

Cross-Border Solution Architecture Playbook

How I would approach taking enterprise and BFSI clients from first call to scale, across languages, cultures and time zones.

Role at a glance

Discover 1-2 → Design 3 → Integrate 4-5 → Launch 6 → Optimize 7

One architect, one solution, end to end.



Solution architecture & integration

End-to-end solutions and workflows, from discovery to post-go-live.



Pre-sales & technical discovery

Discovery sessions, product deep-dives, security queries and POCs with Sales.



Enterprise & BFSI delivery

Trusted advisor through strict security, compliance and data privacy constraints.



Cross-functional bridge

Market needs and client feedback into the product roadmap.



Troubleshooting & optimization

Fix integration issues, tune API and SDK performance, build execution frameworks.

Delivery flow

1



Discovery: know the client and the region

Regional nuance · communication styles

✗ A "yes" on the call can mean "I heard you", not "we agree".

✓ Map stakeholders, systems and constraints, plus language, time overlap and data law. Meet the local partner.

🏁 **Done when:** Stakeholders, systems, volumes and the data law are written down and confirmed by the client.

2

**Requirements: agree the scope and the words**

Terminology gaps · translation teams

✗ "Webhook" translated three ways in three documents.

✓ A signed-off scope and one shared glossary. AI drafts translations, a native-speaking partner reviews them.

🏁 **Done when:** Scope is signed off and the glossary is locked in both languages.

3

**Solution design: a visual blueprint**

Visual integration blueprints

✗ Long English documents nobody finishes reading.

✓ Diagrams and code first, short sentences second. See the solution blueprint below.

🏁 **Done when:** The client's tech lead approves the blueprint and the data flow.

4

**Integration: an API kit that runs**

Standardized API docs · Postman

✗ Every team reads the API docs differently.

✓ A shared Postman template, filled with the real endpoints during integration. Engineers test instead of reading.

🏁 **Done when:** Every Postman request passes in UAT, and duplicate webhooks are handled safely.

5

**Delivery cadence: async first, live in the overlap**

Runs through every step · W1-W10

Async loops · time zone friction · high-context cultures

✗ One question takes three days of email. Blockers are never said in the group call.

✓ Short calls in shared hours, a written recap in both languages, and a 1:1 with the tech lead.

🏁 **Done when:** Every open question has an owner and a date in the shared log.

6

**Security sign-off & go-live**

Review starts W3

Local data privacy mandates

✗ Go-live stalls while security asks where the selfies are stored.

✓ Bring a data map, the local rules and the security documents to the first security call.

🏁 **Done when:** Security sign-off is in and the [go-live checklist](#) is complete.



Post-go-live: hypercare, scale and support

Troubleshooting · optimization · long-term adoption

- ✗ After launch, drop-offs and slow responses go unnoticed until customers complain.

- ✓ Hypercare, live dashboards with alerts, and a named escalation path in the client's language.

Proposed targets · to agree with the client in the SLA



Hypercare **PROPOSED**

First 30 days: daily check-in, then weekly. A shared issue log.



Monitor **PROPOSED**

Completion funnel, p95 latency, error codes and auto-approve rate, with alerts on drops.



Scale **PROPOSED**

Load-test at 3× peak before campaigns, paydays, Tét and Ramadan.



