On Error WTF?

Posted on March 17, 2015 by romperstomper

One of the more frequent questions I come across relates to the situation where an active and enabled error handler section handles the *first* error as expected but then fails to handle *any subsequent errors*. (An **enabled** error handler is one that is turned on by an On Error statement and an **active** error handler is an enabled handler that is in the process of handling an error.)

Here's the explanation (it's a little long, but bear with me!):

The On Error statement is the heart of VBA error-handling. Without an On Error statement, any run-time error that occurs will display an error message, and code execution will stop.

There are 4 distinct On Error options:

1. On Error Resume Next
2. On Error GoTo some_label/line_number
3. On Error Goto 0
4. On Error Goto -1

Search the site

Search...

Recent posts

IFS is not the same as nested IF functions March 19, 2019

Excel for Mac: are you freaking kidding me?? March 6, 2019

Power Pivot for all! Well, sort of. May 24, 2018

RiP Chip Pearson April 30, 2018

Object model bug with msoElementPrimaryValueAxisTitleAdjacentToAxis April 25, 2018

Top Posts & Pages

ByVal or ByRef - what's the difference?

Transpose bug in 2013 and 2016

Referring to Ranges in VBA

VBA references and early binding vs late binding

Late bound MSForms.DataObject

Home

When is ByRef not ByRef?

Excel forums

On Error WTF?

Office Update breaks ActiveX controls

On Error Resume Next

This is the simplest error handling option but also the most dangerous and most often misused. It ensures that when a run-time error occurs, control simply goes to the statement immediately following the statement where the error occurred, and execution continues from that point. There is no message to alert the user as to the fact that an error has occurred, or to what it might be. A typical good use of this structure is when there is a **predictable** error that you want to override – often assigning an object that may or may not exist to a variable. For example, when testing for the existence of a worksheet in a workbook, you can loop through all the worksheets checking the name of each one, or you can employ an On Error Resume Next statement like this:

```
Dim ws as Worksheet
On Error Resume Next
Set ws = activeworkbook.workshee
If not ws is nothing then
' do stuff
End If
```

The danger of this is if you do not remember to reset error handling (by either simply disabling it with *On Error Goto 0* or enabling an error handler – see below) all further errors in your code will be suppressed, which can make problems very hard to locate and debug – you may not even notice them until your code is already in real use, which is never a good thing!

Frequently I see people simply put *On Error Resume Next* at the top of their procedures when they can't figure out why an error is occurring –

Recent comments

romperstomper
You just can't. There is, ...

RV
It's more like Cousin IT. ...

Neman Syed
Amazing. Thank you for th ...

Excel Series.Value ...
[...] Heute, 09:42 Zum ...

Walter Rauschenber ...
The work is once, the ben ...

Post archive

Select Month

THIS IS NOT A GOOD IDEA!! It's the code equivalent of hearing a strange noise coming from your car engine and simply turning the radio up. Sure, you can't hear the noise anymore, but at some point something very bad is probably going to happen.

On Error GoTo some_label/line_number

Enables the error-handling routine that starts at the specified line label or number. If a run-time error occurs, control passes to that specified line, making the error handler **active**. (The specified line must be in the same procedure as the On Error statement, or a compile-time error will occur).

On Error GoTo 0

Disables any enabled error handler, including On Error Resume Next, in the current procedure. (It doesn't specify line 0 as the start of the error-handling code, even if the procedure contains a line numbered 0!) Without an On Error GoTo 0 statement, an error handler is automatically disabled when a procedure is exited normally.

On Error GoTo -1

Resets the active error status (exception) to Nothing without disabling any currently enabled error handler. You should very rarely see or use this. **If you find yourself using this, you should probably rethink the structure of your code.** (Like Goto 0, it does not specify line -1 as the start of the error-handling code, even if the procedure contains a line numbered -1). Without an On Error GoTo -1 statement, the active error is automatically reset when a procedure is exited normally.

Now that we've covered that, why does the original problem arise? (I'll wait while you go back and read the start to refresh your memory as to what the problem actually was)

Essentially there are two key concepts in error handling in VBA:

- whether an error handler is enabled (we covered this above)
- whether there is an active error condition – this can be a little surprising.

When an error occurs, an active error condition is set (what they call an exception in current VB). If there is no error handler, you see a message and code stops. That's pretty simple.

Where it gets interesting is if there is an *enabled* error handler. If there is, it becomes *active* until the active error condition is reset. The **only ways** to reset an active error condition and deactivate an error handler are via a Resume, Exit Sub, Exit Function, or Exit Property statement, or via an On Error Goto -1 statement. Note: On Error Goto 0 will deactivate an error handler, but **will not reset the active error condition** so you cannot follow it with another On Error statement (other than an On Error Goto -1 to clear the error) and hope to handle further errors. Hence, the following approach will not work:

```
Sub err_foo()

On Error GoTo err_handle

Err.Raise 5

Exit Sub



err_handle:

On Error GoTo 0

On Error Resume Next

Err.Raise 4

MsgBox “You will never see this message”

End Sub
```

While the current procedure’s error handler is active, or there is an active error condition, **no further errors can be handled by that procedure**. If another error occurs during this period, control returns to the calling procedure, if any, or an error message is produced and processing stops.

Typically in the questions I see, there is no Resume statement – there’s either a GoTo statement or the error handling label/line number is just the start of another section of code, or precedes a looping statement (Next, Wend, Loop for example). None of these scenarios will work because the error condition is not reset, and so the error handler is still **active**, and cannot handle further errors.

