# Processing of Application Errors

This section discusses the two basic methods Natural offers for the handling of application errors: default processing and application-specific processing. Furthermore, it describes the options you have to enable the application specific error processing: coding an `ON ERROR` statement block within a programming object or using a separate error transaction program.

Finally, this section gives an overview of the features that are provided to configure Natural's error processing behavior, to retrieve information on an error, to process or debug an application error.

This document covers the following topics:

- Natural's Default Error Processing
- Application Specific Error Processing
- Using an ON ERROR Statement Block
- Using an Error Transaction Program
- Error Processing Related Features

For information on error handling in a Natural RPC environment, see *Handling Errors* in the *Natural Remote Procedure Call* documentation.

---

## Natural's Default Error Processing

When an error occurs in a Natural application, Natural will by default proceed in the following way:

1. Natural terminates the execution of the currently running application object;
2. Natural issues an error message;
3. Natural returns to command input mode.

"Command input mode" means that, depending on your Natural configuration, the Natural main menu, the `NEXT` command prompt, or a user-defined startup menu may appear.

The displayed error message contains the Natural error number, the corresponding message text and the affected Natural object and line number where the error has occurred.

Because after the occurrence of an error the execution of the affected application object is terminated, the status of any pending database transactions may be affected by actions required by the setting of the profile parameters `ETEOP` and `ETIO`. Unless Natural has issued an `END TRANSACTION` statement as a result of the settings of these parameters, a `BACKOUT TRANSACTION` statement is issued when Natural returns to command input mode.

# Application Specific Error Processing

Natural enables you to adapt the error processing if the default error processing does not meet your application's requirements. Possible reasons to establish an application specific error processing may be:

- The information on the error is to be stored for further analysis by the application developer.
- The application execution is to be continued after error recovery, if possible.
- A specific transaction handling is necessary.

Because the execution of the affected Natural application object is terminated after an application error has occurred, the status of the pending database transactions may be influenced by actions which are triggered by the settings of the profile parameters `ETEOP` and `ETIO`. Therefore, further transaction handling (`END TRANSACTION` or `BACKOUT TRANSACTION` statement) has to be performed by the application's error processing.

To enable the application specific error processing, you have the following options:

- You may code an `ON ERROR` statement block within a programming object.
- You may use a separate error transaction program.

These options are described in the following sections.

# Using an ON ERROR Statement Block

You may use the `ON ERROR` statement to intercept execution time errors within the application where an error occurs.

From within an `ON ERROR` statement block, it is possible to resume application execution on the current level or on a superior level.

Moreover, you may specify an `ON ERROR` statement in multiple objects of an application in order to process any errors that have occurred on subordinate levels. Thus, application specific error processing may exactly be tailored to the application's needs.

### Exiting from an `ON ERROR` Statement Block

You may exit from an `ON ERROR` statement block by specifying one of the following statements:

- `RETRY`

  Application execution is resumed on the current level.

- `ESCAPE ROUTINE`

  Error processing is assumed to be complete and application execution is resumed on the superior level.

- FETCH

  Error processing is assumed to be complete and the "fetched" program is executed.

  STOP

  Natural stops the execution of the affected program, ends the application and returns to command input mode.

- TERMINATE

  The execution of the Natural application is stopped and also the Natural session is terminated.

### Error Processing Rules

- If the execution of the ON ERROR statement block is not terminated by one of these statements, the error is percolated to the Natural object on the superior level for processing by an ON ERROR statement block that exists there.
- If none of the Natural objects on any of the superior levels contains an ON ERROR statement block, but if an error transaction program has been specified (as described in the next section), this error transaction program will receive control.
- If none of the Natural objects on any of the superior levels contains an ON ERROR statement block and no error transaction program has been specified there, Natural's default error processing will be performed as described above.

# Using an Error Transaction Program

You may specify an error transaction program in the following places:

- In the profile parameter ETA.
- If Natural Security is installed, within the Natural Security library profile; see *Components of a Library Profile* in the *Natural Security* documentation.
- Within a Natural object by assigning the name of the program which is to receive control in the event of an error condition as a value to the system variable *ERROR-TA, using an ASSIGN, COMPUTE or MOVE statement.