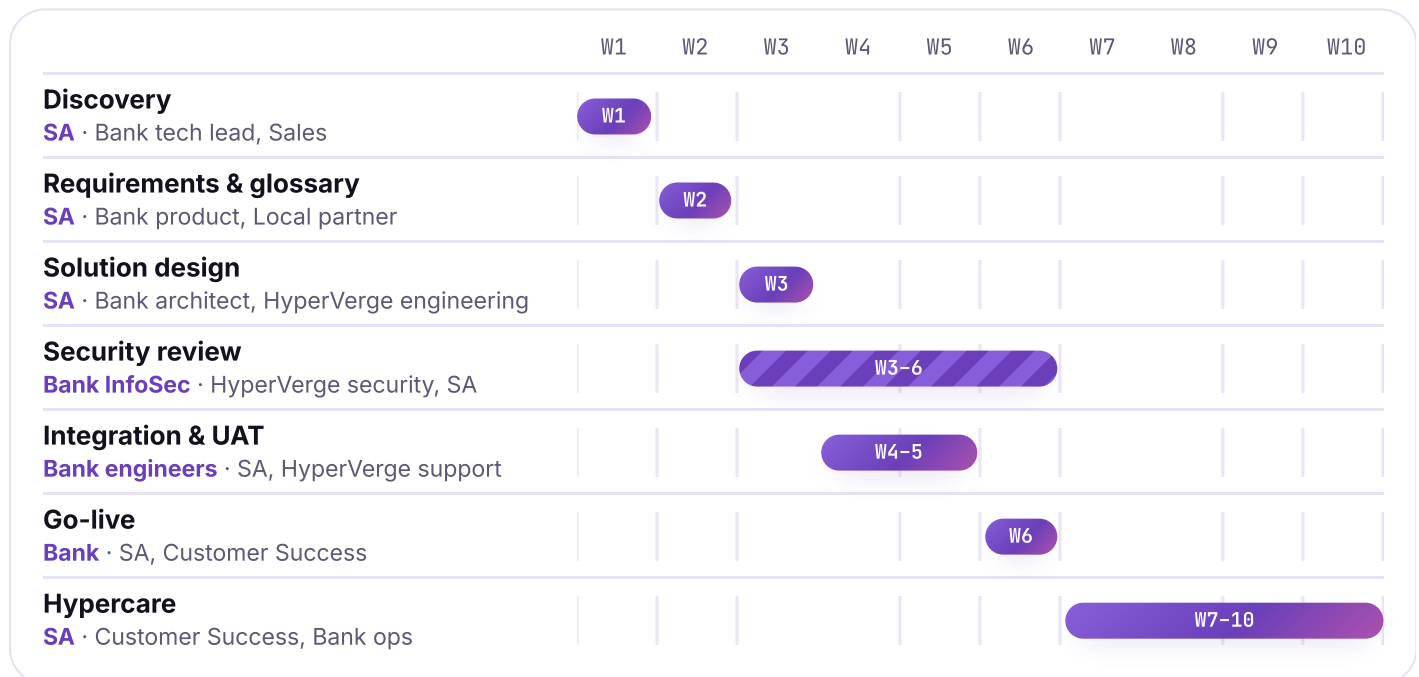
Support **PROPOSED**

P1 response in 30 min, root cause in 48 h, planned SDK upgrades.

🚩 **Done when:** 30 days with no P1 issues, and metrics holding at the agreed targets.

Delivery plan **PROPOSED**

A typical 10-week path for one bank. The security review runs alongside the build, so it never blocks go-live at the end.



Discovery & POC

Discovery questions

- 1 Where do customers drop off today, and how many apply each month?
- 2 Which ID documents and versions are in use (chip cards, older cards)?
- 3 Which phones and networks do your customers have?
- 4 Which regulators and internal policies apply?
- 5 Where must data be stored and processed?
- 6 What happens today to a declined or uncertain case?
- 7 Who signs off security, and how long does that review take?
- 8 What does success look like in 90 days?

POC plan **PROPOSED**

Two to three weeks, one journey, the client's real devices and real ID cards.

