

Community  
Follow

Ask a Question



Login

RSS Feed

## Technical Articles



Jeffrey Towell

October 25, 2015 | 14 minute read

# ABAP 7.40 Quick Reference

65 198 605,340

So you're an experienced ABAP programmer wanting to leverage off the fantastic new functionality available to you in ABAP 7.40!

However, searching for information on this topic leads you to fragmented pages or blogs that refer to only a couple of the new features available to you.

What you need is a quick reference guide which gives you the essentials you need and shows you how the code you are familiar with can be improved with ABAP 7.40.

The below document contains exactly this!

Follow



Like



RSS Feed

It gives examples of “classic” ABAP and its 740 equivalent. It goes into more details on the more difficult topics normally via examples. This allows the reader to dive in to the level they desire. While this document does not contain everything pertaining to ABAP 740 it certainly covers the most useful parts in the experience of the author.

The document has been compiled by drawing on existing material available online as well as trial and error by the author. In particular the blogs by Horst Keller have been useful and are the best reference I have found (prior to this document ). He has a landing page of sorts for his various blogs on the topic here:

[ABAP Language News for Release 7.40](#)

Credit also goes to Naimesh Patel for his useful explanations and examples on ABAP 7.40. Here is his example of the “FOR iteration expression” which I leaned on (links to his other 740 articles can be found at the bottom of the link):

<http://zevolving.com/2015/05/abap-740-for-iteration-expression/>

I compiled the below document to make the transition to using ABAP 740 easier for myself and my project team. It has worked well for us and I hope it will do the same for you.

Regards,

Jeff Towell

## ABAP 7.40 Quick Reference

Author:	Jeffrey Towell
Created:	2015

## Contents

Follow

 Like

 RSS Feed

. Inline Declarations

2. Table Expressions

3. Conversion Operator CONV

I. Definition

II. Example

4. Value Operator VALUE

I. Definition

II. Example for structures

III. Examples for internal tables

5. FOR operator

I. Definition

II. Explanation

III. Example 1

IV. Example 2

V. FOR with THEN and UNTIL|WHILE

## 6. Reduction operator REDUCE

Follow

Definition



Like

. Note

III. Example 1



RSS Feed

IV. Example 2

V. Example 3

## 7. Conditional operators COND and SWITCH

I. Definition

II. Example for COND

III. Example for SWITCH

## 8. CORRESPONDING operator

I. Definition

II. Example Code

III. Output

IV. Explanation

V. Additions MAPPING and EXCEPT

## 9. Strings

## I. String Templates

Follow

### . Concatenation

 Like

### I. Width/Alignment/Padding.

## IV. Case

 RSS Feed

### V. ALPHA conversion

## VI. Date conversion

## 10. Loop at Group By

### I. Definition

### II. Explanation

### III. Example

### IV. Output

## 11. Classes/Methods

### I. Referencing fields within returned structures

### II. Methods that return a type BOOLEAN

### III. NEW operator

## 12. Meshes

### I. Problem

### II. Solution

### III. Output

## 13. Filter

### I. Definition

### II. Problem

### III. Solution

# 1. Inline Declarations

Follow

 Like

 RSS Feed

Description	Before 7.40	With 7.40
Data statement	<pre>DATA text TYPE string. text = 'ABC'.</pre>	<pre>DATA(text) = 'ABC'.</pre>
Loop at into work area	<pre>DATA wa like LINE OF itab. LOOP AT itab INTO wa. ... ENDLOOP.</pre>	<pre>LOOP AT itab INTO DATA(wa) . ... ENDLOOP.</pre>
Call method	<pre>DATA a1 TYPE ... DATA a2 TYPE ... oref-&gt;meth( IMPORTING p1 = a1             IMPORTING p2 = a2             ).</pre>	<pre>oref-&gt;meth(             IMPORTING p1 = DATA(a1)             IMPORTING p2 = DATA(a2) ).</pre>
Loop at assigning	<pre>FIELD-SYMBOLS: &lt;line&gt; type ... LOOP AT itab ASSIGNING &lt;line&gt;. ... ENDLOOP.</pre>	<pre>LOOP AT itab             ASSIGNING FIELD-SYMBOL(&lt;line&gt;) . ... ENDLOOP.</pre>
Read assigning	<pre>FIELD-SYMBOLS: &lt;line&gt; type ... READ TABLE itab             ASSIGNING &lt;line&gt;.</pre>	<pre>READ TABLE itab             ASSIGNING FIELD-SYMBOL(&lt;line&gt;) .</pre>
Select into table	<pre>DATA itab TYPE TABLE OF dbtab. SELECT * FROM dbtab         INTO TABLE itab         WHERE fld1 = lv_fld1.</pre>	<pre>SELECT * FROM dbtab         INTO TABLE @DATA(itab)         WHERE fld1 = @lv_fld1.</pre>
Select single into	<pre>SELECT SINGLE f1 f2         FROM dbtab         INTO (lv_f1, lv_f2)</pre>	<pre>SELECT SINGLE f1 AS my_f1,               f2 AS abc         FROM dbtab</pre>

Follow



Like



RSS Feed

```
WHERE ...  
WRITE: / lv_f1, lv_f2.
```

```
        INTO DATA(ls_structure)  
        WHERE ...  
WRITE: / ls_structure-  
my_f1,          ls_structure-  
abc.
```

## 2. Table Expressions

If a table line is not found, the exception `CX_SY_ITAB_LINE_NOT_FOUND` is raised. No `sy-subrc`.

Descriptio n	Before 7.40	With 7.40
Read Table index	<pre>READ TABLE itab INDEX idx  INTO wa.</pre>	<pre>wa = itab[ idx ].</pre>
Read Table using key	<pre>READ TABLE itab INDEX idx  USING KEY key  INTO wa.</pre>	<pre>wa = itab[ KEY key INDEX idx ].</pre>
Read Table with key	<pre>READ TABLE itab     WITH KEY col1 = ...            col2 = ...     INTO wa.</pre>	<pre>wa = itab[ col1 = ...col2 = ... ].</pre>
Read Table with	<pre>READ TABLE itab     WITH TABLE KEY key     COMPONENTS col1 = ...</pre>	<pre>wa = itab[ KEY key col1 = ...            col2 = ... ].</pre>

Follow



Like



RSS Feed

key compone nts	col2 = ...  INTO wa.	
Does record exist?	<pre>READ TABLE itab ...  TRANSPORTING NO FIELDS.  IF sy-subrc = 0.  ...  ENDIF.</pre>	<pre>IF line_exists( itab[ ... ] ).  ...  ENDIF.</pre>
Get table index	<pre>DATA idx type sy-tabix.  READ TABLE ...  TRANSPORTING NO FIELDS.  idx = sy-tabix.</pre>	<pre>DATA(idx) =     line_index( itab[ ... ] ).</pre>

**NB:** There will be a short dump if you use an inline expression that references a non-existent record.

SAP says you should therefore assign a field symbol and check sy-subrc.

```
ASSIGN lt_tab[ 1 ] to FIELD-SYMBOL(<ls_tab>).  
IF sy-subrc = 0.  
...  
ENDIF.
```



NB: Use `itab [ table_line = ... ]` for untyped tables.

