SET AS [DIRECT]

SET AS [DIRECT] action

Function

The action allows to change object's link to
- local variable of *ALIAS* type,
- item of an structure of *Object* type (reference to an object):
  - for local variables of *Record* type or typed *ALIAS*,
  - for objects of Structured variable type.

Declaration

```
SET dstIdent  AS srcIdent [DIRECT]
```

Parameters

dstIdent	in	Destination of values and links (value identifier of *ALIAS* type (or *Object* type in case of a structure item) row identifier, or whole structure identifier).
srctIdent	in	Source of values and links (value identifier of *ALIAS* type (or *Object* type in case of a structure item) row identifier, or whole structure identifier).

Description

The structure type and the size of *dstIdent* and *srcIdent* must be identical (otherwise the action generates the error *_ERR_RECORD_NO_COMP*). Their type may be:

- RECORD,
- ALIAS,
- Object of *Structured variable* type

The action sets the values of items from `srcIdent_Rec` to the values of items in `dstIdent_Rec`. If the parameter `dstIdent_Rec` is the reference to a row, then the parameter `srcIdent_Rec` must be the reference to a row, too. Accordingly, if `dstIdent_Rec` is the reference to a whole value, `srcIdent_Rec` must be the reference to a whole value, too.

The action rules are similar to the action SET WITH rules.

- The same function for items of structures of a type other than *Object* (reference to an object).
- For an item of *Object* type, it doesn't assign a value, but changes linked object.

Action allows using row index 0 in following case:

SET *dstIdent*[0]^Col1 AS *srcIdent*[0]^Col2 whereby

1. *dstIdent* is local structured variable and *Col1* is of *Object type*
2. *srcIdent* is:

- local structured variable and *Col2* is *of Objekt type*
- type *ALIAS* (*ALIAS* (structDef) *_rA*) and *Col2* is of *Object* type
- object type structured variable; it is possible to use key word DIRECT (if *Col2* is not type reference to object, DIRECT is obligatory)

Action realizes specified command (SET AS) for all rows of the column.
Hence the sizes of both structures (*dstIdent* and *srcIdent*) must be identical.

Example

```
ALIAS _a
INT   _i

SET _a AS U.INT

; the first test
IF U.Int = _a THEN
ENDIF

; the second test
_i := U.Int
IF U.Int = _i THEN
ENDIF
```

The first test for the value equality will be always true, because the local variable *_a* of *ALIAS* type is "pointed" to the object *U.Int*.

The result of the second test depends on it, whether the object *U.Int* has not changed its value since the moment when the value was assigned to *_i* and the test executed.

Value change:

```
ALIAS _a

SET _a AS U.INT
_a := 1
```

Assignment is interpreted by the script as the assigning the value of 1 to *U.Int* !!!.

Facilities of a local variable of *ALIAS* type are identical with items of *Object* type

- objects of *Structured variable* type,
- local variables of *RECORD* type (without the modifier NOALIAS).

For example:

```
ALIAS _a

SET _a AS U.Int
_a := 1  ; the first assignment

RECORD (SD.RecordDef) _lArr1
SET _lArr1[1]^Object AS U.Int
_lArr1[1]^Object := 1  ; the second assignment
```

The first and also the second assignment has the same effect, in regard of the object *U.Int*.

Change of pointing an *Object* type item of a Structured variable type object:

```
SET  SV.Structure[1]^Object AS U.Int
```

The action will point the item *SV.Structurea[1]^Object* to the object *U.Int*. From this moment, the value of the item will be identical with the value of the object *U.Int*. The assignment:

```
SV.Structure[1]^Object := 1
```

It assigns the value of 1 to the object *U.Int* (analogous to the previous examples).

```
RECORD (SD.RecordDef) _lArr1

REDIM _lArr1[2]
SET _lArr1[1]^Object AS U.Int             ; assignment 1
SET _lArr1[2]^Object AS U.Int2            ; assignment 2
SET _lArr1[1]^Object AS _lArr1[2]^Object  ; assignment 3
```

The meaning of the assignments of 1 and 2 is listed above. The meaning of the assignment 3 in the given context identical with the action **SET _lArr1[1]^Object AS U.Int2**. Or:

```
SET SV.Structure[1]^Object AS U.Int             ; assignment 1
WAIT
SET SV.Structure[2]^Object AS U.Int2            ; assignment 2
WAIT
SET SV.Structure[1]^Object AS SV.Structure[2]^Object  ; assignment 3
WAIT
```

Analogous to the previous example, the last assignment is identical with the assignment **SET SV. Structure[1]^Object AS U.Int2**. As you can see from the example, it is copying the reference(s) declared on the right side (the identifier *srcIdent*).

