ERROR$

ERROR$ generated a dialog to present an error with a possibility to terminate the process hereafter

Syntax

ERROR$(Document$, Error$, Caption$, Mode)

Arguments

- Document$ is a string value to be returned by this function as result (could be an error message)
- Error$ is the error string which will raise an error that gracefully terminates the session if the value is **not** PENDING and Error$ is **not** "NullStrin
  ". This string only controls the error trapping (and can be different from Document$)
- Caption$ is the title/description of the error that is raised
- Mode is an optional parameter to influence the type of buttons presented with the dialog<

  a. Mode% = 0: will create a dialog with **Abort/Retry/Ignore** buttons. Abort leads to a termination. Pressing the other buttons will continue the process;
  b. Mode% = 1: will create a dialog with **OK/Cancel** buttons. After pressing OK or Cancel the process will continue;
  c. Mode% = 2: will create a dialog with an **OK** button. After pressing OK the process will continue;
  d. Mode% = 3: will create a dialog with **Retry/Cancel** buttons. After pressing Retry or Cancel the process will continue;
  e. Mode% = 4: will create a dialog with **Yes/No** buttons. Yes leads to continuation of the process, No to a termination;
  f. Mode% = 5: will create a dialog with **Yes/No/Cancel** buttons. Yes or Cancel leads to continuation of the process, No to a termination;

Remarks

1. The INCASE function is a convenient function to use in combination with the ERROR$ function. The INCASE can be used to define Error$, either to be "NullString" or have a message depending on another value or parameter.
2. Mode's 2 or 4 will be the most useful options because these buttons most clearly provide a indication of what will happen.

Examples

Example 1

The ERROR$ function can be used to gracefully terminate a modeling session if a specific error is generated by a satellite program.

In the example below, the value of Error$ is generated by an INCASE() function interpreting the output of a program named ProgX.

The rogX output results are stored in the Telitab set
ProgX_calculation#

that is assumed to always contain a parameter ProgX_Errors$. The program is successfully completed if the value of
ProgX_calculation#.ProgX_Errors$

is equal to "No errors..."..

If the result of ProgX_calculation#.ProgX_Errors$ is different from "No errors...", a message box is shown with the error string and with the caption "Error running ProgX". Depending on the type of buttons specified in the dialog by means of the Mode parameter, the session can be either immediately be terminated, or you can decide to let Quaestor attempt to continue the session.

The syntax is as follows.

ProgX_Errors$ = ERROR$(ProgX_calculation#.ProgX_Errors$,INCASE(ProgX_calculation#.ProgX_Errors$ = "No errors...", THEN, "NullString", ELSE, ProgX_calculation#.ProgX_Errors$), "Error running ProgX", 4) + Qcrlf

Example 2

Please note that you can combine an understandable message with the error and the selected dialog buttons, such as the example below:

```
DESP_error_flag$ = ERROR$(INCASE(GET$("Desp.err","NullString", Total_Desp$)="NullString", THEN,
                                 "No errors occurred...", ELSE,
                                  DESP_Errors$)
                         ,INCASE(GET$("Desp.err","NullString", Total_Desp$)="NullString", THEN,
                                  "NullString", ELSE,
                                   GET$("Desp.err","NullString", Total_Desp$)+Qcrlf+Qcrlf+
                                   "Please check your input. Do you want to continue?")
                         ,"In the DESP calculation a fatal error has occurred (details shown below).",4)
```

If there is any error, it will show a window with the error as available in the file Desp.err including a description of the next step by adding +Qcrlf+Qcrlf+" Please check your input. Do you want to continue?" to the content of the file. In combination with Mode = 4 for the buttons, this makes an understandable error message.

Quick links: Functions overview | Attribute overview | Constants overview | Dimensions overview