Follow

 Like



RSS Feed

## 3. Conversion Operator CONV

### I. Definition

`CONV dtype | # ( ... )`

**dtype** = Type you want to convert to (explicit)

**#** = compiler must use the context to decide the type to convert to (implicit)

### II. Example

Method `cl_abap_codepage=>convert_to` expects a string

Before 7.40
<pre>DATA text    TYPE c LENGTH 255. DATA helper  TYPE string. DATA xstr    TYPE xstring.  helper = text. xstr = cl_abap_codepage=&gt;convert_to( source = helper ).</pre>
With 7.40
<pre>DATA text TYPE c LENGTH 255. DATA(xstr) = cl_abap_codepage=&gt;convert_to( source = CONV string( text ) ).  OR  DATA(xstr) = cl_abap_codepage=&gt;convert_to( source = CONV #( text ) ).</pre>

Follow

# 1. Value Operator VALUE

 Like

## Definition

**Variables:** `VALUE dtype|#( )`

 RSS Feed

**Structures:** `VALUE dtype|#( comp1 = a1 comp2 = a2 ... )`

**Tables:** `VALUE dtype|#( ( ... ) ( ... ) ... ) ...`

## II. Example for structures

```
TYPES: BEGIN OF ty_columns1, "Simple structure
cols1 TYPE i,
cols2 TYPE i,
END OF ty_columns1.
```

```
TYPES: BEGIN OF ty_columnns2, "Nested structure
coln1 TYPE i,
coln2 TYPE ty_columns1,
END OF ty_columns2.
```

```
DATA: struc_simple TYPE ty_columns1,
      struc_nest TYPE ty_columns2.
```

```
struct_nest = VALUE t_struct(coln1 = 1
                             coln2-cols1 = 1
                             coln2-cols2 = 2 ).
```

OR

Follow



Like

```

struct_nest = VALUE t_struct(coln1 = 1
coln2 = VALUE #( cols1 = 1
cols2 = 2 ) ).

```



RSS Feed

### III. Examples for internal tables

Elementary line type:

```

TYPES t_itab TYPE TABLE OF i WITH EMPTY KEY.
DATA itab TYPE t_itab.

```

```

itab = VALUE #( ( ) ( 1 ) ( 2 ) ).

```

Structured line type (RANGES table):

```

DATA itab TYPE RANGE OF i.
itab = VALUE #( sign = 'I' option = 'BT' ( low = 1 high = 10 )
( low = 21 high = 30 )
( low = 41 high = 50 )
option = 'GE' ( low = 61 ) ).

```

## 5. FOR operator

### I. Definition

```

FOR wa|<fs> IN itab [INDEX INTO idx] [cond]

```

## II. Explanation

Follow

This effectively causes a loop at itab. For each loop the row read is assigned to a work area (wa) or field-symbol(<fs>).



Like



RSS Feed

This wa or <fs> is local to the expression i.e. if declared in a subrourine the variable wa or <fs> is a local variable of

that subroutine. Index like SY-TABIX in loop.

Given:

```
TYPES: BEGIN OF ty_ship,
tknum TYPE tknum,      "Shipment Number
name  TYPE ernam,      "Name of Person who Created the Object
city  TYPE ort01,       "Starting city
route TYPE route,      "Shipment route
END OF ty_ship.

TYPES: ty_ships TYPE SORTED TABLE OF ty_ship WITH UNIQUE KEY tknum.
TYPES: ty_citys TYPE STANDARD TABLE OF ort01 WITH EMPTY KEY.
```

GT\_SHIPS `type` ty\_ships. -> has been populated as follows:

Row	TKNUM[C(10)]	Name[C(12)]	City[C(25)]	Route[C(6)]
1	001	John	Melbourne	R0001
2	002	Gavin	Sydney	R0003
3	003	Lucy	Adelaide	R0001
4	004	Elaine	Perth	R0003

## III. Example 1

Populate internal table GT\_CITYS with the cities from GT\_SHIPS.

Follow

 Like RSS Feed**Before 7.40**

```

DATA: gt_citys TYPE ty_citys,
      gs_ship  TYPE ty_ship,
      gs_city  TYPE ort01.

LOOP AT gt_ships INTO gs_ship.
  gs_city = gs_ship-city.
  APPEND gs_city TO gt_citys.
ENDLOOP.

```

**With 7.40**

```
DATA(gt_citys) = VALUE ty_citys( FOR ls_ship IN gt_ships ( ls_ship-city ) ).
```

## IV. Example 2

Populate internal table GT\_CITYS with the cities from GT\_SHIPS where the route is R0001.

**Before 7.40**

```

DATA: gt_citys TYPE ty_citys,
      gs_ship  TYPE ty_ship,
      gs_city  TYPE ort01.

LOOP AT gt_ships INTO gs_ship WHERE route = 'R0001'.
  gs_city = gs_ship-city.
  APPEND gs_city TO gt_citys.
ENDLOOP.

```

**With 7.40**

```
DATA(gt_citys) = VALUE ty_citys( FOR ls_ship IN gt_ships
                                WHERE ( route = 'R0001' ) ( ls_ship-city ) ).
```

Note: `ls_ship` does not appear to have been declared but it is declared implicitly.

Follow

 Like



RSS Feed

## FOR with THEN and UNTIL|WHILE

```
FOR i = ... [THEN expr] UNTIL|WHILE log_exp
```

Populate an internal table as follows:

TYPES:

```
BEGIN OF ty_line,
```

```
  col1 TYPE i,
```

```
  col2 TYPE i,
```

```
  col3 TYPE i,
```

```
END OF ty_line,
```

```
ty_tab TYPE STANDARD TABLE OF ty_line WITH EMPTY KEY.
```

**Before 7.40**

Follow

 Like RSS Feed**Before 7.40**

```

DATA: gt_itab TYPE ty_tab,
      j       TYPE i.
FIELD-SYMBOLS <ls_tab> TYPE ty_line.

j = 1.
DO.
  j = j + 10.
  IF j > 40. EXIT. ENDIF.
  APPEND INITIAL LINE TO gt_itab ASSIGNING <ls_tab>.
  <ls_tab>-col1 = j.
  <ls_tab>-col2 = j + 1.
  <ls_tab>-col3 = j + 2.
ENDDO.

```

**With 7.40**

```

DATA(gt_itab) = VALUE ty_tab( FOR j = 11 THEN j + 10 UNTIL j > 40
                              ( col1 = j col2 = j + 1 col3 = j + 2 ) ).

```

## 6. Reduction operator REDUCE

### I. Definition

```

... REDUCE type(
INIT result = start_value
...
FOR for_exp1
FOR for_exp2
...
NEXT ...

```

```
result = iterated_value
```

```
. )
```

Follow

I. Note

While VALUE and NEW expressions can include FOR expressions, REDUCE must include at least one FOR expression. You can use all kinds of FOR expressions in REDUCE:



RSS Feed

- with IN for iterating internal tables
- with UNTIL or WHILE for conditional iterations

III. Example 1

Count lines of table that meet a condition (field F1 contains “XYZ”).

