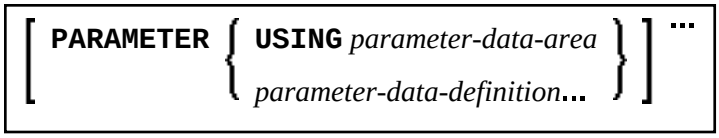
Definition von Parameter Data

Allgemeine Syntax von `DEFINE DATA PARAMETER`:



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Einschränkungen
- Syntax-Beschreibung

Eine Erläuterung der in dem Syntax-Diagramm verwendeten Symbole entnehmen Sie dem Abschnitt *Syntax-Symbole*.

Funktion

Das `DEFINE DATA PARAMETER`-Statement wird benutzt, um die Datenelemente zu definieren, die als Eingabeparameter in einem Natural-Subprogramm, einer externen Subroutine oder Helproutine verwendet werden sollen. Diese Parameter können innerhalb des Statements selbst definiert werden (siehe *Parameter-Daten-Definition* weiter unten); oder sie können außerhalb des Programms in einer Parameter Data Area (PDA) definiert werden, wobei das Statement diese Data Area referenziert.

Einschränkungen

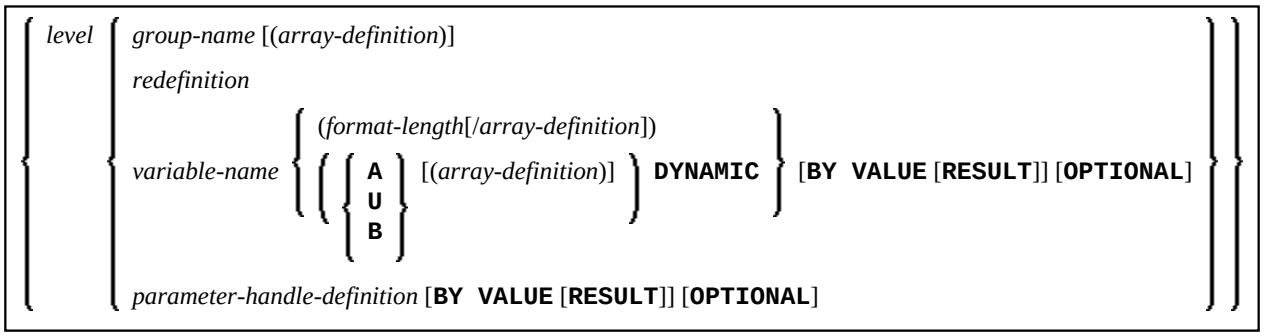
- Parameter-Datenelementen dürfen keine Ausgangswerte oder auch Konstanten-Werte zugewiesen werden, und sie dürfen keine Editiermasken-Definitionen (EM), Kopfzeilen-Definitionen (HD) oder Druckmodus-Definitionen (PM) haben (siehe auch *EM-*, *HD-*, *PM-Parameter für Feld/Variable*).
- Die Parameter Data Area und die sie referenzierenden Objekte müssen in derselben Library (oder in einer Steplib) enthalten sein.

Syntax-Beschreibung

USING ***parameter-data-area***	Der Name der *parameter-data-area*, die Datenelemente enthält, die als Parameter in einem Subprogramm, einer externen Subroutine oder einem Dialog benutzt werden.
parameter-data-definition	Anstatt eine Parameter Data Area zu definieren, können Parameter auch direkt in einem Programm oder einer Routine definiert werden; siehe *Definition von Parameterdaten* weiter unten.
END-DEFINE	Das für Natural reservierte Wort `END-DEFINE` muss zum Beenden des `DEFINE DATA`-Statements benutzt werden.

Definition von Parameterdaten

Für die direkte Parameter-Daten-Definition gilt die folgende Syntax:



Syntax-Element-Beschreibung:

level	Dies ist eine ein- oder zweistellige Zahl im Bereich von 01 bis 99 (die vorangestellte 0 ist nicht erforderlich), die in Verbindung mit der Gruppierung von Feldern verwendet wird. Felder mit einer Level-Nummer von 02 an aufwärts werden als Teil einer unmittelbar vorangehenden Gruppe mit einer jeweils nächst-niedrigeren Level-Nummer betrachtet. Durch die Definition einer Gruppe (die auch nur aus einem Feld bestehen kann) ist es möglich, durch Angabe lediglich des Gruppennamens eine ganze Reihe von aufeinanderfolgenden Feldern gleichzeitig zu referenzieren. Bei manchen Statements (`CALL`, `CALLNAT`, `RESET`, `WRITE` usw.) können Sie den Gruppennamen als Aufrufnamen angeben, um die in der Gruppe enthaltenen Felder zu referenzieren. Eine Gruppe kann aus weiteren Gruppen bestehen. Bei der Vergabe von Level-Nummern für eine Gruppe darf kein Level ausgelassen werden.
group-name	Der Name einer Gruppe. Der Name muss den Regeln zur Definition eines Natural-Variablennamens entsprechen. Siehe auch die folgenden Abschnitte: • *Namen von Benutzervariablen* in der Dokumentation *Natural benutzen*. • *Datenstrukturen qualifizieren* im *Leitfaden zur Programmierung*.
array-definition	Mit *array-definition* definieren Sie die untere und obere Grenze einer Dimension in einer Array-Definition. Siehe *Definition von Array-Dimensionen* und *Variable Arrays in einer Parameter Data Area*.