Completion rate

Started vs finished, against today's baseline

False rejections

Genuine customers wrongly declined

Latency

p95 time per step and end to end

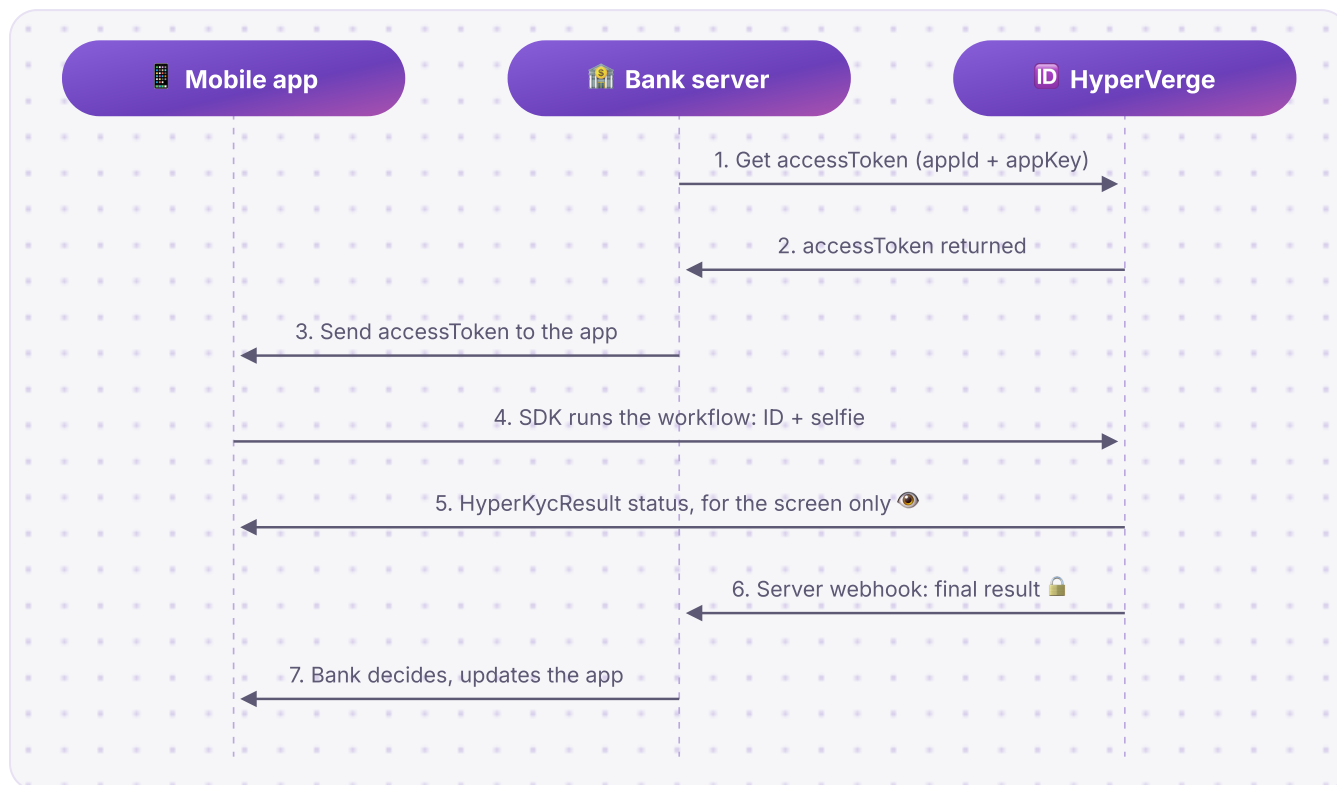
Fraud caught

On an agreed test set of spoofs and fake IDs

- ✓ Success criteria and targets signed before the POC starts
- ✓ Daily written update, in both languages
- ✓ Executive readout ending in a clear go / no-go

Solution blueprint (live demo), in their language Interactive

MESSAGE FLOW



👁 Display only, never used to approve 🔒 Server-to-server: the only result the bank decides on

Arabic mirrors right-to-left

API kit

Postman collection SAMPLE

Four requests in HyperKYC SDK terms: appld/appKey, accessToken, workflowId and transactionId.

POST Generate accessToken
server only, never in the app

POST Fetch results by transactionId
for missed webhooks

POST Replay a sample webhook
to your server

POST Replay it again
duplicates must be safe

A sample collection to import into Postman. Endpoint URLs and the webhook body are placeholders to replace from your HyperVerge docs.

Failure handling

What breaks in production, how we spot it, and what the customer sees instead.

Liveness or face match fails

DETECT

auto_declined or needs_review

FALLBACK

Guided retry, then manual review or video KYC.

Network drops mid-capture

DETECT

error or user_cancelled from the SDK

FALLBACK

Resume with the same transactionId, never a new application.

Webhook never arrives

DETECT

No callback within the agreed window

FALLBACK

Poll the results API by transactionId and alert if the gap persists.