Before 7.40
<pre>DATA: lv_lines TYPE i. LOOP AT gt_itab INTO ls_itab where F1 = 'XYZ'. lv_lines = lv_lines + 1. ENDLOOP.</pre>
With 7.40
<pre>DATA(lv_lines) = REDUCE i( INIT x = 0 FOR wa IN gt_itab WHERE( F1 = 'XYZ' ) NEXT x = x + 1 ).</pre>

IV. Example 2

Sum the values 1 to 10 stored in the column of a table defined as follows

```
DATA gt_itab TYPE STANDARD TABLE OF i WITH EMPTY KEY.  
gt_itab = VALUE #( FOR j = 1 WHILE j <= 10 ( j ) ).
```

Before 7.40
-------------



Follow

 Like RSS Feed**Before 7.40**

```
DATA: lv_line TYPE i,
      lv_sum  TYPE i.
LOOP AT gt_itab INTO lv_line.
  lv_sum = lv_sum + lv_line.
ENDLOOP.
```

**With 7.40**

```
DATA(lv_sum) = REDUCE i( INIT x = 0 FOR wa IN itab NEXT x = x + wa ).
```

## V. Example 3

Using a class reference – works because “write” method returns reference to instance object

**With 7.40**

```
TYPES outref TYPE REF TO if_demo_output.

DATA(output) = REDUCE outref( INIT out = cl_demo_output=>new( )
text = `Count up:`
                                FOR n = 1 UNTIL n > 11
                                NEXT out = out->write( text )
text = |{ n }| ).

output->display( ).
```

## 7. Conditional operators COND and SWITCH

### I. Definition

```
... COND dtype|#( WHEN log_exp1 THEN result1
[ WHEN log_exp2 THEN result2 ]
```

Follow

 Like RSS Feed

```

...
ELSE resultn ] ) ...
SWITCH dtype|#( operand
WHEN const1 THEN result1
WHEN const2 THEN result2 ]
...

```

## I. Example for COND

```

DATA(time) =
COND string(
  WHEN sy-timlo < '120000' THEN
    |{ sy-timlo TIME = ISO } AM|
  WHEN sy-timlo > '120000' THEN
    |{ CONV t( sy-timlo - 12 * 3600 )
TIME = ISO } PM|
  WHEN sy-timlo = '120000' THEN
    |High Noon|
  ELSE
    THROW cx_cant_be( ) ).

```

## III. Example for SWITCH

```

DATA(text) =
NEW class( )->meth(
  SWITCH #( sy-langu
    WHEN 'D' THEN `DE`
    WHEN 'E' THEN `EN`
    ELSE THROW cx_langu_not_supported( ) ) ).

```

## 8. Corresponding Operator

Follow

. Definition

Like

CORRESPONDING type( [BASE ( base )] struct|itab [mapping|except] )

1. Example Code

RSS Feed

With 7.40

```

TYPES: BEGIN OF line1, col1 TYPE i, col2 TYPE i, END OF line1.
TYPES: BEGIN OF line2, col1 TYPE i, col2 TYPE i, col3 TYPE i, END OF line2.
DATA(ls_line1) = VALUE line1( col1 = 1 col2 = 2 ).
WRITE: / 'ls_line1 =' ,15 ls_line1-col1, ls_line1-col2.
DATA(ls_line2) = VALUE line2( col1 = 4 col2 = 5 col3 = 6 ).
WRITE: / 'ls_line2 =' ,15 ls_line2-col1, ls_line2-col2, ls_line2-col3.
SKIP 2.

ls_line2 = CORRESPONDING #( ls_line1 ).
WRITE: / 'ls_line2 = CORRESPONDING #( ls_line1 )'

        ,70 'Result is ls_line2 = '
        ,ls_line2-col1, ls_line2-col2, ls_line2-col3.
SKIP.

ls_line2 = VALUE line2( col1 = 4 col2 = 5 col3 = 6 ). "Restore ls_line2
ls_line2 = CORRESPONDING #( BASE ( ls_line2 ) ls_line1 ).
WRITE: / 'ls_line2 = CORRESPONDING #( BASE ( ls_line2 ) ls_line1 )'

        , 70 'Result is ls_line2 = ', ls_line2-col1

        , ls_line2-col2, ls_line2-col3.
SKIP.

```

Follow



Like



RSS Feed

With 7.40

```
s_line2 = VALUE line2( col1 = 4 col2 = 5 col3 = 6 ).    "Restore ls_line2
DATA(ls_line3) = CORRESPONDING line2( BASE ( ls_line2 ) ls_line1 ).
WRITE: / 'DATA(ls_line3) = CORRESPONDING line2( BASE ( ls_line2 ) ls_line1
)',
      , 70 'Result is ls_line3 = ' , ls_line3-col1
      , ls_line3-col2, ls_line3-col3.
```

III. Output

```
ls_line1 =          1          2
ls_line2 =          4          5          6

ls_line2 = CORRESPONDING #( ls_line1 )           Result is ls_line2 =          1          2          0
ls_line2 = CORRESPONDING #( BASE ( ls_line2 ) ls_line1 )           Result is ls_line2 =          1          2          6
DATA(ls_line3) = CORRESPONDING line2( BASE ( ls_line2 ) ls_line1 )           Result is ls_line3 =          1          2          6
```

IV. Explanation

Given structures ls\_line1 & ls\_line2 defined and populated as above.

	Before 7.40	With 7.40
1	<pre>CLEAR ls_line2.  MOVE-CORRESPONDING ls_line1  TO ls_line2.</pre>	<pre>ls_line2 = CORRESPONDING #( ls_line1 ). </pre>
2	<pre>MOVE-CORRESPONDING ls_line1  TO ls_line2.</pre>	<pre>ls_line2 = CORRESPONDING # ( BASE ( ls_line2 ) ls_line1 ).</pre>

Follow

 Like RSS Feed

	Before 7.40	With 7.40
3	<pre>DATA: ls_line3 like ls_line2. ls_line3 = ls_line2. MOVE-CORRESPONDING ls_line1       TO ls_line2.</pre>	<pre>DATA(ls_line3) = CORRESPONDING line2       ( BASE ( ls_line2 ) ls_line1       ).</pre>

1. The contents of ls\_line1 are moved to ls\_line2 where there is a matching column name. Where there is no match the column of ls\_line2 **is initialised**.

2. This uses the existing contents of ls\_line2 as a base and overwrites the matching columns from ls\_line1.

**This is exactly like MOVE-CORRESPONDING.**

3. This creates a third and new structure (ls\_line3) which is based on ls\_line2 but overwritten by matching columns of ls\_line1.

## V. Additions MAPPING and EXCEPT

MAPPING allows you to map fields with non-identically named components to qualify for the data transfer.

```
... MAPPING  t1 = s1 t2 = s2
```

EXCEPT allows you to list fields that must be excluded from the data transfer

```
... EXCEPT {t1 t2 ...}
```

## 9. Strings

## I. String Templates

Follow



Like



RSS Feed

A string template is enclosed by two characters “|” and creates a character string.

Literal text consists of all characters that are not in braces {}. The braces can contain:

- data objects,
- calculation expressions,
- constructor expressions,
- table expressions,
- predefined functions, or
- functional methods and method chainings

Before 7.40
<pre>DATA itab TYPE TABLE OF scarr.  SELECT * FROM scarr INTO TABLE itab.  DATA wa LIKE LINE OF itab.  READ TABLE itab WITH KEY carrid = 'LH' INTO wa.  DATA output TYPE string.  CONCATENATE 'Carrier:' wa-carrname INTO output SEPARATED BY space.  cl_demo_output=&gt;display( output ).</pre>
With 7.40
<pre>SELECT * FROM scarr INTO TABLE @DATA(lt_scarr).  cl_demo_output=&gt;display(  Carrier: { lt_scarr[ carrid = 'LH' ]-carrname }  ).</pre>

## II. Concatenation

Follow

Like

RSS Feed

Before 7.40
<pre>DATA lv_output TYPE string. CONCATENATE 'Hello' 'world' INTO lv_output SEPARATED BY space.</pre>
With 7.40
<pre>DATA(lv_out) =  Hello  &amp;     &amp;  world .</pre>

