

SAP Automated Test Runner Documentation

















Adrià Baltrons Mata

Web Team Trainee

Date: 4/1/2025

Index

 SAP Automated Test Runner - Documentation	2
 What This Does	2
 Folder Structure	3
 How to Create a Test Case	4
 Built-in Utility Functions	5
Step 1: Create helpers.vbs	5
Step 2: Load Helpers in Each Test Automatically	6
 Example Test Script (VBScript)	7
 Environment Selection: DEMO vs. PROD	9
 How to Run a Test	10
 What You Get	11
 How It Works	11
 Advanced	12
 Cleanup and Safety	12
 Want to Create a New Test?	13
 Authors	13



SAP Automated Test Runner - Documentation

Welcome to your custom **End-to-End Test Automation Framework for SAP GUI**, designed for QA analysts and SAP testers who are not necessarily web developers. This framework allows you to run SAP test scripts, capture screenshots automatically, and generate professional PDF evidence — all with minimal effort.



What This Does

This framework automates the following E2E flow:

1. Launch SAP GUI using credentials
2. Run your SAP `.vbs` test script
3. Automatically capture screenshots at key steps
4. Generate a professional PDF report with metadata
5. Clean up files and close SAP GUI automatically

Folder Structure

/e2e

```
|— /SAP
|   |— runner.js          # Main script to run SAP E2E tests
|   |— .env              # SAP_USER and SAP_PASSWORD
|   |— testcases.json     # Test metadata (titles, descriptions, etc.)
|   |— README.md         # QA-friendly doc
|
|   |— /tests            # SAP .vbs scripts (your actual test cases)
|   |   |— Test_CreateClient.vbs
|   |   |— Test_CheckStock.vbs
|   |   └— Test_Order.vbs
|
|   |— /scripts          # Helper scripts (login, setup)
|   |   |— openSAP.vbs
|   |   └— helpers.vbs    # Shared VBScript functions
|
|   |— /evidence          # Auto-generated screenshots (cleaned)
|
|   └— /pdf              # Output folder for final PDF reports
|       └— SAP_CreateClient.pdf
|
|— /Support
|   └— PDFReport/
|       |— generatePdfReport.js
|       |— formatDuration.js
|       └— werfen-logo.png
```

How to Create a Test Case

1. **Record your SAP script** using SAP GUI scripting (VBScript)
2. Use the function `RequestScreenshot("Step description")` to flag each screenshot step:

`RequestScreenshot "Pressed Execute Button"`

3. On test failure, use the `CreateFailFlag "Reason"` function to mark a failed step and stop execution:


`CreateFailFlag "Material code was incorrect"`

4. At the end of the script, create an `end.flag`:

```
Set end = fso.CreateTextFile("C:\SAP-CAPTURES\end.flag", True)
```

```
end.WriteLine "END"
```

```
end.Close
```

 That's all you need to connect to the runner.

Built-in Utility Functions

You can avoid rewriting common logic like screenshot capture or fail handling in every test by placing them in a shared file and injecting them at runtime.

Step 1: Create **helpers.vbs**

Inside **SAP/scripts/helpers.vbs**:

```
Set fso = CreateObject("Scripting.FileSystemObject")

Sub RequestScreenshot(stepName)

    Dim flag

    Set flag = fso.CreateTextFile("C:\SAP-CAPTURES\signal.flag", True)

    flag.WriteLine stepName

    flag.Close

    WScript.Sleep 1500

End Sub

Sub CreateFailFlag(reason)

    Dim failFlag

    Set failFlag = fso.CreateTextFile("C:\SAP-CAPTURES\fail.flag", True)

    failFlag.WriteLine reason

    failFlag.Close

    WScript.Echo "❌ Fail flag created: " & reason

    WScript.Quit

End Sub
```

Step 2: Load Helpers in Each Test Automatically

At the top of your test `.vbs` file:

```
Dim helperCode
```

```
Set f =
```

```
CreateObject("Scripting.FileSystemObject").OpenTextFile("C:\path\to\SAP\scripts\hel  
pers.vbs", 1)
```

```
helperCode = f.ReadAll
```

```
f.Close
```

```
ExecuteGlobal helperCode
```

This way, `RequestScreenshot` and `CreateFailFlag` are available without redefining them in every test.