Such a definition do not allow to point (make a reference) a structure item to another structure item. Therefore, there has been introduced the modifier **DIRECT** that makes it possible. The action **SET AS DIRECT** reduces possible types of parameters to these types:

`dstIdent` - *Object* type item of a *RECORD* type local variable, of a local variable "**typed alias**", or of a *S tructured variable* type object.
`srcIdent` - item of a *Structured variable* type object.

For example:

```
RECORD (SD.RecordDef) _lArr1

REDIM _lArr1[2]
SET _lArr1[1]^Object AS U.Int DIRECT              ;assignment 1
SET  _lArr1[2]^Object AS U.Int2 DIRECT             ;assignment 2
SET _lArr1[1]^Object AS _lArr1[2]^Object DIRECT    ;assignment 3
SET  _lArr1[1]^Object AS SV.Structure[2]^Int DIRECT ;assignment 4
```

The assignments 1, 2, 3 are not admissible. The assignment 4 points to the item of the *Structured variable* type object.

If HOBJ of structured variable only is known, the setting of the reference on its field can be done as follows:

SET _locRec[2]^Object AS (HOBJ, ROW, COL) DIRECT

For example:

```
 ; used objects
  ; SD.Active: Structure definition with one column of Object type. Column
name is "Object".
  ; SV.Active: two line structured variable of SD.Active type
 RECORD (SD.Active) _b
 BOOL _bOpenOk


 ; creating of the reference from the field SV.Active[1]^Object to the
field in row 1, column 1 of object with a unique identifier 2131 (HOBJ)
 ; if object 2131 is not structured variable or row index/column index is
outside the range, the action will not be done
 SET SV.Active[1]^Object AS (2131, 10, 10) DIRECT
  ; unsuccessful process of previous action will generate Run-Time error
 WAIT


  ; creating of the reference from the field _b[1]^Object to the field in
row 2, column 1 of the object with unique identifier 24623 (HOBJ)
  ; unlike the previous example, the object 24623 must be opened
 _bOpenOk := %OpenRefToObject(24623, @TRUE)
 IF _bOpenOk THEN
 SET _b[1]^Object AS (24623, 2, 1) DIRECT
 ENDIF
```

!!! The action **SET SV.Structure[1]^Object AS U.Int** (analogous to the assignment of value to object) is processed by means of the process D2000 Server. It takes an effect later, in dependence on the system load. In the script, using WAIT action allows to wait for its execution!!!

The action allows to change the value of ALIAS type.

```
ALIAS (SD._System_Redundancy) _aT
 ALIAS (SD._System_Redundancy) _aT1
 SET _aT AS SV._System_Redundancy
 SET _aT1 AS _aT
```

The revision of admissibility of assignment in regard of structure will be made when the ESL script is saved.
The value of ALIAS type can be changed without revision in following way:

```
BOOL _bOk
 ALIAS (SD._System_Redundancy) _aT
 INT _hbj

 ; example 1
 _hbj := SV._System_Redundancy\HBJ
 SET _aT AS (_hbj)

 ; example 2
 _hbj := %StrToHBJ("SV._System_Redundancy")
 _bOk := %OpenRefToObject(_hbj, @TRUE)
 SET _aT AS (_hbj)

 ; or example 3
 ALIAS _a
 _hbj := %StrToHBJ("SV._System_Redundancy")
 _bOk := %OpenRefToObject(_hbj, @TRUE)
 SET _a AS (_hbj)
 SET _aT AS _a
```

The revision of assignment will be made when the action is executed and ESL script (or object containing the ESL script) must contain the reference to assigned structured variable or this reference was created by the script calling of the function %OpenRefToObject.

See also:

- Script local variables
- Permissible combinations of parameters
- SET WITH action

Related pages:

Script actions