## III. Width/Alignment/Padding

```
WRITE / |{ 'Left'      WIDTH = 20 ALIGN = LEFT   PAD = '0' }|.
WRITE / |{ 'Centre'   WIDTH = 20 ALIGN = CENTER PAD = '0' }|.
WRITE / |{ 'Right'    WIDTH = 20 ALIGN = RIGHT  PAD = '0' }|.
```

## IV. Case

```
WRITE / |{ 'Text' CASE = (cl_abap_format=>c_raw) }|.
WRITE / |{ 'Text' CASE = (cl_abap_format=>c_upper) }|.
WRITE / |{ 'Text' CASE = (cl_abap_format=>c_lower) }|.
```

## V. ALPHA conversion

```
DATA(lv_vbeln) = '0000012345'.
WRITE / |{ lv_vbeln ALPHA = OUT }|.      "or use ALPHA = IN to go in other direction
```

## VI. Date conversion

```
WRITE / |{ pa_date DATE = ISO }|.      "Date Format YYYY-MM-DD
WRITE / |{ pa_date DATE = User }|.     "As per user settings
WRITE / |{ pa_date DATE = Environment }|. "Formatting setting of language environment
```

## 10. Loop at Group By

## I. Definition

Follow

 Like

 RSS Feed

```

LOOP AT itab result [cond] GROUP BY key ( key1 = dobj1 key2 = dobj2 ...
      [gs = GROUP SIZE] [gi = GROUP INDEX] )
      ASCENDING|DESCENDING [AS TEXT]]
      [WITHOUT MEMBERS]
      {INTO group}|{ASSIGNING <group>}}]

[LOOP AT GROUP group|<group>
...
ENDLOOP.]
...
ENDLOOP.

```

## II. Explanation

The outer loop will do one iteration per key. So if 3 records match the key there will only be one iteration for these 3 records. The structure “group” (or “<group>”) is unusual in that it can be looped over using the “LOOP AT GROUP” statement. This will loop over the 3 records (members) of the group. The structure “group” also contains the current key as well as the size of the group and index of the group ( if GROUP SIZE and GROUP INDEX have been assigned a field name). This is best understood by an example.

## III. Example

With 7.40



With 7.40

Follow



Like



RSS Feed

TYPES: BEGIN OF ty\_employee,

name TYPE char30,

role TYPE char30,

age TYPE i,

END OF ty\_employee,

ty\_employee\_t TYPE STANDARD TABLE OF ty\_employee WITH KEY name.

DATA(gt\_employee) = VALUE ty\_employee\_t(

( name = 'John' role = 'ABAP guru' age = 34 )

( name = 'Alice' role = 'FI Consultant' age = 42 )

( name = 'Barry' role = 'ABAP guru' age = 54 )

( name = 'Mary' role = 'FI Consultant' age = 37 )

( name = 'Arthur' role = 'ABAP guru' age = 34 )

( name = 'Mandy' role = 'SD Consultant' age = 64 ) ).

DATA: gv\_tot\_age TYPE i,

gv\_avg\_age TYPE decfloat34.

“Loop with grouping on Role

With 7.40

Follow



Like



RSS Feed

```
LOOP AT gt_employee INTO DATA(ls_employee)
```

```
GROUP BY ( role = ls_employee-role
```

```
size = GROUP SIZE
```

```
index = GROUP INDEX )
```

```
ASCENDING
```

```
ASSIGNING FIELD-SYMBOL(<group>).
```

```
CLEAR: gv_tot_age.
```

```
"Output info at group level
```

```
WRITE: / |Group: { <group>-index } Role: { <group>-role WIDTH = 15 }|
```

```
& | Number in this role: { <group>-size }|.
```

```
"Loop at members of the group
```

```
LOOP AT GROUP <group> ASSIGNING FIELD-SYMBOL(<ls_member>).
```

```
gv_tot_age = gv_tot_age + <ls_member>-age.
```

```
WRITE: /13 <ls_member>-name.
```

```
ENDLOOP.
```

```
"Average age
```

With 7.40

Follow

$$gv\_avg\_age = gv\_tot\_age / \text{<group>-size.}$$


 Like



 [RSS Feed](#)

**WRITE:** / |Average age: { gv\_avg\_age }|.

SKIP.

ENDLOOP.

#### IV. Output

Group: 1    Role: ABAP guru    Number in this role: 3

John

Barry

Arthur

Average age: 40.666666666666666666666666666667

Group: 2    Role: FI Consultant    Number in this role: 2

Alice

Mary

Average age: 39.5

Group: 3    Role: SD Consultant    Number in this role: 1

Mandy

Average age: 64

## 11. Classes/Methods

## I. Referencing fields within returned structures

**Before 7.40**

Follow

 Like RSS Feed**Before 7.40**

```
DATA: ls_lfa1 TYPE lfa1,
lv_name1 TYPE lfa1-name1.

ls_lfa1 = My_Class=>get_lfa1( ).
lv_name1 = ls_lfa1-name1.
```

**With 7.40**

```
DATA(lv_name1) = My_Class=>get_lfa1( )-name1.
```

## II. Methods that return a type BOOLEAN

**Before 7.40**

```
IF My_Class=>return_boolean( ) = abap_true.
...
ENDIF.
```

**With 7.40**

```
IF My_Class=>return_boolean( ).
...
ENDIF.
```

NB: The type “BOOLEAN” is not a true Boolean but a char1 with allowed values X,- and <blank>.

Using type “FLAG” or “WDY\_BOOLEAN” works just as well.

## III. NEW operator

This operator can be used to instantiate an object.

**Before 7.40**

Follow

 Like RSS Feed**Before 7.40**

```
DATA: lo_delivs TYPE REF TO zcl_sd_delivs,
      lo_deliv TYPE REF TO zcl_sd_deliv.

CREATE OBJECT lo_delivs.
CREATE OBJECT lo_deliv.

lo_deliv = lo_delivs->get_deliv( lv_vbeln ).
```

**With 7.40**

```
DATA(lo_deliv) = new zcl_sd_delivs( )->get_deliv( lv_vbeln ).
```

## 12. Meshes

Allows an association to be set up between related data groups.

### I. Problem

Given the following 2 internal tables:

```
TYPES: BEGIN OF t_manager,
       name TYPE char10,
       salary TYPE int4,
END OF t_manager,

tt_manager TYPE SORTED TABLE OF t_manager WITH UNIQUE KEY name.

TYPES: BEGIN OF t_developer,
       name TYPE char10,
       salary TYPE int4,
       manager TYPE char10, "Name of manager
```

```
END OF t_developer,
```

```
t_developer TYPE SORTED TABLE OF t_developer WITH UNIQUE KEY name.
```

Follow



Like

populated as follows:



RSS Feed

Row	Name[C(10)]	Salary[I(4)]
1	Jason	3000
2	Thomas	3200

Row	Name[C(10)]	Salary[I(4)]	Manager[C(10)]
1	Bob	2100	Jason
2	David	2000	Thomas
3	Jack	1000	Thomas
4	Jerry	1000	Jason
5	John	2100	Thomas
6	Tom	2000	Jason

Get the details of Jerry's manager and all developers managed by Thomas.