Example Test Script (VBScript)

Save inside `SAP/tests/Test_CheckStock.vbs`:

```
On Error Resume Next
```

```
Dim SapGuiAuto, application, connection, session
```

```
Dim fso, helperCode, f
```

```
Set fso = CreateObject("Scripting.FileSystemObject")
```

```
' Load shared helper functions
```

```
Set f = fso.OpenTextFile("C:\path\to\SAP\scripts\helpers.vbs", 1)
```

```
helperCode = f.ReadAll
```

```
f.Close
```

```
ExecuteGlobal helperCode
```

```
' Connect to SAP GUI
```

```
Set SapGuiAuto = GetObject("SAPGUI")
```

```
Set application = SapGuiAuto.GetScriptingEngine
```

```
Set connection = application.Children(0)
```

```
Set session = connection.Children(0)
```

```
session.findById("wnd[0]").maximize
```



```
session.findById("wnd[0]/usr/ctxtS_MATNR-LOW").text = "MAT-001"
```

```
RequestScreenshot "Material code entered"
```

```
session.findById("wnd[0]/tbar[1]/btn[8]").press
```

```
RequestScreenshot "Search executed"
```

```
If session.findById("wnd[0]/usr/lbl[1,1]").Text <> "Available" Then
```

```
    CreateFailFlag "Stock status not available"
```

```
End If
```

```
RequestScreenshot "Stock is available"
```

```
Dim endFlag
```

```
Set endFlag = fso.CreateTextFile("C:\SAP-CAPTURES\end.flag", True)
```

```
endFlag.WriteLine "END"
```

```
endFlag.Close
```

Environment Selection: DEMO vs. PROD

You can define which SAP environment to open by setting `SAP_ENV` as an environment variable (`PROD` or `DEMO`).

In your `.env` file:

```
SAP_USER=yourusername
```

```
SAP_PASSWORD=yourpassword
```

```
SAP_ENV=PROD
```

The `openSAP.vbs` script will map this to the correct connection:

```
If UCase(sapEnv) = "PROD" Then
```

```
    sapConn = "ERP__PRD_CH1"
```

```
Elseif UCase(sapEnv) = "DEMO" Then
```

```
    sapConn = "ERP_QAS_CH2"
```

```
Else
```

```
    MsgBox "❌ Invalid SAP_ENV value: " & sapEnv
```

```
    WScript.Quit
```

```
End If
```

This makes it easier to switch SAP environments without editing your `.vbs`.

How to Run a Test

Make sure you have a `.env` file like:

`SAP_USER=yourusername`

`SAP_PASSWORD=yourpassword`

`SAP_ENV=PROD`

Then from the command line:

`node ./SAP/runner.js --index=1 --env=PROD`

`--script=./SAP/tests/Test_CheckStock.vbs`

- `--index=1` → selects the 2nd test from `testcases.json`
- `--env=PROD` → test environment shown in PDF and used for SAP login
- `--script=...` → VBS test script to execute



What You Get

After successful execution, you'll find a file like:

SAP/pdf/SAP_Stock Consultation.pdf

It includes:

- General test metadata
- One page per screenshot with the description
- Timestamps and tester name



How It Works

1. ****Runner starts **openSAP.vbs**
2. VBS logs in with env vars: **SAP_USER, SAP_PASSWORD, SAP_ENV**
3. Test script is executed (VBS)
4. Whenever **RequestScreenshot()** is called, it creates a **signal.flag**
5. Node detects this and takes a desktop screenshot
6. When **end.flag** is found, runner stops
7. If **fail.flag** exists, it logs an error and skips PDF generation

Advanced

- VBS can assert SAP values using:

If `table.GetCellValue(0, "MATNR") <> "ABC123"` Then ...

- The `runner.js` will detect failure automatically via `fail.flag`
- All screenshot steps are timestamped and cleaned afterward

Cleanup and Safety

- `cleanSapCaptureFolder()` ensures that flags are deleted after each run
- `closeSAP()` force-kills `saplogon.exe` in case SAP is stuck
- Timeout kills the test if it's taking too long (default: 5 min)



Want to Create a New Test?

1. Record a `.vbs` script with SAP GUI
2. Add `RequestScreenshot()` calls where you want evidence
3. Use `CreateFailFlag()` in any condition that should stop the test
4. Add a test entry in `testcases.json`:

```
{  
  
  "code": "SAP1004",  
  
  "title": "Sales_Order_Creation",  
  
  "description": "End-to-end sales order creation",  
  
  "expectedResults": "Order is saved successfully in SAP"  
}
```

5. Run via:

```
node ./SAP/runner.js --index=3 --env=QA --script=./SAP/tests/Test_Order.vbs
```



Authors

Built by Adrià Baltrons Mata, text me through Teams if you have any question 🔥