

By Kent Gorell

Error Handling

I use a standard error handler throughout my apps which replaces the MsgBox title with the application name instead of “Microsoft Access”.

Closing objects is handled in the Exit procedure so that all objects are closed or set to nothing even if an error occurs. I precede these with an “On Error Resume Next” as the last thing we want is an unhandled error occurring here.

In the error handler the first resume statement returns to the Exit_Procedure which handles the closing of any objects that may have otherwise been left open when an error occurs.

The second Resume is used for debugging only. If an error occurs, Access allows you to use Control + Break to debug. You should secure your code or create an MDE if you don’t want users access to this, I will create an article on this soon so check back.

Once in the code window you will be on then line following the message box. If you drag the yellow marker down to the second resume and hit F8 it will take you back to the line that failed.

```
On Error GoTo Error_Handler
```

```
Exit_Procedure:
```

```
    on error resume next
    <insert code to close objects or set
        them to nothing eg>
    rstCustomers.close
    Set clsOption = Nothing
    Exit Sub
```

```
Error_Handler:
```

```
    MsgBox "Error: " & Err.Number & "; Description: " _
        & Err.Description, _
        & vbCritical, CurrentDb.Properties("AppTitle")
    Resume Exit_Procedure
    Resume ` For debugging
```

Reports without data –

If you launch a report from a form, that allows you to set criteria for the report, then you will want it to elegantly handle the situation where the report has no data to fit the criteria.

In the report, inset this code in the No_Data event -

```
MsgBox "This Report Contains No Data", _  
    & vbOK, CurrentDb.Properties("AppTitle")  
Cancel = True
```

Then in the code on the form that opens the report you will need to handle the error that occurs when the report open is cancelled. The error number is 2501 so insert this code, in the error handler, to prevent a second message.

```
Error_Handler:  
Const NODATA as integer = 2501  
  
If Err.Number<> NODATA then  
    MsgBox "Error: " & Err.Number & "; Description: " _  
    & Err.Description, _  
    vbCritical, CurrentDb.Properties("AppTitle")  
End if  
Resume Exit_Procedure  
Resume ` For debugging
```

Debugging

If I hit an error while testing and debugging I open the code window with ctrl+Break which puts me in the error handler. Now I want to know which line failed, I drag the yellow current line arrow down to the Resume line so that the code resumes on the line that failed when I hit F8.

```
Resume Exit_Procedure
```

```
Resume ` For debugging
```

Thanks for Garry Robinson at [GR-FX](#) for this last tip.

Release Procedure

Whenever I release a version or update I have a set of procedures that I follow to ensure that none of my debugging code escapes into the wild.

This includes - Clear all breakpoints

For more tips or to get help with Your Access Database go to Osel's Access Development Site. [Free Access Tips](#)

For Help with your Access Database [Osel Access Consulting](#)