## II. Solution

### With 7.40

```
TYPES: BEGIN OF MESH m_team,
managers  TYPE tt_manager ASSOCIATION my_employee TO developers
           ON manager = name,
developers TYPE tt_developer ASSOCIATION my_manager TO managers
           ON name = manager,
END OF MESH m_team.
```

Follow

 Like RSS Feed

## With 7.40

```
DATA: ls_team TYPE m_team.
ls_team-managers = lt_manager.
ls_team-developers = lt_developer.

*Get details of Jerry's manager *

"get line of dev table
ASSIGN lt_developer[ name = 'Jerry' ] TO FIELD-SYMBOL(<ls_jerry>).
DATA(ls_jmanager) = ls_team-developers\my_manager[ <ls_jerry> ].
WRITE: / |Jerry's manager: { ls_jmanager-name }|,30

           |Salary: { ls_jmanager-salary }|.

"Get Thomas' developers
SKIP.
WRITE: / |Thomas' developers:|.

"line of manager table
ASSIGN lt_manager[ name = 'Thomas' ] TO FIELD-SYMBOL(<ls_thomas>).
LOOP AT ls_team-managers\my_employee[ <ls_thomas> ]
    ASSIGNING FIELD-SYMBOL(<ls_emp>).
    WRITE: / |Employee name: { <ls_emp>-name }|.
ENDLOOP.
```

### III. Output

Jerry's manager: Jason      Salary: 3000

Thomas' developers:

Employee name: David

Employee name: Jack

Follow

Employee name: John

 Like

 RSS Feed

## 3. Filter

Filter the records in a table based on records in another table.

### I. Definition

```
... FILTER type( itab [EXCEPT] [IN ftab] [USING KEY keyname]  
WHERE c1 op f1 [AND c2 op f2 [...]] )
```

### II. Problem

Filter an internal table of Flight Schedules (SPFLI) to only those flights based on a filter table that contains the fields Cityfrom and CityTo.

### III. Solution

With 7.40



Follow



Like



RSS Feed

Alert Moderator

With 7.40

```
TYPES: BEGIN OF ty_filter,
cityfrom TYPE spfli-cityfrom,
cityto   TYPE spfli-cityto,
f3       TYPE i,
END OF ty_filter,
ty_filter_tab TYPE HASHED TABLE OF ty_filter
              WITH UNIQUE KEY cityfrom cityto.

DATA: lt_splfi TYPE STANDARD TABLE OF spfli.

SELECT * FROM spfli APPENDING TABLE lt_splfi.
```

Assigned tags

- 740
- ABAP Development
- SAP NetWeaver
- abap
- document
- overveiw
- reference

```
DATA(lt_filter) = VALUE ty_filter_tab( f3 = 2
                                         ( cityfrom = 'NEW YORK'  cityto = 'SAN FRANCISCO' )
                                         ( cityfrom = 'FRANKFURT' cityto = 'NEW YORK' ) ).

DATA(lt_myrecs) = FILTER #( lt_splfi IN lt_filter
                             WHERE cityfrom = cityfrom
                             AND cityto = cityto ).

"Output filtered records

LOOP AT lt_myrecs ASSIGNING FIELD-SYMBOL(<ls_rec>).
  WRITE: / <ls_rec>-carrid,8 <ls_rec>-cityfrom,30
          <ls_rec>-cityto,45 <ls_rec>-detime
```

View more... >

Similar Blog Posts

ABAP Language News for Release 7.40

By Horst Keller Jul 22, 2013

Note: using the keyword “EXCEPT” (see definition above) would have returned the exact opposite records i.e all records EXCEPT for those those returned above.

ABAP Language News for Release 7.40, SP08

By Horst Keller Oct 13, 2014

---

[Follow](#)  
[ABAP News for Release 7.50 - What is ABAP 7.50?](#)

By Horst Keller Oct 20, 2015



Like

## Related Questions

---

[OO ABAP 7.50 quick reference](#)

By Himansu Gyalá Apr 16, 2020

---

[About reference book](#)

By Former Member Jan 08, 2007

---

[Difference between ABAP with Netweaver 7.40 and ABAP with Netweaver 7.02](#)

By Manu Kapur Apr 20, 2015

## 65 Comments

---

You must be [Logged on](#) to comment or reply to a post.



Jitendra Soni

October 25, 2015 at 1:42 pm

Hi Jeffrey,

Very informative blog.

Below syntax is not working for me.

```
"SELECT * FROM dbtab INTO TABLE @DATA(lt_dbtab) WHERE field1 = @lv_field1."
```

ABAP version:

SAP\_BASIS 740 0007 SAPKB74007 0000 - SAP Basis Component

SAP\_ABA 740 0007 SAPKA74007 0000 - Cross-Application Component

Like 0 | Share



**Jeffrey Towell** | Blog Post Author

October 26, 2015 at 8:42 am

Thanks Jitendra.

I am not sure which bits of ABAP 7.40 come in with exactly which version but here is some working code. If this does not work on your box then its fair to say you do not have the relevant version yet.

```
DATA: lv_bukrs type bukrs VALUE '0001'.
```

```
SELECT * FROM t001 INTO TABLE @DATA(lt_t001)
```

```
WHERE bukrs = @lv_bukrs.
```

Like 0 | Share



**Christiano José Beltrão Magalhães**

October 26, 2015 at 12:20 pm

Hi Jitendra/Jeffrey,

the new open sql syntax was created in ABAP 7.40 SP05 and enhanced in SP08. More information in [ABAP News for 7.40, SP08 - Open SQL](#).

Jeffrey, great blog... very useful.

BR,

Christiano.

Like 0 | Share

 RSS Feed



**Paul Bakker**

October 25, 2015 at 9:52 pm

Thanks for going to so much effort! Very interesting reading.

Unfortunately some of the code (inside the black borders) is truncated on the right hand side. But I think we can work it out

cheers

Paul

Like 0 | Share



**Jeffrey Towell** | Blog Post Author

October 26, 2015 at 12:10 am

Thanks for your comments Paul.

Was also concerned about truncation on the right but found that if you click on the text and drag to the right that it all becomes visible. Alternatively the scroll bar at the bottom works but it's a bit inconvenient scrolling down to find it.

Cheers,

Jeff

Like 0 | Share

**Former Member**

October 26, 2015 at 5:03 am

Very much useful document Paul!

Like 0 | Share



RSS Feed

**Manu Kapur**

October 26, 2015 at 11:22 am

Brilliant. Thanks for sharing.

Like 0 | Share

**Raphael Pacheco**

October 26, 2015 at 11:45 am

Great post [Jeffrey](#) !

Just a suggestion ... I believe that would be less harmful to the blocks with commands have the edges a little thinner.

Like 0 | Share

**Jeffrey Towell** | Blog Post Author

October 26, 2015 at 12:05 pm

Good point Raphael! If I can find a relatively easy way to do that I think I will.

Like 0 | Share



Former Member

October 27, 2015 at 1:18 pm

Brilliant, looking forward for future blogs..

Like 0 | Share



RSS Feed



Former Member

October 28, 2015 at 12:20 pm

very helpful, can't wait to use some of the inline expressions

Like 0 | Share



Guy Lamoureux

October 28, 2015 at 1:34 pm

Very Interesting. But I see that clarity and "ease of reading" continues to be vastly underestimated and undervalued. ABAP is going to the dark side



Like 1 | Share



Jeffrey Towell | Blog Post Author

October 29, 2015 at 2:30 am

Guy, I thought the exact same thing at first along with others I have chatted to. However, after using it a while I realise it becomes more clear as you get more familiar with the syntax. After years of using the old syntax it has become so familiar to us that it feels like we have to think too much to understand what is being coded in the new syntax. Soon it will be second nature to you and hence easy to read.

