

X-Arrays

Note:

X-arrays are only available under Windows and UNIX.

This section covers the following topics:

- Definition
- Storage Management of X-Arrays
- Storage Management of X-Group Arrays
- Referencing an X-Array
- Parameter Transfer with X-Arrays
- Dynamic X-Arrays
- System Variables *LBOUND and *UBOUND

At application development time you probably will not know the exact number of occurrences of an array. However, you may want be able to change the size of an array at runtime.

For this purpose you can define what is called an X-array (eXtensible array). An X-array can be resized at runtime and can help you manage memory more efficiently. For example, you can use a large number of array occurrences for a short time and then reduce memory when the application is no longer using the array.

Definition

An X-array is an array of which the number of occurrences is not known at compile time. An X-array can only be defined in a DEFINE DATA statement by specifying an asterisk (*) for at least one bound of at least one dimension of the array. The asterisk (*) in the bound definition indicates that the corresponding bound is extensible. Only one bound-- either upper or lower--may be extensible, but not both.

An X-array can be defined whenever a (fixed) array can be defined, i.e. at any level or even as an indexed group. A multidimensional array may have a mixture of constant and extensible bounds.

Example

```

DEFINE DATA LOCAL
1 #X-ARR1 (A5/1:*)           /* lower bound is always 1, upper bound is extensible
1 #X-ARR2 (A5/*)           /* same as #X-ARR1 (A5/1:*)
1 #X-ARR3 (A5/*:100)       /* lower bound is extensible, upper bound is always 100
1 #X-ARR4 (A5/1:10,1:*)    /* upper and lower bounds of 1st dimension are constant
                             /* (10 occurrences)
END-DEFINE                  /* lower bound of 2nd dimension is 1, upper bound is extensible

```

Storage Management of X-Arrays

The EXPAND, RESIZE and REDUCE statements may change the number of occurrences by specifying lower and upper bounds for each dimension. However, the number of dimensions of the X-array (1, 2 or 3 dimensions) cannot be changed.

Example:

```

DEFINE DATA LOCAL
1 #X-ARR(I4/10:*)
END-DEFINE

EXPAND ARRAY #X-ARR TO (10:10000)

/* #X-ARR(10) to #X-ARR(10000) are accessible

WRITE *LBOUND(#X-ARR)           /* is 10
   *UBOUND(#X-ARR)             /* is 10000
   *OCCURRENCE(#X-ARR)        /* is 9991
#X-ARR1(*) := 4711              /* same as #X-ARR(10:10000) := 4711

/* resize array from current lower bound=10 to upper bound =1000

RESIZE ARRAY #X-ARR TO (*:1000)

/* #X-ARR(10) to #X-ARR(1000) are accessible
/* #X-ARR(1001) to #X-ARR(10000) are released

WRITE *LBOUND(#X-ARR)           /* is 10
   *UBOUND(#X-ARR)             /* is 1000
   *OCCURRENCE(#X-ARR)        /* is 991

/* release all occurrences

REDUCE ARRAY #X-ARR TO 0
WRITE *OCCURRENCE(#X-ARR)      /* is 0

```

Storage Management of X-Group Arrays

If you want to increase or decrease occurrences of X-group arrays, you must distinguish between independent and dependent dimensions.

A dimension which is specified directly (not inherited) for an X-(group) array is **independent**.

A dimension which is **not** specified directly (inherited) for an X-(group) array is **dependent**.

Only the upper or lower bounds of independent dimensions can be changed in the EXPAND, RESIZE and REDUCE statements. The upper and lower bounds of dependent dimensions **cannot** be changed directly. The upper and lower bounds of dependent dimensions must be changed by using the corresponding X-group array name.

Example - Independent/Dependent Dimensions:

```

DEFINE DATA LOCAL
1 #X-GROUP-ARR1 (10:*)          /* (10:*)
  2 #X-ARR1 (I4)                /* (10:*)
  2 #X-ARR2 (I4/20:*)          /* (10:*, 20:*)
  2 #X-GROUP-ARR2              /* (10:*)
  3 #X-ARR3 (I4)                /* (10:*)
  3 #X-ARR4 (I4/30:*)          /* (10:*, 30:*)
  3 #X-ARR5 (I4/40:*, 50:*)    /* (10:*, 40:*, 50:*)
END-DEFINE

```

The following table shows whether the dimensions in the above program are independent or dependent.

Name	Dependent Dimension	Independent Dimension
#X-GROUP-ARR1		(10:*)
#X-ARR1	(10:*)	
#X-ARR2	(10:*)	(20:*)
#X-GROUP-ARR2	(10:*)	
#X-ARR3	(10:*)	
#X-ARR4	(10:*)	(30:*)
#X-ARR5	(10:*)	(40:*,50:*)

Only the dimension notation asterisk (*) for the upper and lower bound is allowed for dependent dimensions. The asterisk is used to indicate that the bounds of the dependent dimension must be kept as they are and cannot be changed.

