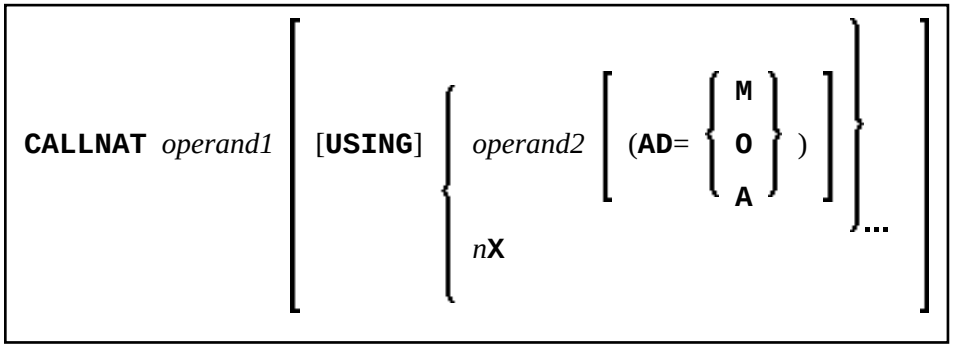
CALLNAT



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Syntax-Beschreibung
- Übertragung von Parametern mit dynamischen Variablen
- Beispiele

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

Verwandte Statements: `CALL` | `CALL FILE` | `CALL LOOP` | `DEFINE SUBROUTINE` | `ESCAPE` | `FETCH` | `PERFORM`

Gehört zur Funktionsgruppe: *Aufrufen von Programmen und Unterprogrammen*

Funktion

Das Statement `CALLNAT` dient dazu, ein Natural-Subprogramm zur Ausführung aufzurufen. Ein Natural-Subprogramm kann nur über ein `CALLNAT`-Statement aufgerufen werden; es kann nicht selbständig ausgeführt werden.

Wenn das `CALLNAT`-Statement ausgeführt wird, wird die Ausführung des aufrufenden Objekts (d.h. des Objekts, das das `CALLNAT`-Statement enthält) unterbrochen und das aufgerufene Subprogramm ausgeführt. Die Ausführung des Subprogramms dauert an, bis entweder sein `END`-Statement erreicht ist oder die Verarbeitung des Subprogramms durch die Ausführung eines `ESCAPE ROUTINE`-Statements gestoppt wird. In beiden Fällen wird dann die Verarbeitung des aufrufenden Objekts mit dem nächsten Statement nach dem `CALLNAT`-Statement fortgesetzt.

Anmerkungen:

1. Ein Subprogramm kann wiederum andere Subprogramme aufrufen.
2. Ein Subprogramm hat keinen Zugriff auf die von dem aufrufenden Objekt benutzte Global Data Area. Wenn ein Subprogramm wiederum eine Subroutine oder Helproutine aufruft, kann es seine eigene Global Data Area erstellen, und diese zusammen mit der Subroutine/Helproutine gemeinsam benutzen.

Syntax-Beschreibung

Operanden-Definitionstabelle:

Operand	Mögliche Struktur					Mögliche Formate												Referenzierung erlaubt	Dynam. Definition
operand1	C	S				A												ja	nein
operand2	C	S	A	G		A	U	N	P	I	F	B	D	T	L	C	G	O	ja	ja

Syntax-Element-Beschreibung:

operand1	**Subprogramm-Name:** Als *operand1* geben Sie den Namen des Subprogramms an, das aufgerufen werden soll. Dieser kann entweder als 1 bis 8 Zeichen lange Konstante angegeben werden oder — falls je nach Programmlogik unterschiedliche Subprogramme aufgerufen werden sollen — als alphanumerische Variable mit Länge 1 bis 8. Der Name des Subprogramms darf ein Und-Zeichen (&) enthalten; zur Ausführungszeit wird dieses Zeichen durch den aus einem Zeichen bestehenden Code ersetzt, der dem aktuellen Wert der Systemvariablen `*LANGUAGE` entspricht. Dadurch ist es beispielsweise möglich, je nachdem in welcher Sprache eine Eingabe gemacht wird, zur Verarbeitung der Eingabe unterschiedliche Subprogramme aufzurufen.