Sometimes I see people try to use Err.Clear to re-set the error condition but in actual fact this merely clears the properties of the Err object, which is always available and holds information about the last error to occur. It is not the same as the active error condition and cannot be used to reset it.

General comments:

An error-handling routine is not a Sub procedure or a Function procedure. It is simply a section of code marked by a line label or a line number.

To prevent error-handling code from running when no error has occurred, place an Exit Sub, Exit Function, or Exit Property statement imme-diately before the error-handling routine.

Examples:

I plan to add some code snippets here soon as a test of what you just read – your task will be to figure out what will happen in each of them be-fore actually running the code! 😉

Final takeaway:

If you find yourself using On Error Goto -1 a lot (or at all), you need to stop and rethink what you are doing! I have *never, ever*, seen well written code that required it and have never used it myself in actual production code.

Related

UDFs and Conditional Formatting	Transposing an array using an in-memory listbox	Office Update breaks ActiveX controls
June 17, 2014	May 21, 2013	December 10, 2014
With 2 comments	In "Array"	In "ActiveX"

Posted in Uncategorized | 32 Comments
Bookmark the permalink.

← When is a FormatCondition not a FormatCondition?

Free online Virtual conference hosted by MVPs →

32 thoughts on "On Error WTF?"

Pingback: Error Handler not activating

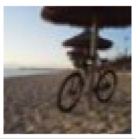
Pingback: Anonymous

Pingback: Testies - Page 4

Pingback: Comparing description between files, if matches pull price data

Pingback: Issue with Error Handler

Pingback: Multiple on error goto statements



***Alan* says:**

Hi,

Just a quick confirmation please:.

: On Error Resume Next' works' by or

"does" the following
. a) "instructing" to carry on following the line just after where the error occurred,
BUT ALSO:
. b) it prevents an exception being raised? (For this reason I can disable it regardless of what happened in the program by enabling a different error handler (or using the error statement On Error Goto 0)
. c) In effect On Error Resume Next results in things going on as if no error occurred. This is because with no raised exception VBA "knows" of no error?
. – Correct?
Thanks for Blogging
Alan

May 27, 2015 at 4:59 pm
Reply

Pingback: On Error Resume Label. Stuck as a sticky thing

Pingback: Multiple error handling

Pingback: "On Error" Statement nested in for loop

Pingback: VBA "On Error" Statement Not Being Recognised

Pingback: VBA Ignoring On Error GoTo Command.. XL 2010

Pingback: Error handling Resume v Goto

Pingback: On error GoTo doesn't work properly

Pingback: On Error GoTo only works once

Pingback: On Error GoTo - Issue

Pingback: checking if some specific workbook is open; if not then it should be opened.

Pingback: VBA Error handling stops working always on the same product (in this example)

Pingback: Using UNION and Ranges To Speed Up Deleting Many Columns?

Pingback: Funny behaviour when trying to check for a range's name

Pingback: Error handling

Pingback: ErrorHandling - RunTime Error 5 on Second Run

Pingback: Stepping Through Code With Unexpected Exit From Procedure

Pingback: On Error Goto Blue Does not work

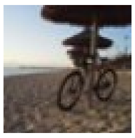
Pingback: on error goto issue

Pingback: multiple error handler fail in form image properties

Pingback: Error Handling Issues

Pingback: [on error go to] function

Pingback: Why this nested error handler works?



***Alan Elston* says:**

Hi
I avoid On Error Resume Next Error Handler as much as possible.
Occasionally I can find no alternative.
I understand that if I do use it, then the arousal to an Erected Exceptional Error condition is suppressed. Because of this, one may (and it is very advisable to) use the On Error GoTo 0 to disable this Error Handler. Alternatively using another Error Handler Statement , such as On Error GoTo some_label/line_number will also "change" the enabled error handler.
(Again this is only possible as the erecting of an Exceptional Error condition has been suppressed)

I see that often at some point after a On Error Resume Next , a code will use a check on a Property of the Err Object such as_…
If Err.Description = ""
_..to see if an Error occurred. (This is possible, as it would appear that even when the erecting of an Exceptional Error condition is suppressed, on an error occurring, the Err Object will still be filled with information about the last error that occurred)
I have seen some errors in code however, which attempt to use this technique to determine if an error had occurred. If one uses this technique, then the check should be done before any On Error GoTo 0 or On Error GoTo some_label/line_number. This is because it appears that these two Error Handler Statements Clear the Err Object. (This clearing can also be done using the Method Err.Clear)
Alan
_…
So one possible way of utilising the technique:
On Error Resume Next
' code that may error
If Err.Description ="" Then
' action to be taken for no error occurring
Else
' action to be taken on an error occurring
Err.Clear 'Clear the Err Object. This will allow the technique to be used again.
End If

_….
This would be better
On Error Resume Next
' code that may error
If Err.Description ="" Then
On Error GoTo 0 'Disable error handler
' action to be taken for no error
occurring
Else
On Error GoTo 0 'Disable error handler
and Clear the Err Object
' action to be taken on an error
occurring
End If

October 17, 2016 at 12:38 pm
Reply

Pingback: On Error goto

Pingback: On Error Goto -1

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

Website

☐ Save my name, email, and website in this browser for the next time I comment.

☐ Notify me of follow-up comments by email.

☐ Notify me of new posts by email.

Post Comment

This site uses Akismet to reduce spam. Learn how your comment data is processed.

Fund my coffee habit if you feel so inclined

To keep me permanently awake!

Donate