

Re: Error handling problem.

Source:

<http://www.tech-archive.net/Archive/Windows/microsoft.public.windows.server.scripting/2005-04/msg00013.html>

- *From:* "Joe Earnest" <jearnest3-SPAM@xxxxxxxxxxxxxx>
 - *Date:* Thu, 31 Mar 2005 14:27:25 -0700
-

Hi,

"Brian Free" <bfree@xxxxxxxxxxxxxx> wrote in message
news:uxkSs1hNFHA.2880@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

>I have written a script that queries a specified OU and then resets the
>administrator password for the local machine account of all of the
>computers in the OU. The script appears to be working fine, but I'm having
>a problem with the error handling. For a reason I can't determine, when a
>computer is offline the error isn't reported until the next time through
>the loop, so the computer reported in error is actually fine, but the
>computer before it that doesn't get logged as a success or an error was
>actually in error. I can't figure out why Err doesn't get set until after
>the sub completes. Any help would be greatly appreciated. Below is the
>sub in use. I can provide the entire script if needed, but felt it would
>be too much to take in if I posted the whole thing right off.

>

> Thanks in advance,

> Brian Free

>

>

> Sub ChangePassword()

> ' Connect to the computer\administrator account

> Set objUser = GetObject("WinNT://" & strComputer & "/Administrator, user")

> If Err.Number <> 0 Then

> outFile.writeline Now & vbTab & "Error connecting to " & strComputer &

> " ---- " & Err.Description

> errorFile.writeline strComputer

> j = j + 1 'Add 1 to failed count

> Err.Clear

> ErrorOccurred = True

> Else

> ' Set the password for the account

> objUser.SetPassword newPassword

> objUser.SetInfo

> If Err.Number <> 0 Then

> outFile.writeline Now & vbTab & "Error setting password for " &

> strComputer & _

> "\Administrator" & " ---- " & Err.Description

Re: Error handling problem.

```
> errorFile.writeline strComputer
> j = j + 1 'Add 1 to failed count
> Err.Clear
> ErrorOccurred = True
> Else
> outFile.writeline (Now & vbTab & "Password set for " & strComputer &
> "\Administrator")
> successFile.writeline (strComputer)
> i = i + 1 'Add 1 to the success count!
> End If
> End If
> End Sub ' ChangePassword
```

Since no one else has jumped in yet, I suspect that you are having error scoping problems and that, if you remove the error handling lines that enclose the call to this routine, and place them in the routine, you'll get the result that you anticipate. This assumes that, though you haven't given us the calling code, that you have On Error statements surrounding a loop that calls this subroutine.

I suspect that when the error is encountered in your procedure, the script is jumping back up to the level in the calling code where the On Error statement is located, and handling the error there. The Err.Number property remains set on entry to the next call to this subroutine, and triggers the error handling at that time.

The Err object is implemented in scope-specific manner. The MS CHM file documentation on the ERR object is simply wrong in a couple of regards, and insufficient on the issue of scope. FWIW, here's a standard schpeal that I put together a few years back from some very informative posts by Michael Harris and Torgeir Bakken, and my own experimentation.

The MS VBS CHM file documentation indicates that the On Error Resume Next statement, but not the On Error Goto 0 statement, will implicitly clear (reset) any error data; and that Exit ... and End ... procedure statements will implicitly clear (reset) any error settings. In fact, VBS handles additional error object instances or settings in a scope-sensitive manner; both of the On Error Goto 0 does implicitly clear (reset) any error data; and Exit ... and End ... procedure statements do NOT implicitly clear (reset) any error settings.

The Err object is an intrinsic global object available under all hosts. When an error occurs, VBS will start with the present procedure, if any, and, if no error-trapping is presently in force in the procedure, will then look back along the call stack to the global script, to ascertain whether there is error-trapping status in effect at any level in the stack. If it finds one, it will jump to the next statement AT THAT LEVEL (potentially after the call to the error-producing procedure or one of its parent procedures). If not, VBS displays a runtime error message and terminates.

Re: Error handling problem.

Re: Error handling problem.

The Err object has two intrinsic methods: (1) Clear, which expressly clears the Err object properties and resets them, as if no error has occurred, and (2) Raise, which simulates a runtime error. Both of the On Error ... statements will implicitly clear (reset) the Err object properties. If exiting a procedure after a trapped error, without first executing an Err.Clear statement or proceeding through an On Error Goto 0 statement, the status of the Err object properties will be maintained, and any conditional statement in the script back along the call stack relying on that status (e.g., "if err then ...") will be implemented, regardless of whether error-trapping status is presently in force in that code.

The Err object has two particularly significant properties: (1) Number, which returns 0, if no error has been registered or if an error has been cleared (reset), or a non-zero long integer error code, indicating the error that has been registered; and (2) Description, which is a text description of the error. Number is the default property and can be omitted in assignment or condition statements (e.g., "cErr= err" or "if err then ..."). While the error code number is usually sufficient to identify an error, text descriptions are occasionally different for different errors represented by the same error code, so resort to the text of the description may sometimes also be necessary, to better resolve the nature of the error.

Invocation of error-trapping through On Error Resume Next in prior script in the calling stack does not extend into a called procedure's scope. This permits a script to place a generic error handler earlier in the calling stack, while still using specific error handlers in the later procedures. Unless one has well-crafted and specific plan for catch-all error handling in a script of limited scope, however, it is usually best to set local error handling only, and for as narrow a range as feasible, to avoid masking routine errors that can be corrected, as well as unforeseen errors that will be mistaken for a different type of error.

VBS error handling lacks call stack tracing, but that aspect can be added with privately implemented procedures or classes. Michael Harris has posted some elegant script creating an error call stack. Similarly, VBS error handling lacks the ability to jump to different line numbers on errors. That ability can be partially simulated, where necessary, through procedure design as simple as pseudo-labels in a do-looped select case block, so that implementation of script arguments that call those procedures in different order, and the script reruns itself with the appropriate argument and quits, when a specific error is encountered.

One especially touchy area of error handling involves the use of With blocks. A With block must always be literally exited, by allowing operation to proceed through the End With statement. As the MS WSH documentation makes clear, you cannot exit a With by simply terminating the script or through any of the standard block exits, without risking "memory leaks" and/or some very peculiar errors, including some unenlightening error messages generated when the script tries to quit. So it may be necessary to use one or more faux-loops inside a With block, to exit to the End With

Re: Error handling problem.

Re: Error handling problem.

statement, prior to acting on the error.

Joe Earnest

-
- *Follow-Ups:*
 - ◆ *Re: Error handling problem.*
 - ◇ *From:* Brian Free

 - *References:*
 - ◆ *Error handling problem.*
 - ◇ *From:* Brian Free

 - Prev by Date: *Error handling problem.*
 - Next by Date: *Re: Error handling problem.*
 - Previous by thread: *Error handling problem.*
 - Next by thread: *Re: Error handling problem.*
 - Index(es):
 - ◆ *Date*
 - ◆ *Thread*