operand2	**Parameter:** Werden Parameter an das Subprogramm übergeben, muss die Struktur der Parameterliste in einem DEFINE DATA PARAMETER-Statement definiert werden. Die mit dem CALLNAT-Statement angegebenen Parameter sind die einzigen Daten, die dem Subprogramm vom aufrufenden Objekt zur Verfügung stehen. Standardmäßig erfolgt die Übergabe der Parameter durch Referenzierung ("By Reference"), d.h. die Daten werden über Adress-Parameter übergeben, die Parameterwerte selbst werden nicht übertragen. Es besteht aber auch die Möglichkeit, die Parameterwerte selbst zu übergeben. Hierzu definieren Sie die betreffenden Felder im DEFINE DATA PARAMETER-Statement des Subprogramms mit der Option BY VALUE bzw. BY VALUE RESULT (siehe *parameter-data-definition* in der Beschreibung des DEFINE DATA-Statements). ● Für die Parameterübergabe durch Referenzierung ("By Reference") gilt: Reihenfolge, Format und Länge der Parameter im aufrufenden Objekt müssen genau den Angaben im DEFINE DATA PARAMETER- Statement des Subprogramms entsprechen. Die Namen der Variablen im aufrufenden Objekt und im aufgerufenen Subprogramm können unterschiedlich sein. ● Für die Übergabe der Parameterwerte selbst ("By Value") gilt: die Reihenfolge der Parameter im aufrufenden Objekt muss der Reihenfolge im DEFINE DATA PARAMETER-Statement des Subprogramms entsprechen. Formate und Längen der Variablen im aufrufenden Objekt und im Subprogramm können unterschiedlich sein, müssen aber datenübertragungskompatibel sein (vgl. entsprechende Tabelle im Abschnitt *Regeln für arithmetische Operationen, Datenübertragung* im *Leitfaden zur Programmierung*. Die Namen der Variablen im aufrufenden Objekt und im aufgerufenen Subprogramm können unterschiedlich sein. Um Parameterwerte, die im Subprogramm verändert wurden, an das aufrufende Objekt zurückgeben zu können, müssen Sie die betreffenden Felder mit BY VALUE RESULT definieren. Mit BY VALUE (ohne RESULT) ist es nicht möglich, veränderte Parameterwerte an das aufrufende Objekt zurückzugeben (unabhängig von der AD-Parameter-Angabe; vgl. unten). **Anmerkung:** Intern wird bei BY VALUE eine Kopie der Parameterwerte erzeugt. Das Subprogramm greift auf diese Kopie zu und kann sie modifizieren, was aber keinen Einfluss auf die Originalparameterwerte im aufrufenden Objekt hat. Bei BY VALUE RESULT wird ebenfalls eine Kopie erzeugt, aber nach Beendigung des Subprogramms überschreiben die (modifizierten) Werte der Kopie die Originalparameterwerte. Für beide Arten der Parameterübergabe sind folgende Punkte zu beachten: ● Wenn als *operand2* eine Gruppe angegeben wird, werden die einzelnen in der Gruppe enthaltenen Felder an das Subprogramm übergeben; d.h. für jedes dieser Felder muss in der Parameter Data Area des Subprogramms ein entsprechendes Feld definiert werden. ● Eine Gruppe darf in der Parameter Data Area eines Subprogramms nur innerhalb eines REDEFINE-Blocks redefiniert werden. ● Bei der Übergabe eines Arrays muss die Anzahl seiner Dimensionen und Ausprägungen in der Parameter Data Area des Subprogramms denen in der CALLNAT-Parameterliste entsprechen. **Anmerkung:** Wenn mehrere Ausprägungen eines Arrays, das als Teil einer indizierten Gruppe definiert ist, mit dem CALLNAT-Statement übergeben werden, dürfen die entsprechenden Felder in der Parameter Data Area des Subprogramms nicht redefiniert werden, da sonst die falschen Adressen übergeben werden. Wenn die Option PCHECK des Systemkommandos COMPOPT auf ON gesetzt ist, überprüft der Compiler Anzahl, Format, Länge und Array-Indexgrenzen der Parameter, die in einem CALLNAT-Statement angegeben sind. Die Funktion OPTIONAL des DEFINE DATA PARAMETER-Statements wird bei der Parameter-Prüfung mit berücksichtigt.

AD=	**Attribut-Definition:** Wenn *operand2* eine Variable ist, können Sie sie wie folgt kennzeichnen:	
	AD=O	Nicht modifizierbar, siehe Session-Parameter AD=O. **Anmerkung:** Intern wird AD=O genauso verarbeitet wie BY VALUE (siehe *parameter-data-definition* in der Beschreibung des DEFINE DATA-Statements).
	AD=M	Modifizierbar, siehe Session-Parameter AD=M. Dies ist die Standardeinstellung.
	AD=A	Nur für Eingabe, siehe Session-Parameter AD=A.
	Wenn *operand2* eine Konstante ist, kann der Session-Parameter AD nicht explizit angegeben werden. Für Konstanten gilt immer AD=O.	
***n*X**	Mit der Notation *n*X können Sie angeben, dass die nächsten *n* Parameter übersprungen werden sollen (z.B. 1X, um den nächsten Parameter zu überspringen, oder 3X, um die nächsten drei Parameter zu überspringen); dies bedeutet, dass für die nächsten *n* Parameter keine Werte an das Subprogramm übergeben werden. Der mögliche Wertebereich für *n* ist 1 – 4096. Ein zu überspringender Parameter muss mit dem Schlüsselwort OPTIONAL im DEFINE DATA PARAMETER-Statement des Subprogramms definiert werden. OPTIONAL bedeutet, dass ein Wert vom aufrufenden Objekt an solch einen Parameter übergeben werden kann, aber nicht unbedingt übergeben werden muss.	

