CLOUDNIX TECHNICAL REPORT · CNX-TR-2026-03

Engineering Enterprise eCommerce Platforms

Ajmal Abubacker, Goraknath, Avinash KS, Upendranath Metikala and Avinash Nishant

Cloudnix Software Labs, Bengaluru, India

CNX-TR-2026-03 · Version 1.2 · November 2026

Abstract

Enterprise commerce fails at its edges: search that slows as catalogues pass a million products, sale-day spikes that reach the database, payments marked paid on unverified callbacks, tax lines that do not reconcile, branches that cannot bill offline, unchecked marketplace edits and migrations that lose search rankings. This report describes how we engineer around each of these in a multi-tenant commerce core we have built and operated in production: request-level store resolution and scoping; catalogue and faceted search on a dedicated index with per-store caching; a single gateway contract with gateway-specific callback verification across ten gateways; an eleven-state order model; exact GST computation in the core; per-outlet inventory with pincode serviceability; an offline point-of-sale system using branch key ranges and ordered, watermarked synchronisation; field-level approval for marketplace sellers; integrations; peak readiness; security; a parallel-run migration method; and the operating model. We conclude with an anonymised case study, a roadmap and a 20-point checklist.

Keywords: *eCommerce; multi-tenancy; catalogue search; caching; payment gateways; callback verification; order lifecycle; GST; omnichannel; offline point of sale; synchronisation; marketplace; migration*

Contents

1. Executive summary
2. Where commerce platforms struggle
3. Reference architecture
4. Multi-tenancy and request routing
5. Catalogue and search at scale
6. Checkout and payment orchestration
7. Order lifecycle
8. Tax in the core
9. Inventory, outlets and serviceability
10. Offline point of sale
11. Marketplace governance
12. Integrations
13. Performance and peak readiness
14. Security
15. Migration methodology
16. Operations and service
17. Case study: a multi-branch retailer, online and offline
18. Implementation roadmap
19. Checklist: 20 questions
20. Glossary
21. References

1 Executive summary

Enterprise commerce fails at its edges: a search that slows as the catalogue grows past a million products, a sale-day spike that reaches the database, a payment marked paid on an unverified callback, a tax line that does not reconcile, a branch that cannot bill when the internet drops, a marketplace seller whose edits go live unchecked, or a migration that loses years of search rankings. This report describes how we engineer

around each of these, from a commerce core we have built and operated in production for many years: a multi-tenant platform serving brands, franchise outlets, wholesale stores and marketplaces on one codebase, an offline point-of-sale system for branches, and the integrations, operations and support model around them.

Principles.

1. **Resolve the store first.** Every request is bound to one store before anything else runs; every query, file and cache entry is scoped to it.
2. **Keep the database out of the hot path.** CDN, per-store page caches and a dedicated search index answer most traffic.
3. **Verify money before state.** An order becomes paid only after the gateway's own proof is checked against the order.
4. **Model exceptions explicitly.** Delays, rejections and returns are states, not notes.
5. **Compute tax in the core.** Every line carries its code and rates; inclusive prices are back-calculated exactly.
6. **Let branches work alone.** Offline billing with non-colliding keys and ordered synchronisation.
7. **Move businesses, not just data.** Migrations run in parallel and switch over only when proven.

2 Where commerce platforms struggle

Problem	Symptom	Pattern (chapter)
Catalogue growth	Facets and search slow down past a million products	Dedicated search index, attribute facets (5)
Sale-day traffic	Database saturates within minutes	Layered caching and CDN (13)
Many brands on one platform	Data or settings leak between stores	Store resolution and scoping (4)
Payment trust	Orders marked paid on forged or replayed callbacks	Gateway contract and verification (6)
Tax reconciliation	Paise-level mismatches, wrong IGST/CGST split	Tax in the core (8)
Branch connectivity	Billing stops when the line drops	Offline POS with key ranges (10)
Marketplace quality	Seller edits go live unchecked	Field-level approval (11)
Re-platforming	Lost orders, broken URLs, lost rankings	Parallel-run migration (15)