If you assign the name of an error transaction program to the system variable *ERROR-TA during the Natural session, this assignment supersedes an error transaction program specified using the profile parameter ETA. Regardless of whether you use the ETA profile parameter or assign a value to the system variable *ERROR-TA, the error transaction program names are not saved and restored by Natural for different levels of the call hierarchy. Therefore, if you assign a name to the system variable *ERROR-TA in a Natural object, the specified program will be invoked to process any error that occurs in the current Natural session after the assignment.

On the one hand, if you specify an error transaction program by using the profile parameter ETA, an error transaction is defined for the complete Natural session without having the need for individual assignments in Natural objects. On the other hand, the method of assigning a program to the system variable *ERROR-TA provides more flexibility and, for example, allows you to have different error transaction programs in different application branches.

If the system variable *ERROR-TA is reset to blank, Natural's default error processing will be performed as described above.

If an error transaction program is specified and an application error occurs, execution of the application is terminated, and the specified error transaction program receives control to perform the following actions:

- Analyze the error;
- Log the error information;
- Terminate the Natural session;
- Continue the application execution by calling a program using the FETCH statement.

Because the error transaction program receives control in the same way as if it had been called from the command prompt, it is not possible to resume application execution in one of the Natural objects that were active at the time when the error occurred.

If the Natural profile parameter SYNERR is set to ON and a syntax error occurs at compile time, the error transaction program will also receive control.

An error transaction program must be located in the library to which you are currently logged on or in a current steplib library.

When an error occurs, Natural executes a STACK TOP DATA statement and places the following information at the top of the stack:

| **Field Contents** | | **Format/Length** |
|---|---|---|
| Natural error number | | N4 if session parameter SG is set to OFF; N5 if SG=ON |
| Number of the line where the error has occurred | | N4 |
| Status Code: | | A1 |
| C | Command processing error (the line number will be zero) | |
| L | Logon processing error (the line number will be zero) | |
| O | Object (execution) time error | |
| R | Error on remote server (in conjunction with Natural RPC) | |
| S | Syntax error | |
| Name of the Natural object where the error has occurred | | A8 |
| Level number of the Natural object where the error has occurred | | N2 |

If the Natural profile parameter SYNERR is set to ON and a syntax error occurs at compile time, the level number will be zero, and the following information will be stacked in addition:

| Field Contents | Format/Length |
|---|---|
| Position of the offending item in the source line | N3 |
| Length of the offending item | N3 |

This information can be retrieved in the error transaction program using an INPUT statement.

Example:

```
DEFINE DATA LOCAL
1 #ERROR-NR           (N5)
1 #LINE               (N4)
1 #STATUS-CODE        (A1)
1 #PROGRAM            (A8)
1 #LEVEL              (N2)
1 #POSITION-IN-LINE   (N3)
1 #LENGTH-OF-ERRTOKEN (N3)
END-DEFINE
IF *DATA > 6 THEN       /* SYNERR = ON and a syntax error occurred
  INPUT
    #ERROR-NR
    #LINE
    #STATUS-CODE
    #PROGRAM
    #LEVEL
    #POSITION-IN-LINE
    #LENGTH-OF-ERRTOKEN
ELSE
  INPUT                 /* other error
    #ERROR-NR
    #LINE
    #STATUS-CODE
    #PROGRAM
    #LEVEL
END-IF
WRITE #STATUS-CODE
* DECIDE ON FIRST VALUE OF STATUS-CODE
*  ... /* process error
* END-DECIDE
END
```

Some of the information placed on top of the stack is equivalent to the contents of several system variables that are available in an ON ERROR statement block:

| Field Contents | Equivalent System Variable in ON ERROR Statement Block |
|---|---|
| Natural error number | *ERROR-NR |
| Number of the line where the error has occurred | *ERROR-LINE |
| Name of the Natural object where the error has occurred | *PROGRAM |
| Level number of the program where the error has occurred | *LEVEL |

### Rules under Natural Security

If Natural Security is installed, the additional rules for the processing of logon errors apply. For further information, see *Transactions* in the *Natural Security* documentation.