Übertragung von Parametern mit dynamischen Variablen

Dynamische Variablen können als Parameter an ein aufgerufenes Programmobjekt (CALLNAT, PERFORM) übergeben werden.

Eine Übergabe durch Referenzierung ("Call By Reference") ist möglich, weil dynamische Variablen einen zusammenhängenden Wertebereich darstellen.

Bei einer Übergabe der Parameterwerte selbst ("Call By Value") wird die Variablen-Definition des Aufrufenden als Ausgangsoperand und die Parameter-Definition als Zieloperand zugewiesen. Ein "Call By Value Result "bewirkt des weiteren eine Umkehrung der Zuordnung. Bei "Call By Reference" müssen beide Definitionen DYNAMIC sein. Wenn nur eine von ihnen DYNAMIC ist, wird ein Laufzeitfehler hervorgerufen. Bei "Call By Value (Result)" sind alle Kombinationen möglich.

Die folgende Tabelle veranschaulicht die gültigen Kombinationen statisch und dynamisch definierter Variablen des Aufrufenden und statisch und dynamisch definierter Parameter hinsichtlich der Übertragung von Parametern.

Call By Reference

***operand2* vom Aufrufenden**	**Parameter-Definition**	
	Statisch	**Dynamisch**
Statisch	ja	nein
Dynamisch	nein	ja

Die Formate der dynamischen Variablen A oder B müssen miteinander übereinstimmen.

Call by Value (Result)

operand2 vom Aufrufenden	Parameter-Definition	
	Statisch	**Dynamisch**
Statisch	ja	ja
Dynamisch	ja	ja

Anmerkung:
Bei statischen/dynamischen oder dynamischen/statischen Definitionen kann es vorkommen, dass ein Wert nach den Datenübertragungsregeln der entsprechenden Zuweisungen abgeschnitten wird.

Beispiele

- Beispiel 1
- Beispiel 2

Beispiel 1

Aufrufendes Programm:

```
** Example 'CNTEX1': CALLNAT
************************************************************************
DEFINE DATA LOCAL
1 #FIELD1 (N6)
1 #FIELD2 (A20)
1 #FIELD3 (A10)
END-DEFINE
*
CALLNAT 'CNTEX1N' #FIELD1 (AD=M) #FIELD2 (AD=O) #FIELD3 'P4 TEXT'
*
WRITE '=' #FIELD1 '=' #FIELD2 '=' #FIELD3
*
END
```

Aufgerufenes Subprogramm `CNTEX1N`:

```
** Example 'CNTEX1N': CALLNAT  (called by CNTEX1)
************************************************************************
DEFINE DATA PARAMETER
1 #FIELDA (N6)
1 #FIELDB (A20)
1 #FIELDC (A10)
1 #FIELDD (A7)
END-DEFINE
*
*
#FIELDA := 4711
*
#FIELDB := 'HALLO'
*
#FIELDC := 'ABC'
```

```
*
WRITE '=' #FIELDA '=' #FIELDB '=' #FIELDC '=' #FIELDD
*
END
```

Beispiel 2

Aufrufendes Programm:

```
** Example 'CNTEX2': CALLNAT
************************************************************************
DEFINE DATA LOCAL
1 #ARRAY1  (N4/1:10,1:10)
1 #NUM     (N2)
END-DEFINE
*
*
CALLNAT 'CNTEX2N' #ARRAY1 (2:5,*)
*
FOR #NUM 1 TO 10
  WRITE #NUM #ARRAY1(#NUM,1:10)
END-FOR
*
END
```

Aufgerufenes Subprogramm CNTEX2N:

```
** Example 'CNTEX2N': CALLNAT  (called by CNTEX2)
************************************************************************
DEFINE DATA
PARAMETER
1 #ARRAY (N4/1:4,1:10)
LOCAL
1 I      (I2)
END-DEFINE
*
*
FOR I 1 10
  #ARRAY(1,I) := I
  #ARRAY(2,I) := 100 + I
  #ARRAY(3,I) := 200 + I
  #ARRAY(4,I) := 300 + I
END-FOR
*
END
```