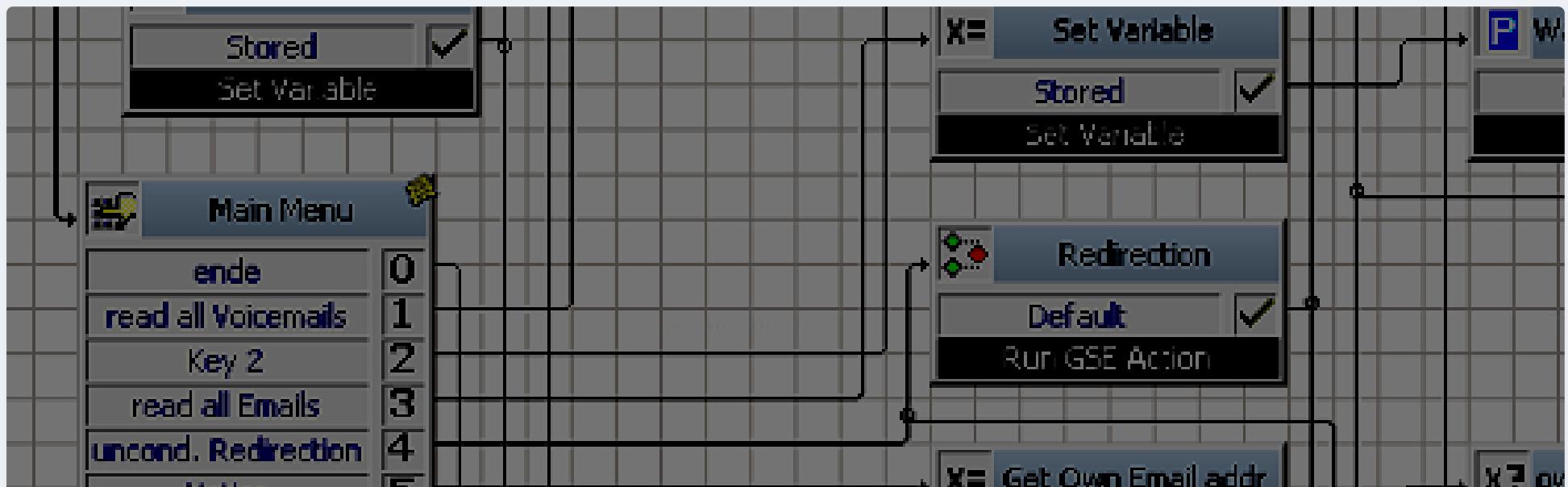


## The Call Routing Guy

A blog by Tom Wellige in General

Followers

1



## #9: Don't be shy, be chatty!



Entry posted by Tom Wellige in General June 17, 2023  
1,611 views

[Share](#)

Followers

1

VBScript

Lua

When it comes to develop call routing scripts with some own VBScript code in it, it might become handy to be able to have a look on what your code is doing.

There are lots of so called "debugging" possibilities, but at this point you have to believe me, the only really reliably working method is placing your own log/trace information into the server trace file.

Before you say that those trace files are way too large and confusing for you, they are not, once you know how to read them and how to filter the stuff you are interested in out. In the forum post "[How to filter SwyxWare traces for call routing output of a single call](#)" I have explained already in detail how this can easily be done. So lets focus on how to get your own information into the server trace file.

This can simply be done with `PBXScript.OutputTrace` (Vbs)/ `PBXScript.OutputTrace()` (Lua).

For VBScript based Call Routing:

```
PBXScript.OutputTrace "Hello World!"
```

For Lua based Call Routing:

```
PBXScript.OutputTrace("Hello World")
```

So what should be written into the server trace and what not? And how will I find my own trace output quickly and easily?

By taking a look into the server trace file you figure that the SwyxWare traces quite a lot of stuff, not only errors but also tons of other information and events. You should do the same in your own VBScript/Lua code as well. Do not only trace errors your run into but also every kind of information that might become useful later on when you have to analyse the call routing if for some reason it doesn't work anymore like the many months before. The more information you place into the trace file the more easy it will become to pin point a problem later on. You might not even have to get into the script and work yourself into it if the traces are already informative enough.

Lets assume you have written yourself a VBScript/Lua function to do what ever you need to do, placed it into the **Start** block and call it later on e.g. in an **Evaluate** block.

Taking all said above into consideration, that following is something like a "best practice" function template in terms of tracing.