# Error Processing Related Features

Natural provides a variety of error processing related features that

- Enable you to configure Natural's error processing behavior;
- Help you in retrieving information about errors that have occurred;
- Support you in processing these errors;
- Support you in debugging application errors.

These features can be grouped as follows:

- Profile parameters
- System variables
- Terminal Commands
- System commands
- Application programming interfaces

### Profile Parameters

The following profile parameters have an influence on the behavior of Natural in the event of an error:

| Profile Parameter | Purpose |
|---|---|
| CPCVERR | Conversion Error |
| DBGERR | Automatic Start of Debugger at Runtime Error |
| ETA | Error Transaction Program |
| ETEOP | Issue END TRANSACTION at End of Program |
| ETIO | Issue END TRANSACTION upon Terminal I/O |
| RCFIND | Handling of Response Code 113 for FIND Statement |
| RCGET | Handling of Response Code 113 for GET Statement |
| SYNERR | Control of Syntax Errors |

## System Variables

The following application related system variables can be used to locate an error or to obtain/specify the name of the program which is to receive control in the event of an error condition:

| System Variable | Content |
|---|---|
| *ERROR-LINE | Source-code line number of the statement that caused an error.<br><br>See Example 1. |
| *ERROR-NR | Error number of the error which caused an ON ERROR condition to be entered. |
| *ERROR-TA | Name of the program which is to receive control in the event of an error condition.<br><br>See Example 2. |
| *LEVEL | Level number of the Natural object where the error has occurred. |
| *LIBRARY-ID | Name of the library to which the user is currently logged on. |
| *PROGRAM | Name of the Natural object that is currently being executed.<br><br>See Example 1. |

Example 1:

```
...
    /*
    ON ERROR
      IF *ERROR-NR = 3009 THEN
        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
    /*
...
```

Example 2:

```
...
    *ERROR-TA := 'ERRORTA1'
    /* from now on, program ERRORTA1 will be invoked
    /* to process application errors
...
    MOVE 'ERRORTA2' TO *ERROR-TA
    /* change error transaction program to ERRORTA2
...
```

For further information on these system variables, see the corresponding sections in the *System Variables* documentation.

## Terminal Commands

The following terminal command has an influence on the behavior of Natural in the event of an error:

| Terminal Command | Purpose |
|---|---|
| %E= | Activate/Deactivate Error Processing |

## System Commands

The following system commands provide additional information on an error situation or invoke the utilities for debugging or logging database calls:

| System Command | Purpose |
|---|---|
| LASTMSG | Display additional information on the error situation which has occurred last. |
| RPCERR | Display the last Natural error number and message if it was Natural RPC related. |
| TECH | Display technical and other information about your Natural session, for example, information on the last error that occurred. |
| TEST | Invoke the debugger. |
| TEST DBLOG | Invoke the DBLOG utility, which is used for logging database calls. |

## Application Programming Interfaces

The following application programming interfaces (APIs) are generally available for getting additional information on an error situation or to install an error transaction.

| API | Purpose |
|---|---|
| USR0040N | Get type of last error |
| USR1016N | Display error level for copycode |
| USR1037N | Information about Natural ABEND data |
| USR1041N | Sample error transaction program (*ERROR-TA) |
| USR2001N | Read information about last error |
| USR2006N | Get detailed message information |
| USR2007N | Set/get RPC default server information |
| USR2010N | Display DB error information |
| USR2026N | Get TECH information |
| USR2030N | Read dynamic error parts :1:,... |
| USR2034N | Read system or user error message text from either FNAT or FUSER |
| USR3320N | Find user short error message from FNAT or FUSER |
| USR4214N | Display program level information |

For further information, see *SYSEXT - Natural Application Programming Interfaces* in the *Utilities* documentation.

For SQL calls, the following application programming interfaces are available:

| API | Purpose |
|---|---|
| NDBERR | Provides diagnostic information on the most recently executed SQL call. |
| NDBNOERR | Suppresses Natural’s error handling for errors caused by the next SQL call. |

For further information, see:

- *Interface Subprograms* in the *Natural for DB2* documentation.
- *Interface Subprograms* in the *Natural for SQL/DS* documentation.
- *Interface Subprograms* in the *Natural SQL Gateway* documentation.