In SAP whenever the data is to be taken from Internal Table to any variable in Code for further processing, Loop or Read Table are generally used to get that data from Internal Table to work area and then move to another variable, so this effort can be lessened by using the new syntax.

**\*Get line from table to Work Area.**

\*&---------------------------------------------------------------------\*  
\*& Report ZYNY\_NEW\_ABAP\_03  
\*&---------------------------------------------------------------------\*  
\*&  
\*&---------------------------------------------------------------------\*  
REPORT ZYNY\_NEW\_ABAP\_03.  
  
\*Old Syntax.  
READ TABLE itab INTO DATA(wa) WITH KEY fld1 = var1.  
  
\*New Syntax.  
DATA(wa) = itab[ fld1 = var1 ].

**\*Read table with Index 1.**

\*&---------------------------------------------------------------------\*  
\*& Report ZYNY\_NEW\_ABAP\_03  
\*&---------------------------------------------------------------------\*  
\*&  
\*&---------------------------------------------------------------------\*  
REPORT ZYNY\_NEW\_ABAP\_03.  
  
\*Old Syntax.  
READ TABLE itab INTO DATA(wa) index 1.  
  
\*New Syntax.  
DATA(wa) = itab[ 1 ].

Get line from table to Work Area.

**Old Syntax.**  
READ TABLE itab INTO DATA(wa) WITH KEY fld1 = var1.**New Syntax.**  
DATA(wa) = itab[ fld1 = var1 ].

Read table with Index 1.

**Old Syntax.**  
READ TABLE itab INTO DATA(wa) index 1.**New Syntax.**  
DATA(wa) = itab[ 1 ].

Get a particular field from one table to another variable.

**Old Syntax.**  
READ TABLE itab INTO wa WITH KEY fld1 = var1.  
IF sy-subrc EQ 0.  
lv\_var2 = wa-var2.  
ENDIF.**New Syntax.**  
lv\_var2 = itab[ fld1 = var1 ]-var2.

Check particular Value is in Internal Table.

**Old Syntax.**  
READ TABLE itab TRANSPORTING NO FIELDS WITH KEY fld1 = var1.  
IF sy-subrc = 0.ENDIF**.New Syntax.**  
IF line\_exists( itab[ fld1 = var1 ] ).ENDIF.

Get Index Number of Particular Entry From Internal Table.

**Old Syntax.**  
READ TABLE itab TRANSPORTING NO FIELDS fld1 = var1.   
IF sy-subrc eq 0  
lv\_tabix = sy-tabix.  
ENDIF.**New Syntax.**  
lv\_tabix = line\_index( itab[ fld1 = var1 ] ).

Modify Entry in Table.

**Old Syntax.**READ TABLE itab ASSIGNING <lfs\_tab> fld1 = var1.   
IF sy-subrc eq 0  
<lfs\_tab>-fld2 = 'Text'.  
ENDIF.**New Syntax.**itab[ fld1 = var1 ]-fld2 = 'Text'.

Note: While Using these Syntaxes in case of Table Expression use with Try Catch with ***cx\_sy\_itab\_line\_not\_found***and Entry.

***3.Data Operations:***Here are some of the new Syntax which can be used for manipulating, changing, and storing data according to requirement.

Adding or Removing Leading Zeros to a Variable.

**Old Synatx.**   
For Removing Leading Zeros  
CALL FUNCTION ‘CONVERSION\_EXIT\_ALPHA\_OUTPUT’  
 EXPORTING  
 input = var1  
 IMPORTING  
 OUTPUT = var1 .  
For Adding Leading Zeros  
CALL FUNCTION ‘CONVERSION\_EXIT\_ALPHA\_INPUT’  
 EXPORTING  
 input = var1  
 IMPORTING  
 OUTPUT = var1 .**New Syntax.**  
For Removing Leading Zeros  
var1 = |{ var1 ALPHA = OUT}|.  
For Adding Leading Zeros  
var1 = |{ var1 ALPHA = IN }|.

Getting Data into Variable Depending on Condition.

**Old Syntax.**  
IF cond1.  
 var1 = fld1.  
ELSE.  
 var1 = fld2.  
ENDIF.**New Syntax.**  
var1 = COND #( WHEN cond1 THEN fld1 ELSE fld2 ).

Case Endcase.

**Old Syntax.**  
CASE var1.   
WHEN cond1.  
 var1 = fld1.  
WHEN cond2.  
 var1 = fld2.  
ENDCASE.**New Syntax.**  
var1 = SWITCH #( var1 WHEN cond1 THEN fld1   
 WHEN cond2 THEN fld2 ).