3 Reference architecture

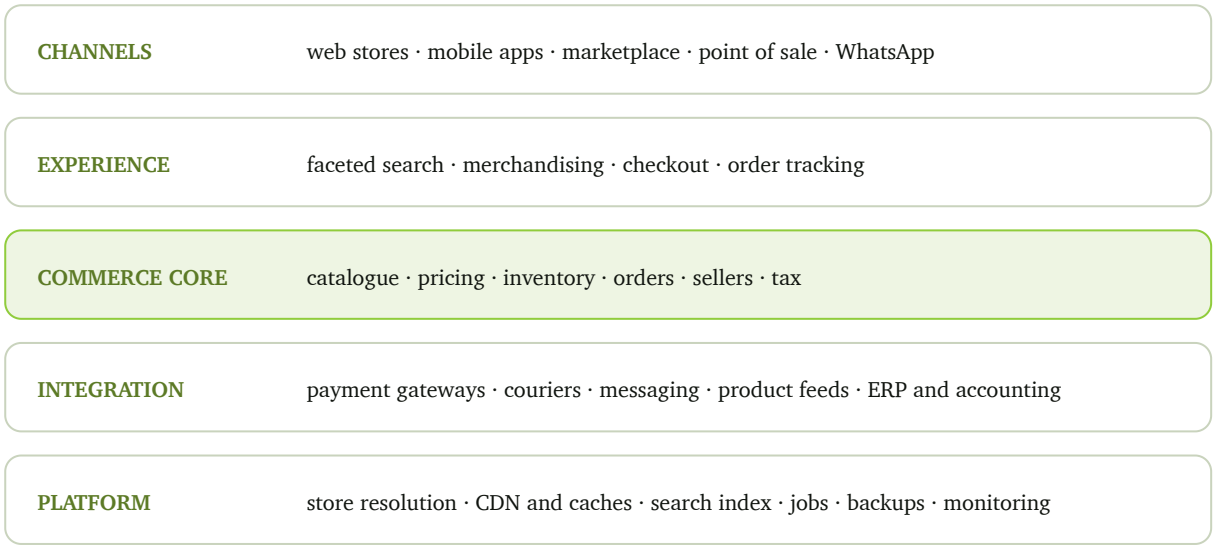


Figure 1. The five layers of the commerce platform.

3.1 *One codebase, many stores*

All stores share one codebase and one release. A front controller dispatches views, form posts and modules; an administration area follows a per-entity pattern of controller, model and data access. The relational schema of the commerce core holds around fifty tables, every one of them carrying the store's identifier. Scheduled jobs handle feeds, sitemaps, cache housekeeping, backups and synchronisation.

3.2 *Deployment options*

Option	When
Shared cloud (data in India)	Most brands; fastest to start
Dedicated cloud	Isolation, custom capacity, regional hosting
On-premise	Regulatory or internal policy requirements

4 Multi-tenancy and request routing

A multi-brand platform is only as safe as its tenant boundary. Before any business logic runs, each request passes malicious-request blocking and a request-rate check, then the host name (without `www`) is looked up against a unique domain column to find the store.

```
# every request
block_malicious(request); check_request_rate(ip, user_agent)
store = lookup_store(host_without_www)            # unique domain → store id
if not store: render("can't find the shop"); stop
context = {
  templates:  "themes/" + store.theme,
  media:      "media/" + store.domain + "/",       # per-store files
  cache:      store.domain + "/cache/",            # per-store cache namespace
  cdn:        store.media_cdn, store.theme_cdn,
  currency:   store.currency, store.decimals,
  payments:   store.gateway_credentials,
  device:     mobile | tablet | desktop | app }
```

Listing 1. Pseudocode.

