

COMMAND: GETSTRING

The GETSTRING Command is used to 'cut' a piece out of a String Value. For example: -

G ME = L(1) L(10)

Where the Tag Value in this case (ME) is re-assigned to itself after taking whatever is at Character Position '1', for a length of '10' characters.

The format is –

G (Command)

VAR (Variable that may be assigned)

= (Equal Symbol)

Literal (must be a number > 0, indicates Character Start Position)

Literal (must be a number > 0, indicates Number of Characters to use)

COMMAND: ICONV

The ICONV Command functions the same as the OCONV Command, but performs an ICONV function.

I ME = ME L(d/)

This command assigns the current Tag Value to itself after performing an Input Date Conversion (i.e. to determine the internal Value of a Date) on itself.

COMMAND: JUMP

J VAR LABEL

The JUMP Command causes execution of the logic to branch to the LABEL specified, IF VAR # 1. VAR would have been previously 'tested' with an ASSIGN Command that includes a TEST operation, which would set VAR to either '1' or '0'.

LABEL means a label set with a prior DIVERT Command.

COMMAND: MAP

The MAP Command is specifically designed for use with IES Business Data Sources, which support dynamic Financial Period Mapping. For Example, if in a particular IES Business DataMart the current Year is set to be from Period 13 to 24, and the Current Period = 20, then this means Start of Year = Period 13, End of Year = Period 24. If a NAME Variable is now used on a Key from the Ledger, i.e. to get the Year-To-Date Balance of an Account, then this Value will be based on the Current year Settings. However, the MAP command allows us to specify to the system to interpret the YTD Value according to our Requirements (which may in fact be a Runtime Choice for the User, and specified on the Runtime Record). Therefore, if this Report also contains a RunTime (PreRun) Record where the User specifies his/her requirement of how the Financial Period should be interpreted, then we use the MAP command to indicate this before executing the ASSIGN Command that will perform retrieval of the NAME value. For example:-

M R(1) R(2) R(3)

A ME = L:YTD

The MAP Command must always pass 3 Values: Start of Year, End of Year, Current Period. In the example above, this was specified by the User on the PreRun Runtime Record, from where we access Fields 1, 2 and 3 respectively.

After performing the MAP Command, we assign to the current Tag the YTD Value of the Account Key we are addressing in the Tag Logic, and the Value is now an interpretation of the YTD Value according to User requirements, rather than what the system settings state for current Financial Year. (Very powerful stuff.) Please note that 'L:YTD' is simply an example of a NAME Variable, but the Names that are available for selection with this Command may be different.

A MAP Command is valid until superseded by another MAP Command, or until the end of Logic Execution for the current Tag.

COMMAND: NOTE

The purpose of the NOTE Command is to provide a comment, perhaps to explain a point about the logic for the Tag.

N text

If a Command starts with 'N', then it is ignored for processing purposes.

COMMAND: OCONV

The OCONV Command may appear anywhere in Tag Logic, and includes an Assignment construct, as follows –

OCONV VAR = VAR VAR

The 1st VAR, i.e. on the left of the '=' sign, must be a Variable Type that a value may be assigned to, e.g. a TAG or WORK variable.

The next VAR, i.e. the 1 st following '=' is the Value to use for the OCONV, while the last VAR is the OCONV OPERATION to perform, and is ALWAYS a Literal.

Here is an example:

O W(1) = C(D) L(d4/)

Have another look at the Commands and the Variable types listed previously, and note that each Command and Variable type has a unique 1 st Character, i.e. there is only 1 Command that starts with 'O', and therefore, although you may type 'OCONV', it is also permissible to just use the 1 st letter of the Command, and the same with the Variables. Therefore, in the example above, the 'O' represents the 'OCONV' Command. Next, we have 'W' that indicates a WORK Variable # 1, and we are assigning to this Variable the result of the Operation that is to follow. The Operation is based on the 'Date', which comes form the COMMON Variables, and the OPERATION to perform is specified as a Literal Value, i.e. "d4/" which would turn a Date into a format like "12/12/2004".

If we should specify "O ME = ME L(d/4)" then we are assigning the main Value of the current Tag to itself, after performing a 'd4/' Date Conversion on it.

COMMAND: PLACE

The PLACE Command is used to position an existing Value on a pre-defined, pre-justified space. For example, to place a Value as Left Justified in a String Space that is 20 Characters wide, irrespective of the current length of the Value to Place, the Command may take a format like this –

P W(3) = L(L) L(20)

The format is discerned as follows: -

P (Command)

W(3) (The Variable being assigned)

= (Equal Symbol)

L(L) (The Justification, must be a Literal which is either 'L' or 'R')

L(20) (The width or length of the final Value in number of characters, must be a

Literal)

COMMAND: RESULT

The RESULT Command is used for simple assignment of a Value to the current Tag, like –

R VAR

Note: While the RESULT Command is commonly used to assign the end Value of Tag Logic to the Tag, it should

not be included in the Logic of a Tag that calls a Unit. UNIT logic assigns the UNIT result to the Tag automatically.

