

Array Dimension Definition

The *array-dimension-definition* is used in the following statements: `DEFINE DATA OBJECT` and in the *variable-definition* option of `DEFINE DATA LOCAL`, `DEFINE DATA INDEPENDENT`, `DEFINE DATA CONTEXT`, `DEFINE DATA OBJECT`.

The *array-dimension-definition* has the following syntax:

<code>{bound[:bound]},... 3</code>

This chapter covers the following topics:

- Function
- Syntax Description

Function

With an *array-dimension-definition*, you define the lower and upper bound of a dimension in an array-definition.

You can define up to 3 dimensions for an array.

See also *Arrays* in the *Programming Guide*.

Syntax Description

<i>bound</i>	<p>A bound can be one of the following:</p> <ul style="list-style-type: none"> ● a numeric integer constant; ● a previously defined named constant; ● (for database arrays) a previously defined user-defined variable; or ● an asterisk (*) defines an extensible bound, otherwise known as an X-array. <p>If only one bound is specified, the value represents the upper bound and the lower bound is assumed to be 1.</p>
---------------------	--

X-Arrays

If at least one bound in at least one dimension of an array is specified as extensible, that array is then called an X-array (eXtensible array). Only one bound (either upper or lower) may be extensible in any one dimension, but not both. Multi-dimensional arrays may have a mixture of constant and extensible bounds, e.g. `#a(1:100, 1:*)`.

Example:

```
DEFINE DATA LOCAL
1 #ARRAY1(I4/1:10)
1 #ARRAY2(I4/10)
1 #X-ARRAY3(I4/1:*)
1 #X-ARRAY4(I4/*,1:5)
1 #X-ARRAY5(I4/*:10)
1 #X-ARRAY6(I4/1:10,100:* ,*:1000)
END-DEFINE
```

In the following table you can see the bounds of the arrays in the above program more clearly.

	Dimension1		Dimension2		Dimension3	
	Lower bound	Upper bound	Lower bound	Upper bound	Lower bound	Upper bound
#ARRAY1	1	10	-	-	-	-
#ARRAY2	1	10	-	-	-	-
#X-ARRAY3	1	eXtensible	-	-	-	-
#X-ARRAY4	1	eXtensible	1	5	-	-
#X-ARRAY5	eXtensible	10	-	-	-	-
#X-ARRAY6	1	10	100	eXtensible	eXtensible	1000

Examples of array definitions:

```
#ARRAY2(I4/10) /* a one-dimensional array with 10 occurrences (1:10)
#X-ARRAY4(I4/*,1:5) /* a two-dimensional array
#X-ARRAY6(I4/1:10,100:* ,*:1000) /* a three-dimensional array
```

Variable Arrays in a Parameter Data Area

In a parameter data area, you may specify an array with a variable number of occurrences. This is done with the index notation `1:V`.

Example 1: #ARR01 (A5/1:V)

Example 2: #ARR02 (I2/1:V,1:V)

A parameter array which contains a variable index notation `1:V` can only be redefined in the length of

- its elementary field length, if the `1:V` index is right-most; for example:

#ARR(A6/1:V) can be redefined up to a length of 6 bytes

#ARR(A6/1:2,1:V) can be redefined up to a length of 6 bytes

#ARR(A6/1:2,1:3,1:V) can be redefined up to a length of 6 bytes

- the product of the right-most fixed occurrences and the elementary field length; for example:

#ARR(A6/1:V,1:2) can be redefined up to a length of $2*6 = 12$ bytes

#ARR(A6/1:V,1:3,1:2) can be redefined up to a length of $3*2*6 = 36$ bytes

#ARR(A6/1:2,1:V,1:3) can be redefined up to a length of $3*6 = 18$ bytes

A variable index notation $1:V$ cannot be used within a redefinition.

Example:

```
DEFINE DATA PARAMETER
  1 #ARR(A6/1:V)
  1 REDEFINE #ARR
    2 #R-ARR(A1/1:V) /* (1:V) is not allowed in a REDEFINE block
END-DEFINE
```

As the number of occurrences is not known at compilation time, it must not be referenced with the index notation (*) in the statements INPUT, WRITE, READ WORK FILE, WRITE WORK FILE. Index notation (*) may be applied either to all dimensions or to none.

Valid examples:

```
#ARR01 (*)
#ARR02 (*,*)
#ARR01 (1)
#ARR02 (5,#FIELDX)
#ARR02 (1,1:3)
```

Invalid example:

```
#ARRAYY (1,*) /* not allowed
```

To avoid runtime errors, the maximum number of occurrences of such an array should be passed to the subprogram/subroutine via another parameter. Alternatively, you may use the system variable *OCCURRENCE.

Notes:

1. If a parameter data area that contains an index $1:V$ is used as a local data area (that is, specified in a DEFINE DATA LOCAL statement), a variable named V must have been defined as CONSTANT.
2. In a dialog, an index $1:V$ cannot be used in conjunction with BY VALUE.