4.1 *Scoping rules*

- Every query includes the store identifier; there is no query path without it.
- Files, caches and settings live in per-store namespaces.
- Payment credentials are per store, so a brand's receipts never mix with another's.
- Custom domains map one-to-one to stores; a domain can be moved without data changes.

Discipline matters. When scoping is written into each query rather than enforced by a framework, it must be checked in code review and covered by tests that try cross-store access.

5 Catalogue and search at scale

5.1 *Catalogue model*

Concept	Model
Hierarchy	Type → category → sub-category
Variants	Variants and options (size, colour, cut, pack)
Facets	Tags grouped by type, attached to products and carried into order lines
Limits	Product limits per plan, enforced at upload
Bulk	CSV and spreadsheet uploads

5.2 *Search*

For small catalogues, tag filters on the database are enough. Past a few hundred thousand products, facet counting and relevance ranking belong in a dedicated search index (an Apache Solr core in our deployments [6]) synchronised from the catalogue through a change table. The index provides typo tolerance, relevance ranking and fast facet counts; the database remains the system of record.

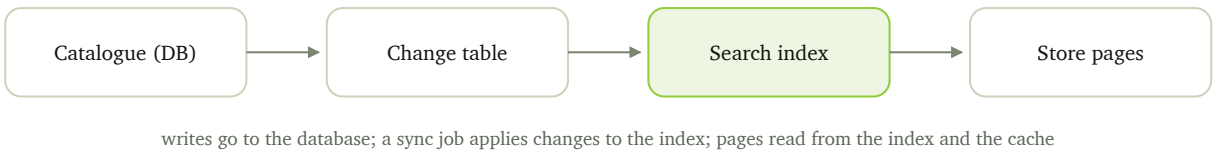


Figure 2. Catalogue to search index.

5.3 *Caching*

A file-based cache stores rendered widgets and pages per store, with a time-to-live, a per-store switch, an allow-list of modules, pattern-based invalidation and a housekeeping job. Widgets that depend on the shopper, such as recently viewed, buy again and recommendations, are never cached. Images are converted to WebP, with AVIF handled where supported [10], and served from a CDN with versioned static paths so a release invalidates nothing it does not need to [7].

5.4 *Scale reference*

The same core runs catalogues of more than ten million products for a single customer and serves more than a million shoppers on one platform.

6 Checkout and payment orchestration

Checkout moves through review, delivery details, payment, confirmation of payment and confirmation of order. Payment is delegated to gateways through a single contract: each gateway is a module that implements processing of the browser return and of the server-to-server notification. The store's configured gateway is loaded at checkout; a separate gateway can be set for international payments.

6.1 *Verification by gateway*

Gateway pattern	Proof checked
Signed webhook with timestamp	Signature header and timestamp verified before processing
Checksum header	sha256(payload + salt) with a key index, recomputed and compared
HMAC response	HMAC-SHA256 over the response with the merchant secret
Signature with order re-read	Amount and gateway order id re-read from the order (status NEW, this store only) and checked against the signature
Reverse hash	Hash of the reply fields in reverse order with the salt

Ten gateways are supported behind this contract, with webhook handlers where the gateway offers them. Card data is entered only on the gateway's hosted page and never stored [1, 2].

6.2 *Good practice*

- Process both the browser return and the webhook; whichever arrives first wins, the second is idempotent.
- Re-read the amount from the order; never trust an amount echoed by the browser.
- Restrict state changes to orders still in NEW for this store.
- Reconcile daily against gateway settlement reports.
- Offer UPI [3], cards, net banking and wallets through the gateway, and cash on delivery by pincode with an optional charge.