For VBScript based Call Routing:

```
Function DoWhatever ( Parameter1, Parameter2 )

    PBXScript.OutputTrace "-----> DoWhatever"
    PBXScript.OutputTrace "Parameter1 = " & Parameter1
    PBXScript.OutputTrace "Parameter2 = " & Parameter2

    On Error Resume Next

    Dim bReturn
    bReturn = False

    ' do what ever your function needs to do
    ' ...
    If Err <> 0 Then
        PBXScript.OutputTrace "Error while doing something!"
        PBXScript.OutputTrace "(Err) " & Err.Description
    Else
        PBXScript.OutputTrace "Successfully did something"
    End If

    ' if you "calculate" something, it is always a good idea to trace the result of it
    Dim sSQL
    sSQL = "SELECT * FROM MyTable WHERE value1 = " & Parameter1 & " AND value2 = " & Parameter2
    PBXScript.OutputTrace sSQL

    DoWhatever = bReturn
```

```
PBXScript.OutputTrace "bReturn = " & bReturn
PBXScript.OutputTrace "<----- DoWhatever"
```

```
End Function
```

For Lua based Call Routing:

```
function DoWhatever ( Parameter1, Parameter2 )

    PBXScript.OutputTrace ("-----> DoWhatever")
    PBXScript.OutputTrace ("Parameter1 = " .. Parameter1)
    PBXScript.OutputTrace ("Parameter2 = " .. Parameter2)

    local bReturn = false
    local oResult = nil

    -- do what ever your function needs to do
    -- ...
    if (oResult == nil) then
        PBXScript.OutputTrace ("Error while doing something!")
    else
        PBXScript.OutputTrace ("Successfully did something")
    end

    -- if you "calculate" something, it is always a good idea to trace the result of it
    local sSQL
    sSQL = "SELECT * FROM MyTable WHERE value1 = " .. Parameter1 .. " AND value2 = " .. Parameter2
    PBXScript.OutputTrace ("sSQL = " .. sSQL)

    PBXScript.OutputTrace ("bReturn = " .. tostring(bReturn))
```

```
PBXScript.OutputTrace ("<----- DoWhatever")  
  
    return bReturn  
end
```

Lets go through the template, step by step...

The first and the last thing you function should do it to trace that you have entered and left it again.

For VBScript based Call Routing:

```
PBXScript.OutputTrace "-----> DoWhatever"  
  
PBXScript.OutputTrace "<----- DoWhatever"
```

For Lua based Call Routing

```
PBXScript.OutputTrace ("-----> DoWhatever")  
  
PBXScript.OutputTrace ("<----- DoWhatever")
```

The arrows should tell you that you go into your function and go out of it again. All trace lines in between these two arrow lines will be made by your function. So they also help to find your stuff quickly.

If your function takes parameters it is a good idea to trace them. If your function doesn't work and it get already come crap parameters it is not surprising that it fails later on.

For VBScript based Call Routing:

```
PBXScript.OutputTrace "Parameter1 = " & Parameter1
PBXScript.OutputTrace "Parameter2 = " & Parameter2
```

For Lua based Call Routing:

```
PBXScript.OutputTrace ("Parameter1 = " .. Parameter1)
PBXScript.OutputTrace ("Parameter2 = " .. Parameter2)
```

For VBScript based Call Routing: If you have disabled the standard error handling (will be part of a future blog post) you need to check every "dangerous" call yourself, if it succeeded or not. For example you try to write something into a text file it could happen that the file doesn't exist or is read only. Or you try to access a database which is currently not reachable. So if you check for the outcome of such a call do not only trace errors but also that it succeeded.

```
If Err <> 0 Then
    PBXScript.OutputTrace "Error while doing something!"
    PBXScript.OutputTrace "(Err) " & Err.Description
Else
    PBXScript.OutputTrace "Successfully did something"
End If
```

For Lua based Call Routing:

```
if (oResult == nil) then
    PBXScript.OutputTrace ("Error while doing something!")
else
    PBXScript.OutputTrace ("Successfully did something")
end
```

If you calculate something within your function, e.g. build an SQL statement to run against a database and use some current values (e.g. the function parameters) to do so, it is a good idea to write the result of this into the trace. To stay with the SQL statement example this will enable you later on to copy&paste the statement from the trace file and test it directly against your database.

For VBScript based Call Routing:

```
Dim sSQL
sSQL = "SELECT * FROM MyTable WHERE value1 = " & Parameter1 & " AND value2 = " & Parameter2
PBXScript.OutputTrace sSQL
```

For Lua based Call Routing:

```
local sSQL
sSQL = "SELECT * FROM MyTable WHERE value1 = " .. Parameter1 .. " AND value2 = " .. Parameter2
PBXScript.OutputTrace ("sSQL = " .. sSQL)
```

And finally, if your function returns a value, just trace this value.

For VBScript based Call Routing:

```
PBXScript.OutputTrace "bReturn = " & bReturn
```

For Lua based Call Routing:

```
PBXScript.OutputTrace ("bReturn = " .. tostring(bReturn))
```

You don't need to put all trace stuff into your function from the beginning, while you are still developing it. But once it is finished and tested you should put trace output in it. You can't trace too much information, just too few. The more information you trace the more easy it will become later on to identify a problem if the call routing script ceases to work.

Enjoy!

PS: don't miss to take a look into the [ECR Useful Link Collection](#)

< Previous entry  
#3: Be more flexible on dates!

Next entry >  
#18: Record and replace Announcements

## 0 Comments

There are no comments to display.

## Create an account or sign in to comment

You need to be a member in order to leave a comment

### Create an account

Sign up for a new account in our community. It's easy!

### Sign in

Already have an account? Sign in here.



[Register a new account](#)

[Sign In Now](#)

Theme ▼

Copyright (c) 2007-2025 by Tom Wellige

Powered by Invision Community