Like 0 | Share



**Guy Lamoureux**  
October 29, 2015 at 11:37 am

Hi Jeffrey,

"after using it a while" the problem is right here. Not everybody is an ABAP programmer and not everybody programs in ABAP on a regular base. I've seen a lot of functional analyst who can follow what's going on in an ABAP program. They do it for many reasons but it's part of their job and the more we change the language to something more obscure, the less they will be able to do it. They will need help from ABAP programmers. This will slow down the process.

On my part, I've worked as an ABAP programmer for 10 years, followed by 10 years of BW developement. I don't write ABAP code on a regular base. This new syntax will keep being obscure.

Like 0 | Share



**Christoph Schreiner**  
October 29, 2015 at 7:59 am

Nice overview, thanks for sharing it with us!

Like 0 | Share



**Former Member**  
November 8, 2015 at 10:56 am

Great job! Thank you for making our life easy...

Like 0 | Share

**Aslam MD**

November 18, 2015 at 7:17 am

Hi Jeffrey,

Very informative material.

Thank you very much

Like 0 | Share

**Former Member**

November 18, 2015 at 8:36 am

Big THX :-).

Just sent this link to the whole team :-).

Like 0 | Share

**Former Member**

November 20, 2015 at 2:32 pm

When I do an inline Declaration of an internal table

```
SELECT ... FROM ... INTO TABLE @data(It_data).
```

Is there also some way, to have this as a sorted / hashed table or at least add secondary keys?

Like 1 | Share

**Former Member**

November 23, 2015 at 5:17 am



Not that I'm aware of Jakob. If you create a "type" of the kind you want with sorting etc. and call it say ty\_mytab you could do a conversion using CONV:

```
TYPES ty_mytab TYPE SORTED TABLE OF t001w WITH NON-UNIQUE KEY fabkl.
```

```
SELECT * FROM t001w INTO TABLE @DATA(lt_t001w).
```

```
DATA(lt_new_tab) = CONV ty_mytab( lt_t001w ).
```

However, this does not save you any time/typing compared to selecting directly into your defined internal table:

```
TYPES ty_mytab TYPE SORTED TABLE OF t001w WITH NON-UNIQUE KEY fabkl.
```

```
DATA: lt_new_tab TYPE ty_mytab.
```

```
SELECT * FROM t001w INTO TABLE lt_new_tab.
```

Like 0 | Share



**Wilbert Sison**

November 26, 2015 at 2:49 am

Nice collection Jeffrey!

Like 0 | Share



**Former Member**

November 26, 2015 at 2:52 am

Cheers Wilbo!

Like 0 | Share



**Michael Calekta**  
May 18, 2016 at 10:13 am

Thanks for your effort Jeffrey!

Yet there's a little mistake in the Mesh-Example:

```
ASSIGN lt_developer[ name = 'Jerry' ] TO FIELD-SYMBOL(<ls_jerry> ).  
DATA(ls_jmanager) = ls_team-developers\my_manager[ jerry ].
```

Second line should read instead:

```
DATA(ls_jmanager) = ls_team-developers\my_manager[ <ls_jerry> ].
```

Same is true for "thomas" a few lines below.

Nevertheless this is the first example I found, where the advantage of meshes can be seen.

All the best

Michael

Like 0 | Share



**Jeffrey Towell** | Blog Post Author

May 19, 2016 at 1:03 pm

Thanks for pointing that out Michael. I have corrected that.

The amazing thing is that the code is a copy and paste from a working program I wrote and still have. I've noticed the "<" and ">" get stripped off my field symbols in this document before. My theory is that when it gets converted to HTML that the field symbols sometimes look like HTML tags because they are between the <>. As such they are sometimes stripped out by this conversion to HTML.

That's my theory anyway.

Thanks again.

Like 0 | Share

Michael Calekta



May 19, 2016 at 1:17 pm

Sorry to interrupt again, but it was not only the <> missing, which you have corrected, but also the 1s\_ which is still missing. I don't think this can get lost by an html-conversion-error. Perhaps a missing definition and value assignment from the original coding.

I have copied the example and tried it, and it really works fine, once I could eliminate the syntax-errors because of the missing letters.

Like 0 | Share

**Jeffrey Towell** | Blog Post Author

May 20, 2016 at 4:45 am

Interruption appreciated as you are correct that I forgot to add the "1s\_" in. However, I can assure you that the original code has both the "<>" and the "1s\_" in. The HTML issue has caused problems in other parts of this document which is why I know about it. In the "Loop at Group By" section it would not let me save the code I added. I finally added the code into the document word by word (i.e. saving after each word) and discovered it was a field symbol causing the problem. When I renamed the field symbol it saved.

Like 0 | Share

**Former Member**

June 8, 2016 at 9:36 am

Thanks for documenting all the new changes. This comes as a helpful doc for all who wants to know the new features of ABAP Programming. The Inline Declaration is a very helpful feature of ABAP 740 and it solves huge efforts of developer.

Regards,

Vinay Mutt

Like 0 | Share

[Follow](#)**Martin Neuß**

June 16, 2016 at 5:19 am

... wonderful !

I am just trying to gather some Information about Netweaver 7.40 ABAP for a forthcoming inhouse training here in our company, and found out soon that the original SAP samples are hardly helpful.

Your examples are really straightforward, easy to understand and useful for "real life" developers.

Thank you !

Regards,

Martin Neuss

Like 0 | Share

**Former Member**

August 18, 2016 at 10:08 am

Hi, experts. How can i fill itab with corresponding fields from structure variable and one field from another table using one statement ? my example:

```
data(RT_CONFIG_PERS_DATA) =  
VALUE BSP_DLCT_PERS( for wa_touser in TOUSER  
( CORRESPONDING #( RS_CONFIG_PERS_DATA EXCEPT PERS_FOR_USER ) PERS_FOR_USER = wa_touser-low ) ).
```

this statement gives syntax error.

so i am just using classic code:

```

data RT_CONFIG_PERS_DATA type BSP_DLCT_PERS.
LOOP AT TOUSER INTO DATA(wa_touser) .
  APPEND INITIAL LINE TO rt_config_pers_data ASSIGNING FIELD-SYMBOL(<fs>).
  MOVE-CORRESPONDING rs_config_pers_data to <fs>.
  <fs>-pers_for_user = wa_touser-low.
ENDLOOP.

```

is it possible to do such actions in one statement ?

Like 0 | Share



**Jeffrey Towell** | Blog Post Author

August 19, 2016 at 12:07 am

Hi Konstantin,

Its possible to get it on one line by using each component of the structure instead of the "CORRESPONDING". In your case this would look like:

```

DATA(rt_config_pers_data) =
VALUE bsp_dlct_pers( FOR wa_touser IN touser
( pers_for_user = wa_touser-low
  component    = rs_config_pers_data-component
  viewname     = rs_config_pers_data-viewname
  role_key     = rs_config_pers_data-role_key
  component_usage = rs_config_pers_data-component_usage
  object_type  = rs_config_pers_data-object_type
  object_sub_type = rs_config_pers_data-object_sub_type
  changed_by   = rs_config_pers_data-changed_by
  changed_at   = rs_config_pers_data-changed_at
  config       = rs_config_pers_data-config

```

```
parameters    = rs_config_pers_data-parameters  
config_type   = rs_config_pers_data-config_type  
invalid_flag  = rs_config_pers_data-invalid_flag  
marking_flag  = rs_config_pers_data-marking_flag  
check_flag    = rs_config_pers_data-check_flag ) ).
```

Of course your "classic code" is better not just because the above is longer but also because the above will not work if there is ever a change to the structure bsp\_dlct\_pers.