7 Order lifecycle

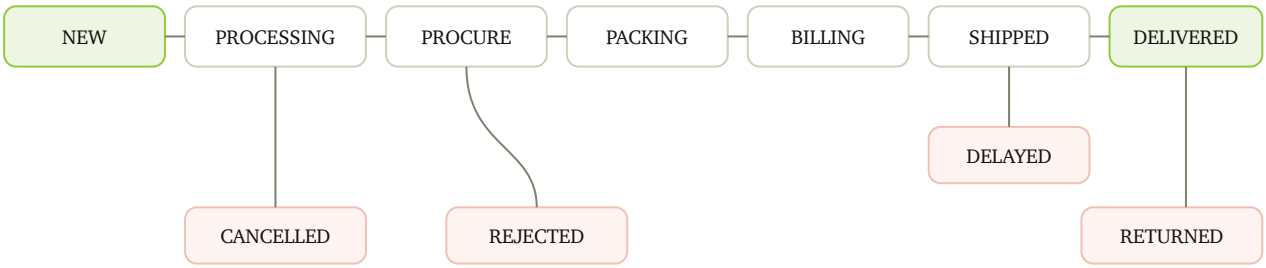


Figure 3. Eleven order states. Payment has its own track: PENDING, PAID, FAILED, REFUNDED.

Seven transitions send the customer a message: processing, shipped, delayed, delivered, cancelled, rejected and returned. Item-level order management lets one line be procured or returned independently of the rest of the order.

8 Tax in the core

Every order line carries its HSN code, CGST, SGST and IGST rates and whether the price includes tax [4, 5]. For a tax-inclusive price P at rate r:

```
base = P − P ÷ (100 + r) × r          tax = P − base
intra-state:  CGST = SGST = tax ÷ 2        inter-state:  IGST = tax
```

Listing 2. Example exchange or schema.

Example	Value
Price, inclusive, 18%	₹1,180.00
Base	₹1,000.00
Same state	CGST ₹90.00 + SGST ₹90.00
Another state	IGST ₹180.00

Invoices render in four layouts: A4 and thermal, each tax-inclusive or exclusive, generated as PDF. A tax report summarises liabilities for the accountant.

9 Inventory, outlets and serviceability

Inventory is keyed by product and outlet, holding stock level, net price and tax rate per outlet. In multi-outlet mode, each outlet (a supplier record) has service areas, and each order carries the outlet that fulfils it.

```
# routing an order in multi-outlet mode
area     = area_of(shopper.pincode)
outlets  = outlets_serving(area)                    # service areas by pincode
outlet   = first(o in outlets where stock(o, items) is sufficient)
order.outlet = outlet; invoice series and GSTIN of outlet
cod      = cod_allowed(shopper.pincode)             # maintained per pincode
```

Listing 3. Example exchange or schema.

Delivery charges come from courier zones and zone rates, with delivery slots and cash-on-delivery charges by area. The platform serves more than 19,000 pincodes.

10 Offline point of sale

Branches bill against a local database and synchronise with head office over HTTP when connected. Twenty-five tables take part, each implemented as a sync class.

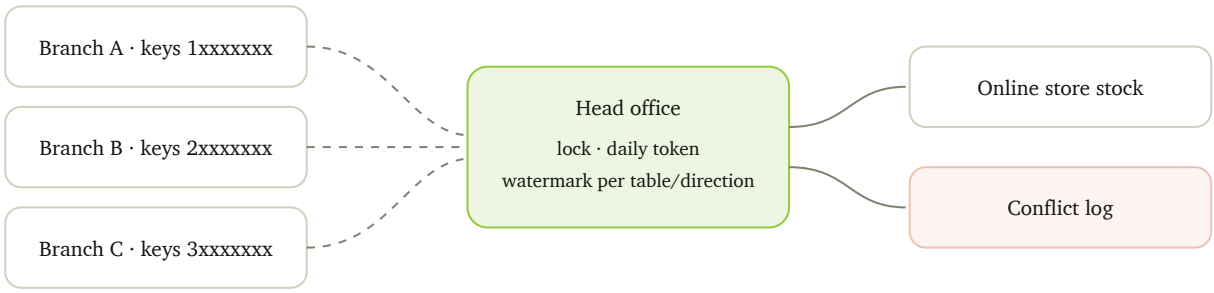


Figure 4. Offline point of sale: branch key ranges, ordered synchronisation, and a bridge to online stock.