Same webhook arrives twice

DETECT

transactionId already processed

FALLBACK

Idempotent handler: process once, still return 200.

API slow or unavailable

DETECT

p95 latency or error-rate alert

FALLBACK

Retry with backoff, a clear "try again later" screen, escalate as P1.

Camera fails on a low-end phone

DETECT

SDK errorCode on camera start

FALLBACK

Plain-language help, then the web flow or assisted onboarding at a branch.

Security & data

Data map PROPOSED

To confirm with HyperVerge's security team and the bank's DPO.

DATA	STORED AT	KEPT FOR
ID and selfie images	HyperVerge, in-country region or on-premise	Agreed retention, then deleted
Extracted fields (name, ID number, date of birth)	Bank's core system	Bank policy
transactionId and decision	Bank server and HyperVerge	Audit period
Consent record	Bank server	As the local law requires
appId / appKey	Bank server only, never the app	Rotated on a schedule

Security review pack

Requested early, so the bank's review runs alongside the build.

- ✓ SOC 2 Type II report (under NDA)
- ✓ ISO/IEC 27001 certificate and scope
- ✓ Latest penetration test summary
- ✓ Data processing agreement (DPA)
- ✓ Data flow diagram and sub-processor list
- ✓ Impact assessment input (DPIA) and consent text in the local language

Data residency decision

In-country cloud

Faster upgrades and new models, shared operations. Fits most data-localisation rules.

On-premise

Full control for the bank. Slower upgrades, and the bank runs the infrastructure.

Chosen by the regulator's rules and the bank's policy, then recorded in the decision log.

Go-live checklist

All ten ticked before launch. Tick them off to try it.

0 / 10

- ☐ Production appld / appKey live on the server only; UAT keys removed from app builds
- ☐ accessToken generated server-side; nothing secret ships in the app
- ☐ Webhook endpoint live, authenticity check on, and it answers 200 quickly
- ☐ Duplicate and late webhooks tested; the results API fallback works
- ☐ Approval decided from the server-side result, never the HyperKycResult status
- ☐ Workflow tested on low-end Android, iOS and web, on slow networks
- ☐ Every local ID card version passes in UAT
- ☐ Consent screen approved by the bank's legal team, in the local language
- ☐ Data residency and retention configured exactly as signed off
- ☐ Dashboards, alerts and the escalation contact list ready; hypercare starts

Key best practices

INTEGRATION



Keys stay on the server

Generate the accessToken server-side. The SDK's own README calls appld/appKey in the app "less secure".



Decide on the server, never the app

Approve from the webhook or results API. The HyperKycResult status is for the screen.



Make webhooks idempotent

The same event can arrive twice. Process it once, by transaction ID.



Test in real conditions

Low-end Android, weak networks, low light, every local ID card version.

COMMUNICATION



Write it down, in both languages

Every call ends with a recap: who does what, by when.



One glossary, everywhere

Lock terms before translating. A native-speaking partner reviews.



Meet in the overlap

Short calls in shared hours to screen-share and debug. Everything else async.



Data stays where the law says

In-country region, a reference ID instead of images, agreed retention.

Regions I plan for

Vietnam

Vietnamese

IST +1:30

Indonesia

Bahasa Indonesia

IST +1:30

Philippines

Filipino, English

IST +2:30

Japan

Japanese

IST +3:30

Saudi Arabia

Arabic

IST -2:30

UAE

Arabic, English

IST -1:30

Senegal · Côte d'Ivoire

French

IST -5:30

Brazil

Portuguese

IST -8:30

Language and time difference from Bangalore. Click a region to see the demo in its language. Patterns to plan for, checked with the local partner in week one.

Prepared by **Syed Aakhib** · [LinkedIn](#)

Proposed solution for a fictional client, Sen Bank; not a record of past KYC delivery. SDK terms and status values follow HyperVerge's public Flutter SDK (hyperkyc_flutter on pub.dev); endpoint URLs and the webhook payload are placeholders. Translations are AI-drafted and not yet reviewed by a native speaker. Independent work sample styled after HyperVerge's public site; not an official HyperVerge page.