**Date:** 10/08/2012
**Procedure:** Visual Basic - ADO Errors
**Source:** LINK
**Permalink:** LINK
**Created by:** HeelpBook Staff
**Document Version:** 1.0

# VISUAL BASIC – ADO ERRORS

ADO errors are reported to your program as run-time errors. You can use the error-trapping mechanism of your programming language to trap and handle them. For example, in Visual Basic, use the **On Error** statement.

In Visual J++, use a **try-catch** block. In Visual C++, it depends on the method you are using to access the **ADO** libraries. With **#import**, use a **try-catch** block. Otherwise, C++ programmers need to explicitly retrieve the error object by calling **GetErrorInfo**. The following **Visual Basic** sub procedure demonstrates trapping an **ADO** error:

```
' BeginErrorHandlingVB01
Private Sub Form_Load()
' Turn on error handling
On Error GoTo FormLoadError
'Open the database and the recordset for processing.
Dim strCnn As String
strCnn = "Provider=sqloledb;" & _
"Data Source=a-iresmi2000;" & _
"Initial Catalog=Northwind;Integrated Security=SSPI"
' cnn is a Public Connection Object because
' it was defined WithEvents
Set cnn = New ADODB.Connection
cnn.Open strCnn
' The next line of code intentionally causes
' an error by trying to open a connection
' that has already been opened.
cnn.Open strCnn
' rst is a Public Recordset because it
' was defined WithEvents
Set rst = New ADODB.Recordset
rst.Open "Customers", cnn
Exit Sub
' Error handler
FormLoadError:
Dim strErr As String
Select Case Err
Case adErrObjectOpen
```

```
strErr = "Error #" & Err.Number & ": " & Err.Description & vbCrLf
strErr = strErr & "Error reported by: " & Err.Source & vbCrLf
strErr = strErr & "Help File: " & Err.HelpFile & vbCrLf
strErr = strErr & "Topic ID: " & Err.HelpContext
MsgBox strErr
Debug.Print strErr
Err.Clear
Resume Next
' If some other error occurs that
' has nothing to do with ADO, show
' the number and description and exit.
Case Else
strErr = "Error #" & Err.Number & ": " & Err.Description & vbCrLf
MsgBox strErr
Debug.Print strErr
Unload Me
End Select
End Sub
' EndErrorHandlingVB01
```

This **Form_Load** event procedure intentionally creates an error by trying to open the same **Connection** object twice. The second time the **Open** method is called, the error handler is activated. In this case the error is of type **adErrObjectOpen**, so the error handler displays the following message before resuming program execution:

```
Error #3705: Operation is not allowed when the object is open.
Error reported by: ADODB.Connection
Help File: E:\WINNT\HELP\ADO260.CHM Topic ID: 1003705
```

The error message includes each piece of information provided by the Visual Basic **Err** object except for the **LastDLLError** value, which does not apply here. The error number tells you which error has occurred. The description is useful in cases in which you do not want to handle the error yourself.

You can simply pass it along to the user. Although you will usually want to use messages customized for your application, you cannot anticipate every error; the description gives some clue as to what went wrong. In the sample code, the error was reported by the **Connection** object. You will see the object's type or programmatic ID here — not a variable name.

## Note

The Visual Basic **Err** object only contains information about the most recent error. The ADO **Errors** collection of the **Connection** object contains one **Error** object for each error raised by the most recent ADO operation. Use the **Errors** collection rather than the **Err** object to handle multiple errors. However, if there is no valid **Connection** object, the **Err** object is the only source for information about ADO errors.

What kinds of operations are likely to cause **ADO** errors? Common **ADO** errors can involve opening an object such as a **Connection** or **Recordset**, attempting to update data, or calling a method or property that is not supported by your provider.