The occurrences of the dependent dimensions can only be changed by manipulating the corresponding array groups.

```

EXPAND ARRAY #X-GROUP-ARR1 TO (10:1000)
/* #X-ARR1(10) to #X-ARR1(1000) are allocated
/* #X-ARR3(10) to #X-ARR3(1000) are allocated

EXPAND ARRAY #X-ARR2 TO (*:*, 20:2000)
/* #X-ARR2(10:1000, 20:2000) are allocated

EXPAND ARRAY #X-ARR4 TO (*:*, 30:3000)
/* #X-ARR4(10:1000, 30:3000) are allocated

EXPAND ARRAY #X-ARR5 TO (*:*, 40:4000, 50:5000)
/* #X-ARR5(10:1000, 40:4000, 50:5000) are allocated

END

```

The EXPAND statements may be coded in an arbitrary order.

The following use of the EXPAND statement is not allowed, since the arrays only have dependent dimensions.

```
EXPAND ARRAY #X-ARR1 TO ...  
EXPAND ARRAY #X-GROUP-ARR2 TO ...  
EXPAND ARRAY #X-ARR3 TO ...
```

Referencing an X-Array

The occurrences of an X-array must be allocated by an EXPAND or RESIZE statement before they can be accessed. Following statements allocate occurrences implicitly:

- PARSE assigns XML data to an X-Array
- READ WORK FILE reads work file data into an X-Array
- READ, FIND and GET fill an X-Array with values from Tamino

The asterisk (*) notation in an array reference stands for the complete range of a dimension. If the array is an X-array, * is the index range of the current allocated lower and upper bound values which are determined by *LBOUND and *UBOUND.

Parameter Transfer with X-Arrays

X-arrays that are used as parameters are treated like constant arrays with regard to the verification of the following:

- format,
- length,
- dimension
or
- number of occurrences.

In addition, X-array parameters can also change the number of occurrences by using the statements RESIZE, as well as REDUCE and EXPAND. The RESIZE of an X-array parameter is dependent on three factors:

- the type of parameter transfer used, i.e. By Reference or By Value,
- the definition of the caller or parameter X-array
and
- the type of X-array range being passed on (complete range or subrange).

The following tables demonstrate when a RESIZE, REDUCE or EXPAND of an X-array parameter is allowed.

Example with CALL By Value

CALLER	PARAMETER		
	<i>Static</i>	<i>Variable (I:V)</i>	<i>X-Array</i>
<i>Static</i>	NO	NO	YES
<i>X-Array subrange</i> e.g. CALLNAT...#XA(1:5)	NO	NO	YES
<i>X-Array complete range</i> e.g. CALLNAT...#XA(*)	NO	NO	YES

CALL By Reference/CALL By Value Result

	PARAMETER			
CALLER	<i>Static</i>	<i>Variable (I:V)</i>	<i>X-Array with a fixed lower bound e.g. DEFINE DATA PARAMETER 1 #PX (A10/1:*)</i>	<i>X-Array with a fixed upper bound e.g. DEFINE DATA PARAMETER 1 #PX (A10/*:1)</i>
<i>Static</i>	NO	NO	NO	NO
<i>X-Array subrange e.g. CALLNAT...#XA(1:5)</i>	NO	NO	NO	NO
<i>X-Array with a fixed lower bound, complete range e.g. DEFINE DATA LOCAL 1 #XA(A10/1:*) ... CALLNAT...#XA(*)</i>	NO	NO	YES	NO
<i>X-Array with a fixed upper bound, complete range e.g. DEFINE DATA LOCAL 1 #XA(A10/*:1) ... CALLNAT...#XA(*)</i>	NO	NO	NO	YES

Dynamic X-Arrays

A dynamic X-array may be allocated by first specifying the number of occurrences using the EXPAND statement and then assigning expanding the length of the previously allocated array occurrences.

Example:

```
DEFINE DATA LOCAL
  1 #X-ARRAY(A/1:*)  DYNAMIC
END-DEFINE
EXPAND ARRAY  #X-ARRAY TO (1:10)
  /* allocate #X-ARRAY(1) to #X-ARRAY(10) with zero length.
  /* *LENGTH(#X-ARRAY(1:10)) is zero

#X-ARRAY(*) := 'abc'
  /* #X-ARRAY(1:10) contains 'abc',
  /* *LENGTH(#X-ARRAY(1:10)) is 3

EXPAND ARRAY  #X-ARRAY TO (1:20)
  /* allocate #X-ARRAY(11) to #X-ARRAY(20) with zero length
  /* *LENGTH(#X-ARRAY(11:20)) is zero

#X-ARRAY(11:20) := 'def'
  /* #X-ARRAY(11:20) contains 'def'
  /* *LENGTH(#X-ARRAY(11:20)) is 3
```

System Variables *LBOUND and *UBOUND

The system variables *LBOUND and *UBOUND contain the current lower and upper bound of an array for the specified dimension(s) (1,2 or 3)

If no occurrences of an X-array have been allocated, the access to *LBOUND or *UBOUND is undefined and leads to a runtime error. In order to avoid a runtime error, *OCCURRENCE may be used to check against zero occurrences before *LBOUND or *UBOUND is evaluated:

Example:

```
IF *OCCURRENCE (#A) NE 0 AND *UBOUND(#A) < 100 THEN ...
```