Like 0 | Share



**PRUTHVIRAJ DAYAM**

August 30, 2016 at 2:09 pm

Cant we use Filter with Non-Key fields! .. any manipulation possible with declaration?!

Like 0 | Share



**Rohit Gupta**

February 23, 2017 at 6:11 pm

Are constructor operators are better in performance ? or It is just a different way of writing the code.

Like 0 | Share

**Ramesh Kothapally**



March 17, 2017 at 9:27 am

Hi Jeffrey,

Thanks for sharing very informative document with us. This blog helps for all who want to know new features and techniques in ABAP 7.4 programming and helpful to getting started with ABAP 7.4/7.5

Thank you very much.

Thanks and Regards,

Ramesh Kothapally

Like 0 | Share



Sawyer Peng

July 12, 2017 at 6:57 am

Great blog, many thanks.

Like 0 | Share



Sawyer Peng

July 12, 2017 at 7:14 am

There is a typo for the select into table:

```
SELECT * FROM dbtab  
      INTO TABLE DATA(itab)  
      WHERE fld1 = @lv_fld1.
```

it should be:

```
SELECT * FROM dbtab
```

```
INTO TABLE @DATA(itab)
      WHERE fld1 = lv_fld1.
```

Please help to correct it.

Like 0 | Share



**Anurag Kashyap**

November 30, 2017 at 1:12 pm

This can be written also as :

SELECT \* FROM dbtab INTO TABLE @DATA(itab WHERE FLD1 = @P\_FIELD1. " P\_FIELD1 – Is the value coming from selection screen.

Like 0 | Share



**sridhar reddy**

July 20, 2017 at 7:18 pm

Thanks for the wonderful blog Jeffrey.

BTW, how do we READ table using binary search with the new syntax?

Like 0 | Share



**Freek Cavens**

July 24, 2017 at 1:50 pm

In the new syntax you would probably use a sorted or hashed table. A problem that I have encountered numerous times with the binary search is that the table is not sorted correctly (often because the sort order is changed in a later adjustment of the code and the binary search is overlooked), leading to an incorrect result. Using sorted table makes sure that the sorting of the table is correct. If you need to read the table using different access paths, you can just declare multiple keys.



it would be something like this :

data : lt\_kunnr TYPE HASHED TABLE OF kna1 WITH UNIQUE KEY kunnr

with non-unique sorted key k\_city components ORTO1,

\*\*Get a specific customer (if no key is specified, the default key is used, in this case the hashed key)

assign lt\_kunnr[ kunnr = '1000023653' ] to field-symbol(<ls\_kunnr>).

\*\*Get the first customer of a city, using the sorted key

assign lt\_kunnr[ key k\_city orto1 = 'BRUSSELS' ] to <ls\_kunnr>.

Like 2 | Share



Former Member

August 3, 2017 at 9:35 am

Really very good informative post.....Thanks alot

Like 0 | Share



Ruthiel Trevisan

November 14, 2017 at 11:20 am

Thanks a lot [Jeffrey Towell](#) ! This article is amazing!

I'll try to implement this features on my developments!

Like 1 | Share

**Antonis Ioannidis**

February 2, 2018 at 2:54 pm

First of all, **Great Job Jeffrey Towell!** This is an excellent post providing very useful information. Thank you!

But I cannot stop to wonder, are those new ways of writing any better than the older ones performance-wise?

In my point of view, if there is no actual performance gain by using the new methods, apart from some new additions like CONV which are indeed very useful, it seems to me that it will just make the code a lot more complex for other programmers, not familiar with the new methods, to read.

What are your thoughts on this?

Like 0 | Share

**Michael Rudolph**

March 9, 2018 at 2:18 pm

Hi Antonis,

maybe not better than older ones performance-wise. But the way you can code now saves a lot of performance while your typing! Don't forget that every letter you have not to type are saving time. Isn't it? Sure at the beginning it is sometimes hard to read but it becomes clear after a while. Now ABAP is a little bit closer to other programming languages.

regards

Micha

Like 1 | Share

**Jeffrey Towell | Blog Post Author**

April 23, 2018 at 1:45 am

Hi Antonis,

I haven't tested the performance of old vs new syntax however I would be surprised if SAP have made the new syntax work slower.

Presumably where one line of code in the new syntax does the work of multiple lines in the old then the new syntax will be quicker as it will be optimized for the specific function it is carrying out.

In terms of readability it actually becomes easier to read once you are familiar with the syntax. Taking your CONV example, previously you might have passed a value from one variable (say Type I) to another (say Char3) to convert it. While reading this you would not know for sure a conversion is taking place. A value might just be shared between two variables of the same type. With CONV it is obvious what the intent is.

Old: var2 = var1. (Is this a conversion or just a shared value between vars of the same type ?)

New: var2 = conv char3( var1 ).

Like 0 | Share

 RSS Feed



**Himansu Gyala**

May 15, 2018 at 8:28 am

Much Informative

Like 0 | Share



**Ebrahim Hatem**

June 20, 2018 at 3:34 pm

it is really interesting and anybody can find all information which ich related to ABAP 740. But I have an comment to the II. Methods that return a type BOOLEAN.

```
IF My_Class=>return_boolean( ). " True ('X')
```

```
...
```

```
ENDIF.
```

```
IF NOT My_Class=>return_boolean( ). " false empty
```

```
...
```

```
ENDIF.
```

Regards

Ebrahim

Like 0 | Share

 RSS Feed**Bärbel Winkler**

June 22, 2018 at 12:34 pm

Rather belated thanks from me as well, [Jeffrey Towell](#) for this detailed and very helpful list (h/t [Jonathan Capps](#) whose [recent post](#) linked to yours)! This list will help me to wrap my head around the (no longer really) new options to write ABAP-statements. I however also share some misgivings others have mentioned earlier, namely that this shortened and arguably streamlined way to write ABAP-code is no longer quite as easy to read and parse - esp. for people new to programming or to folks mostly working on the functional and customizing part of SAP within IT. With the old "long-form" ABAP with spelled out statements, it was usually possible for a technically-minded colleague to at least understand the gist of what is going on in a program, while either looking at the code in SE38/SE80 or during debugging. Considering that I'm having a hard time quickly remembering and understanding what I'm looking at with many of the "new" constructs I can imagine how even more confusing this might look for non-developers.

So, I'm wondering if there's perhaps some additional information needed to highlight the advantage(s) of the new constructs apart from potentially having to type a few characters less? One such advantage might be performance or another heightened security. For me, brevity is not always a bonus and longer but more self-explanatory statements can make life easier once the time comes that changes need to be applied.

Cheers

Baerbel

Like 0 | Share

**Jeffrey Towell** | Blog Post Author

February 1, 2019 at 5:01 am

Apologies Barbel. My response is even more belated than your comment 😊

I think the readability issues are due to us not being familiar with the new syntax. If, like me, you are still looking up some of the syntax when coding then reading existing code will also be slower. However, a given statement in the new syntax can only have one meaning and once we are "fluent" in the syntax its as easy to read as to write.

Your point about non-developers is well taken. Where non-developers have spent years slowly learning what is now legacy syntax they will now be impeded when trying to read/debug code in new syntax.

If I wrote: "Thx 4 ur comment" it would save me 8 characters. If I was writing this statement frequently it would start saving me time and I'd be able to read it as quickly as the full version.