Creating Table and Filling Data into a Table

**Old Syntax.**  
DATA: itab TYPE TABLE OF structure,  
 wa TYPE structure.wa-fld1 = 1.  
wa-fld2 = 'A'.  
APPEND wa TO itab.  
CLEAR WA.wa-fld1 = 2.  
wa-fld2 = 'B'.  
APPEND wa TO itab.  
CLEAR WA.**New Syntax.**  
DATA(itab) = VALUE structure(( fld1 = 1 fld2 = 'A' )  
 ( fld1 = 2 fld2 = 'B' )).

Adding New entries to Internal Table already having entries.

**Old Syntax.**DATA: itab TYPE TABLE OF structure,  
 wa TYPE structure.wa-fld1 = 1.  
wa-fld2 = 'A'.  
APPEND wa TO itab.  
CLEAR WA.wa-fld1 = 2.  
wa-fld2 = 'B'.  
APPEND wa TO itab.  
CLEAR WA.**New Syntax.**itab = VALUE #( BASE itab  
 ( fld1 = 1 fld2 = 'A' )   
 ( fld1 = 2 fld2 = 'B' ) ).

Concatenate for data type apart from C, N, D, T, or STRING. Before 7.4 use to get Syntax Error saying this, but with new Syntax, it’s possible for any Data Type.

**Old Syntax.**  
CONCATENATE 'The Value in inr is' var1 into fld1.**New Syntax.**  
fld1 = |The Value in inr is { var1 }|.

Moving Data from one Container to another.

**Old Syntax.**  
MOVE-CORRESPONDING itab1 to itab2.**New Syntax.**  
itab2 = CORRESPONDING #( itab1 ).

Using Above Syntax only fields which are identical in the container will be moved to move it to other fields this new syntax can be used.

itab2 = CORRESPONDING #( itab1 MAPPING t1\_fld1 = t2\_fld1   
 t1\_fld2 = t2\_fld2 ).

If some fields need to excluded while moving corresponding fields, this syntax can be used.

itab2 = CORRESPONDING #( itab1 EXCEPT t1\_fld3,t1\_fld4 ).

Moving Data From Table to other by Splitting it into two fields.

\*--Defining Structure  
TYPES: BEGIN OF gty\_s\_rseg\_key,  
 belnr TYPE belnr\_d,  
 gjahr TYPE gjahr,  
 END OF gty\_s\_rseg,  
 gty\_rseg TYPE STANDARD TABLE OF gty\_s\_rseg WITH EMPTY KEY.**Old Syntax.**  
DATA: lt\_awkey TYPE gty\_rseg,  
 lw\_awkey TYPE gty\_s\_rseg.LOOP AT lt\_bkpf into lw\_bkpf.  
 lw\_awkey-belnr = lw\_bkpf-awkey+0(10).  
 lw\_awkey-gjahr = lw\_bkpf-awkey+10(4).   
 APPEND lw\_awkey to lt\_awkey  
ENDLOOP.**New Syntax.**  
DATA(lt\_awkey) = VALUE gty\_rseg( FOR <lfs\_bkpf> IN lt\_bkpf  
 ( belnr = <lfs\_bkpf>-awkey+0(10)  
 gjahr = <lfs\_bkpf>-awkey+10(4) ) ).

Display Message combining two or more Fileds.

**Old Syntax.**  
DATA lv\_name TYPE sy-ucomm.  
DATA lv\_output TYPE char20.  
CONCATENATE 'The User ID is' lv\_name INTO lv\_output SEPARATED BY space.  
MESSAGE lv\_output TYPE 'E' DISPLAY LIKE 'E'.**New Syntax.**  
MESSAGE |The User ID is { lv\_name }| TYPE 'E' DISPLAY LIKE 'E'.

Object-Oriented.

**Old Syntax.**  
DATA: binary\_content TYPE solix\_tab.  
DATA: xl\_content TYPE xstring .CALL METHOD cl\_document\_bcs=>xstring\_to\_solix  
EXPORTING  
ip\_xstring = xl\_content  
receiving  
rt\_solix = binary\_content**New Syntax.**  
binary\_content = cl\_bcs\_convert=>xstring\_to\_solix( ip\_xstring = xl\_content ).

Creating Object For OOABAP.

**Old Syntax.**  
DATA oref TYPE REF TO class.  
CREATE OBJECT oref EXPORTING …**New Syntax.**  
DATA oref TYPE REF TO class.  
oref = NEW #( … ).  
or   
with an inline declaration  
DATA(oref) = NEW class( … )