SAP System Integration Test Plan

End-to-end process testing for an S/4HANA program

SIT proves that end-to-end business processes work across modules, interfaces and surrounding systems. It is where most defects are found, where most manual effort is spent, and where a program discovers whether its integration design actually holds.

> ***How to use this template.*** *Everything in italics is guidance and should be deleted once you have replaced it. Square brackets mark a value to fill in. These four documents are designed to be used together: the strategy sets the rules, and the three plans execute them at each level. Where your program uses different level names, keep your names and map them in section 1 rather than renaming everything.*

1. Cycles and scope

Cycle	Dates	Scope	Entry criteria	Exit criteria
Cycle 1	[ ]	[core happy paths]	[FUT complete]	[all scenarios executed once]
Cycle 2	[ ]	[variants, error paths, volumes]	[cycle 1 defects resolved]	[ ]
Cycle 3	[ ]	[regression + residual defects]	[ ]	[ ]

> *Resist the temptation to plan only one cycle. Cycle 1 finds design problems; cycle 2 finds the defects you actually go live with or without.*

2. End-to-end scenarios

> *A SIT scenario is a business journey, not a transaction. Name it the way the business would.*

Scenario ID	Business journey	Systems touched	Process IDs	Cycle	Owner
[SIT-07]	[standard sales order to cash collection]	[S/4, bank, tax engine, output]	[O2C-01, R2R-03]	[1,2,3]	[ ]
[SIT-12]	[requisition to supplier payment]	[Ariba, S/4, bank]	[P2P-01]	[1,2,3]	[ ]

3. Interfaces in scope

> *List every interface the scenarios cross. The seam between two correctly configured systems is the most reliable place in an SAP program for a defect to hide.*

Interface	Direction	Technology (IDoc / API / file / BTP)	Partner system	Sync or async	Error handling tested?
[ORDERS05]	[inbound]	[IDoc]	[EDI partner]	[async]	[ ]

For each asynchronous interface, record the **expected latency**. A test that checks the receiving side immediately will fail intermittently forever, and intermittent failure in SIT is indistinguishable from a real defect until somebody spends a day on it.

4. Master data and org structure

Master data loaded for SIT, and from where

Org structure in use (company codes, plants, sales orgs)

Number ranges

Refresh policy during SIT

Who to contact when data is exhausted

> Agree the refresh policy before cycle 1. A mid-cycle refresh that nobody expected invalidates every in-flight test and is a common cause of a lost week.

5. Execution tracking

Scenario	Cycle 1	Cycle 2	Cycle 3	Defects raised	Status
[SIT-07]	[pass/fail]			[ ]	[ ]

6. Defect management

Severity	Definition	Response	Blocks cycle exit?
1	[process cannot complete; no workaround]	[ ]	Yes
2	[process completes with wrong result]	[ ]	Yes
3	[workaround exists]	[ ]	No
4	[cosmetic]	[ ]	No

Track where each defect originated — configuration, custom code, interface, data, or requirement. After cycle 1 that distribution tells you what to fix about the program, not just about the system.

7. Automation candidates from SIT

By the end of cycle 2 you will have executed the same scenarios by hand two or three times, and you will do so again at every wave, every transport and every support pack. That repetition is the automation case.

Scenario	Times executed manually so far	Access route OK?	Deterministic?	Automate for regression?
[SIT-07]	[3]	[Fiori — yes]	[yes]	[yes]

Studio drives a **real browser**. That covers **SAP Fiori** and **SAP GUI for HTML (Web GUI)**, plus the surrounding web applications in scope — Ariba, SuccessFactors, Concur, supplier and customer portals, and your own custom front ends. It does **not** drive SAP GUI for Windows, which is a desktop client. Classify each in-scope transaction by its access route before you plan automation, because that one column decides what is and is not a candidate.

> **With Functionize Studio:** a scenario that has been walked by hand three times is already fully specified — the steps and the expected values exist. Describing it once in plain language turns it into a test that runs on every transport from then on, which is where a phased program gets its capacity back.

8. Exit

Exit criterion	Met?
All in-scope scenarios executed in the final cycle	[ ]
No open severity 1 or 2 defects	[ ]
All interfaces exercised including error paths	[ ]
Regression candidates identified and agreed	[ ]
UAT entry criteria confirmed achievable	[ ]