I cannot speak to performance in terms of running the code. But in terms of debugging it is quicker as we now have one line of code doing what multiple lines of code used to do. For example a 15 line case statement becomes a 1 line COND statement that can be stepped over with one F6 in debug mode. I also think the COND is as easy to read.

Jeff

Like 0 | Share



Jayaprakash H J

December 21, 2018 at 1:43 pm

Hi,

Under many headings i could only find **Before 7.40** . There is nothing in **With 7.40** .

Please help.

Regards,

Jp

Like 0 | Share

**Srikanth Thogiti**

May 1, 2019 at 3:44 pm

Thanks for sharing the knowledge.

It is really a useful info and It changes our job easy, especially with FILTER, GROUP, VALUE, FOR etc.

Like 0 | Share

**Vimal Sharma**

July 18, 2019 at 4:18 am

How to pass inline declared internal table to a subroutine. e.g.

```
SELECT kappl,
```

```
objky,
```

```
kschl,
```

```
spras,
```

```
FROM nast
```

```
INTO TABLE @DATA(gt_nast) .
```

```
IF sy-subrc is initial.
```

```
Perform get_entries using gt_nast
```

```
ENDIF.
```

```
"Declaration of perform
```

```
GET_ENTRIES USING p_nast type ????
```

If declare a type and then tries to pass it here , it says type mismatch . So what to do while declaring a perform for internal table fetched with literals.

Like 0 | Share

**Sandra Rossi**



July 18, 2019 at 4:27 am

Eclipse ADT "quick fixes" to declare the variable explicitly (DATA BEGIN OF ...), change DATA into TYPES, and use that type name...

Like 2 | Share



Like



Renuka Behara

December 17, 2019 at 6:09 pm

Nice blog.. All at one place.

Like 0 | Share



Vishal Kumar

May 2, 2020 at 5:57 am

Hello

Can someone help me with the syntax error in the attached code ?

It gives error "No components exists with the name 'FOR' "

TYPES:

```
BEGIN OF ty_for_final,  
  vbeln TYPE vbeln_va,  
  vbtyp TYPE vbak-vbtyp,  
  posnr TYPE vbap-posnr,  
END OF ty_for_final.
```

```
DATA(li_for_final) = VALUE ty_for_final( FOR wa_vbak IN for_vbak  
                                         FOR wa_vbap IN for_vbap WHERE ( vbeln = wa_vbak-vbeln )  
                                         ( vbeln = wa_vbak-vbeln vbtyp = wa_vbak-vbtyp posnr = wa_vbap-posnr ) ).
```

Thanks

Like 0 | Share



Sandra Rossi

May 2, 2020 at 12:01 pm

Yes, but only if you ask the question in the forum...

Like 0 | Share



Vishal Kumar

May 2, 2020 at 6:01 am

Getting error with New Operator as well.

TYPES:

```
BEGIN OF ty_ord,  
  vbeln TYPE vbeln_va,  
  posnr TYPE posnr_va,  
  vbtyp TYPE vbak-vbtyp,  
END OF ty_ord.
```

DATA:

```
lv_new_table TYPE REF TO DATA.
```

```
lv_new_table = NEW ty_ord( ( vbeln = '000000001' posnr = '0000001' vbtyp = 'L' ) ( vbeln = '000000002' posnr = '0000002' vbtyp = 'L' ) )
```

Like 0 | Share



Rajesh Nair

May 10, 2020 at 6:14 pm



Hi Vishal,

True. This would be an error since the type ty\_ord is a structure.

```
lv_new_table = NEW ty_ord( ( vbeln = '000000001' posnr = '0000001' vbttyp = 'L' ).
```

This would work. If you want multiple entries, then you could declare a table type as follows and then your code would work.

```
TYPES ty_t_ord TYPE STANDARD TABLE OF ty_ord WITH EMPTY KEY.
```

```
lref_new_table = NEW ty_t_ord( ( vbeln = '000000001' posnr = '0000001' vbttyp = 'L' ) ( vbeln = '000000002' posnr = '0000001' vbttyp = 'L' ) ).
```

Regards,

Rajesh P Nair

Like 0 | Share



**Sandra Rossi**

May 10, 2020 at 7:32 pm

the first one will not work because you still define two opening parentheses ( (

Instead use only one opening parenthesis:

```
lv_new_table = NEW ty_ord( vbeln = '000000001' posnr = '0000001' vbttyp = 'L' ).
```

Like 0 | Share



**Rajesh Nair**

May 11, 2020 at 11:20 pm

Hi Sandra,

You are correct. That was a typo, I have copied from Vishal's message and removed the closing parenthesis, but not the opening one. I was suggesting Vishal that multiple entries will not work for the type ty\_ord since it represents a flat structure and we can use multiple entries only if we use a table type of ty\_ord.

Regards,

Rajesh P Nair

Like 0 | Share

Follow

**RAMNIK DHAR**

June 10, 2020 at 11:31 am

Hi Guys,

Suppose I have a table with only one column and my requirement is to get all the contents of the table in a string separated by (,) and ending with (.)  
e.g. Value1, Value2, Value3.

Any pointers on how to do this with the new syntax without concatenating.

Like 0 | Share

**Joachim Rees**

September 25, 2020 at 6:22 am

Huh, seems I missed this blog so far (found it now via <https://blogs.sap.com/2018/09/13/abaps-new-syntax-tips-from-experience/>) - this seems like a very helpful resource, thanks!

Like 2 | Share

**Ankit Maskara**

October 19, 2020 at 6:51 am

Hi [Joachim Rees](#),

Thanks a lot for recommending my blog. You are also an inspiration for many of us.

Thanks and Regards,

Ankit Maskara.

Like 1 | Share

**Paweł Karp**

May 14, 2021 at 10:59 am

Thank you very much! Incredibly useful post!

I just have only a small question - is in the first table with "inline declarations" not missing a sign "@"?

```
SELECT * FROM dbtab  
INTO TABLE @DATA(itab)  
WHERE fld1 = @lv_fld1.
```

Best regards

Paweł

Like 0 | Share

**Joachim Rees**

May 14, 2021 at 2:36 pm

Yes, I think you are right!

Like 0 | Share

**Jeffrey Towell** | Blog Post Author

May 24, 2021 at 3:45 am

Thanks Paweł (and Joachim).

I've no idea how that slipped through the cracks for the last 5.5 years this article has been up. 😊

I've made the correction.

Like 0 | Share

**Aditya Sharma**

July 8, 2021 at 5:08 pm

When you are working with such a client where issues arise daily, they have to be met daily.

In addition your team get 3-4 Functional specs on daily.  
Stringent timelines have to be submitted to client.  
How can one motivate team to do these adornments ?

I have been requesting sap ,please with joined hands,finalize your product.  
What you want to give to others ?  
You have been used to work in abap with a particular style of coding, why you will change it at first place ?

These things are not enhancements but an open outlet journey for some people to leave field of abap altogether.

You work with team with diverse kind of people. Some teams even are more than 30-50 abaper count .  
Why make life of others hard to fulfill these stupid desires which final equate to same sense ?.  
And mind you its important to understand.You are in field of AI,Machine learning,Deep learning neural networks.But what i think in this case you are trying to prove that human brain is different.  
Just a new version is released,does that mean its the fault of customer or he should be penalized for that ?

Like 0 | Share

### Find us on

Privacy	Terms of Use
Legal Disclosure	Copyright
Trademark	Cookie Preferences

Newsletter

Support

Follow

 Like

 RSS Feed