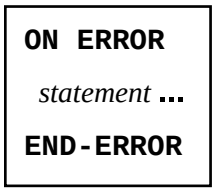
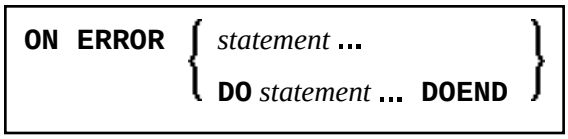
ON ERROR

Structured Mode-Syntax



Reporting Mode-Syntax



Dieses Kapitel behandelt folgende Themen:

- Funktion
- Einschränkung
- Syntax-Beschreibung
- ON ERROR-Verarbeitung in Unterprogrammen
- Systemvariablen *ERROR-NR und *ERROR-LINE
- Beispiel

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

Verwandte Statements: `DECIDE FOR` | `DECIDE ON` | `IF` | `IF SELECTION`

Funktion

Das Statement `ON ERROR` dient dazu, Laufzeitfehler abzufangen, welche andernfalls eine Natural-Fehlermeldung, den Abbruch des gerade ausgeführten Programms und die Rückkehr zum Kommandoeingabe-Modus zur Folge hätten.

Sobald die Ausführung der in einem `ON ERROR`-Statement-Block angegebenen Statements beginnt, ist der normale Programmablauf unterbrochen und kann nicht fortgesetzt werden. Eine Ausnahme bildet Fehler `3145` (angeforderter Datensatz im Hold), bei dem über ein `RETRY`-Statement die Verarbeitung genau an der Stelle, an der sie unterbrochen wurde, fortgesetzt werden kann.

Dieses Statement ist nicht prozedural (das heißt, seine Ausführung hängt von einem Ereignis ab, nicht davon, wo im Programm es steht).

Einschränkung

Jedes Natural-Objekt darf maximal ein ON ERROR-Statement enthalten.

Syntax-Beschreibung

statement...	**Festlegung der ON ERROR-Verarbeitung:** Um die Verarbeitung zu definieren, die stattfinden soll, wenn eine ON ERROR-Bedingung aufgetreten ist, können Sie eines oder mehrere Statements angeben. **Verlassen eines ON ERROR-Blocks:** Ein ON ERROR-Statement-Block kann mit einem der Statements FETCH, STOP, TERMINATE, RETRY or ESCAPE ROUTINE verlassen werden. Wird der Block nicht mit einem dieser Statements verlassen, gibt Natural eine entsprechende Fehlermeldung aus und bricht die Ausführung des Programms ab.
END-ERROR	Das für Natural reservierte Wort END–ERROR muss zum Beenden eines ON–ERROR-Statement-Blocks benutzt werden.

ON ERROR-Verarbeitung in Unterprogrammen

Wird mittels CALLNAT, PERFORM oder FETCH RETURN eine Unterprogramm-Struktur aufgebaut, so darf jedes Modul dieser Struktur ein ON ERROR-Statement enthalten.

Tritt ein Fehler auf, so verfolgt Natural die Unterprogramm-Struktur automatisch zurück und wählt das erste in einem Unterprogramm gefundene ON ERROR-Statement zur Ausführung aus. Enthält keines der Module ein ON ERROR-Statement, gibt Natural eine entsprechende Fehlermeldung aus, und es erfolgt — wie oben beschrieben — ein Programmabbruch.

Systemvariablen *ERROR-NR und *ERROR-LINE

Die folgenden Natural-Systemvariablen können in Zusammenhang mit dem ON ERROR-Statement (wie in dem folgenden Beispiel gezeigt) benutzt werden:

***ERROR-NR**	Enthält die Fehlernummer des von Natural entdeckten Fehlers
***ERROR-LINE**	Enthält die Nummer der Sourcecode-Zeile, in der das Statement steht, das den Fehler verursacht hat.

Beispiel

```
** Example 'ONEEX1': ON ERROR
**
**
CAUTION: Executing this example will modify the database records!
************************************************************************
DEFINE DATA LOCAL
```

```
1 EMPLOY-VIEW VIEW OF EMPLOYEES
  2 NAME
  2 CITY
*
1 #NAME (A20)
1 #CITY (A20)
END-DEFINE
*
REPEAT
  INPUT 'ENTER NAME:' #NAME
  IF #NAME = ' '
    STOP
  END-IF
  FIND EMPLOY-VIEW WITH NAME = #NAME
    INPUT (AD=M) 'ENTER NEW VALUES:' ///
                 'NAME:' NAME /
                 'CITY:' CITY
    UPDATE
    END TRANSACTION
    /*
    ON ERROR
      IF *ERROR-NR = 3009
        WRITE 'LAST TRANSACTION NOT SUCCESSFUL'
            / 'HIT ENTER TO RESTART PROGRAM'
        FETCH 'ONEEX1'
      END-IF
      WRITE 'ERROR' *ERROR-NR 'OCCURRED IN PROGRAM' *PROGRAM
            'AT LINE' *ERROR-LINE
      FETCH 'MENU'
    END-ERROR
    /*
  END-FIND
END-REPEAT
END
```