redefinition	Mit *redefinition* können Sie eine Gruppe oder ein einzelnes Feld oder eine einzelne Variable (d.h. Skalar oder Array) redefinieren. Siehe *Redefinition*. **Anmerkung:** In einer Parameter Data Area ist eine Redefinition von Gruppen nur innerhalb eines `REDEFINE`-Blocks zulässig.
variable-name	Der der Variablen zuzuweisende Name. Es gelten die Regeln für Natural- Variablennamen. Informationen zu Namenskonventionen für Benutzervariablen, siehe *Namen von Benutzervariablen* in der Dokumentation *Natural benutzen*.
format-length	Das Format und die Länge des Feldes. Informationen zu Format und Länge für Benutzervariablen, siehe den Abschnitt *Format und Länge von Benutzervariablen* im *Leitfaden zur Programmierung*.
A, U or B	Datentyp: Alphanumerisch (A) oder Binär (B) für dynamische Variable.
DYNAMIC	Ein Parameter kann als `DYNAMIC` definiert werden. Weitere Informationen zur Verarbeitung von dynamischen Variablen, siehe *Dynamische Variablen* im *Leitfaden zur Programmierung*.
	Call-Modus: Je nachdem, ob der Call-By-Reference- oder Call-By-Value-Modus benutzt wird, gilt die betreffende Übertragungsart. Weitere Informationen siehe `CALLNAT`- Statement.
(without BY VALUE)	**Call-By-Reference-Modus:** Ohne `BY VALUE` (gilt standardmäßig) wird ein Parameter durch Referenzierung seiner Adresse ("By Reference") an ein Subprogramm bzw. eine Subroutine übergeben; das bedeutet, dass ein in einem `CALLNAT`- bzw. `PERFORM`-Statement als Parameter angegebenes Feld dasselbe Format und dieselbe Länge haben muss wie das betreffende Feld in dem/der aufgerufenen Subprogramm/Subroutine.

<table>
<tr><td>BY VALUE</td><td>Call-By-Value-Modus:

Mit BY VALUE wird ein Parameter direkt als Wert (by value) an ein Subprogramm oder eine Subroutine übergeben; d.h. statt der Adresse des Parameters wird der Wert selbst übergeben. Das bedeutet, das Feld in dem Subprogramm bzw. der Subroutine braucht nicht dasselbe Format und dieselbe Länge zu haben wie der Parameter beimCALLNAT-/PERFORM-Statement. Das Format und die Länge der beiden muss lediglich datenübertragungskompatibel gemäß den Regeln für Datenübertragung/-zuweisung sein (siehe Leitfaden zur Programmierung).

Mit BY VALUE können Sie zum Beispiel die Länge eines Feldes in einem Subprogramm bzw. einer Subroutine vergrößern (falls eine Erweiterung des Subprogramms bzw. der Subroutine dies erforderlich machen sollte), ohne deswegen auch die Objekte, die das Subprogramm bzw. die Subroutine aufrufen, anpassen zu müssen.

Beispiel für BY VALUE:

<table>
<tr><td><pre>* Program
DEFINE DATA LOCAL
1 #FIELDA (P5)
...
END-DEFINE
...
CALLNAT 'SUBR01' #FIELDA
...</pre></td><td><pre>* Subroutine SUBR01
DEFINE DATA PARAMETER
1 #FIELDB (P9) BY VALUE
END-DEFINE
...</pre></td></tr>
</table>
</td></tr>
<tr><td>BY VALUE RESULT</td><td>Call-By-Value-Result-Modus:

Während BY VALUE für die Übergabe eines Parameters an ein Subprogramm oder eine Subroutine gilt, bewirkt die Angabe von BY VALUE RESULT die Übergabe des Parameterwertes in beide Richtungen; d.h. der Parameterwert selbst wird vom aufrufenden Objekt an das Subprogramm bzw. die Subroutine übergeben, und bei der Rückkehr zum aufrufenden Objekt wird der Parameterwert selbst von dem Subprogramm bzw. der Subroutine an das aufrufende Objekt zurückgegeben.

Wenn Sie BY VALUE RESULT verwenden, müssen Format und Länge der betreffenden Felder in beide Richtungen datenübertragungskompatibel sein.</td></tr>
<tr><td>OPTIONAL</td><td>Bei einem ohne OPTIONAL definierten Parameter (Standard) muss ein Wert vom aufrufenden Objekt übergeben werden.

Bei einem mit OPTIONAL definierten Parameter muss ein Wert vom aufrufenden Objekt an diesen Parameter nicht unbedingt übergeben werden. Im aufrufenden Objekt wird die Notation nX benutzt, um Parameter anzugeben, die übersprungen werden, d.h. für die keine Werte übergeben werden.

Bei der SPECIFIED-Option können Sie zur Laufzeit ermitteln, ob ein optionaler Parameter definiert worden ist oder nicht.</td></tr>
</table>

parameter-handle-definition	Siehe Abschnitt *Parameter-Handle-Definition* weiter unten.

Parameter-Handle-Definition

Syntax der *parameter-handle-definition*:

handle-name [(*array-definition*)] **HANDLE OF OBJECT**

Syntax-Element-Beschreibung:

handle-name	Der der Handle zuzuweisende Name; es gelten die Namenskonventionen für Benutzervariablen; siehe *Namen von Benutzervariablen* in der Dokumentation *Natural benutzen*.
HANDLE OF OBJECT	Wird benutzt in Zusammenhang mit NaturalX. Siehe *NaturalX* im *Leitfaden zur Programmierung*.
array-definition	Bei einer Array-Definition definieren Sie die untere und obere Grenze einer Dimension in einer Array-Dimension. Siehe *Definition der Array-Dimension*.