COMMAND: SELECT, LOOP, ENDLOOP

These 3 Commands are always used together in a set. The SELECT retrieves Record Keys (or a range of numbers) to work with, LOOP starts execution of the Keys 1 by 1, and may be followed by any number of commands before the ENDLOOP is encountered, and which terminates execution of Commands per Key.

The SELECT Command takes the following formats:-

S O ObjectName (executes a Data Selector = 'ObjectName' and returns the Keys)

S R L(n) (RANGE n is a number between 1 and 100, e.g. if '3' then the result is '1 2 3', and if it is '7', then the result is '1 2 3 4 5 6 7')

S K abc 123 ggh ('K' signals direct specification of Keys, which may not contain spaces, i.e. the Keys are themselves separated by spaces)

S S Select Ledgermast with descr = "a]" (meaning 'Select Statement' and all Text following 'S S' is executed as a direct Statement to retrieve the Keys)

Extended functionality on 'S S': The Direct Select Statement may extract a Value from the Runtime Record, by using the convention "??". For example, if you use a Select Statement where a Value needs to be retrieved from the Runtime Record, this may be encapsulated within "??", as follows: -

Select Ledgermast with Deptpos = "?R(3)?"

Whatever is specified inside "??" will be evaluated as a Runtime Retrieval, and if the specification is valid, then the data will be retrieved and replaced ex the Runtime record. In the example above, the system will retrieve field 3 from the runtime record, and if this value is "101" then the Select statement that will be executed will look like this: -

Select Ledgermast with Deptpos = "101"

If the Retrieved value is "[]", i.e. wildcards to select any Department, then the Select statement to be executed will be: -

Select Ledgermast with Deptpos = "[]"

Here is an example of how to use the 3 Commands in a set: -

```
S O ledgermast
L
  Command
  Command
  Command
E
```

The 1st command signals a Data Selector with the name 'ledgermast' to be executed. The 'L' command starts the loop, whereafter, for each Key retrieved by the Data Selector, 3 Commands are executed, and then the 'E' signals ENDLOOP for each Key being processed.

COMMAND: TOTAL

The TOTAL Command is used to preserve a Value in raw number format, which can be used as input to another Tag Value that represents a Total of various Tag Values. In the previous example where we retrieved and assigned a Year-to-Date Value, we may suggest that the Value could have been something like "200.25", and probably we would have followed that with an OCONV command to present the final Tag Value as "$200.25". However, this final value is no longer a Number, i.e. in Number format, and if the logic of another Tag is specified to reference many other Tags to produce a Total, then it cannot use such a Value. Therefore, if a Tag Value is going to be used by another Tag in a Total, then it is necessary to perform the TOTAL Command on the Value before it is modified for Display purposes. For example: -

M R(1) R(2) R(3)

A ME = L:YTD

T ME

O ME = ME L(mr22,)

In this example, we MAP the Financial Year, then retrieve the YTD Value, then preserve the result with the TOTAL Command, and finally perform an OCONV Command to display the Value in the desired format.

The TOTAL Command is specified only as 'T ME', and no other format is allowed.

Note (1) :

A Tag will only return a Value in the Report if a Command like 'A ME =' or 'R VAR' is specified, i.e. the Tag is actually assigned a Value during the Logic Execution.

Therefore, the use of 'hidden' Tags is allowed – simply specify any number of Tags that are executed to derive Values that will be referenced by other Tags, but the hidden Tags never assign Values to themselves (i.e. to the MAIN Value of the Tag, even if assigned to secondary Values on the Tag), and therefore these Tags do not print Values.

Note (2) :

The TOTAL Command always assigns the specified Value to the 1st additional Tag Value, i.e. 1 of 7 additional Values as allowed. For Tags that are Totals of other Tag Values, please always reference Value 1 of the Tags to total, and always include a 'T ME' for all the Tags that form part of Totals.

COMMAND: UNIT

The UNIT Command is used to execute a HTML Unit, but Units can only be called from the Unit Tag Templates. Please see the Tag Template User Manuals.

COMMAND: W

Not a Command. Reserved for entering the **W**izard during Command specification.

COMMAND: X

Not a Command. Reserved for entering Long Te**X**t Editing during Command specification.