10.1 *Synchronisation protocol*

1. Acquire the server lock for the branch; compute the day's sync token from date, branch and organisation.
2. For each table, in order: push inserts, push updates, pull inserts, pull updates, each since the watermark for that table and direction.
3. Record conflicts in a conflict table for review; never overwrite silently.
4. After pulling, move the local auto-increment past the highest server id.
5. Release the lock.

10.2 *Why key ranges*

Two branches creating invoices offline would otherwise generate the same primary keys. Each branch is initialised with its own series (for example 1xxxxxxx, 2xxxxxxx), so records from different branches never collide and can be merged without renumbering.

Branch features include shifts with day-end settlement, indents and stock requests with automatic raising, inter-branch transfers, part payments, billing by weight and per-device licensing. A bridge job pushes branch stock to the online store's inventory.

11 Marketplace governance

Sellers onboard with tax identifiers, bank account and location, and serve defined areas. Every change a seller makes is captured as a pending approval rather than applied:

```
approval(table, key, field, previous_value, new_value, mode, approved)
```

Listing 4. Example exchange or schema.

The operator approves field by field; only then is the change written to the live record. Commission is configured by product type, category or sub-category; seller subscription plans carry a trial period and a product allowance. Settlement to sellers can run inside the platform or through the operator's accounting process.

12 Integrations

Area	Integration
Product feeds	Google Merchant Center through its Content API [8]: store set-up, nightly feed generation and push, status pull, clean-up
Messaging	Email via SMTP or managed email services; per-store SMS gateways; WhatsApp through the Business Platform Cloud API [9] or providers
Couriers	API booking with an aggregator; zone and rate tables for other couriers
Systems	Open API and webhooks with a key per connected system, for ERP, accounting and CRM

13 Performance and peak readiness

Sale-day traffic arrives within minutes. The platform answers most of it before the database: a CDN with versioned static paths and modern image formats, per-store page and widget caches, and a dedicated search index. The database serves what remains: carts, checkouts and order writes.

Layer	Serves	Invalidation
CDN	images, scripts, styles, cached pages	versioned paths per release
Page and widget cache	category, product and home widgets	time-to-live, pattern purge on change
Search index	search and facets	change-table sync
Database	carts, checkout, orders	—

13.1 *Before a sale*

- Warm caches for campaign pages; pre-generate images.
- Freeze non-essential releases; keep staff on call.
- Check gateway limits and enable the fallback gateway.
- Rehearse with production-like data on staging.

14 Security

Area	Control
Edge	HTTPS everywhere; malicious-request blocking; request-rate limits logged by IP and user agent; bot checks on forms
Identity	OTP and social sign-in; per-staff logins; about 190 permissions; sessions regenerated on sign-in
Application	Prepared statements; store scoping; function-level permission checks; checkout steps enforced in sequence
Payments	Hosted gateway pages; callback verification; card data never stored [1]
Recovery tokens	Password-reset tokens encrypted and authenticated (AES-256 with HMAC-SHA256), expiring, invalidated on password change
Operations	Key-only server access; named logins; monthly access reviews; critical patches within 48 hours; daily log review
Recovery	Daily and weekly backups, a separate copy, monthly restore tests

We assess applications against the OWASP Application Security Verification Standard [11] when hardening.

15 Migration methodology

Step	Work	Exit criterion
1. Audit and plan	Catalogue, customers, orders, URLs, integrations, SEO	Signed migration plan
2. Move and check	Data moved and checked line by line; URL redirect map	Counts and samples match
3. Run in parallel	Old and new live side by side; orders compared	A full business cycle without differences
4. Switch over	DNS and redirects; team on hand	Traffic, orders and rankings stable

Sources include hosted platforms, open-source platforms and self-built systems.

16 Operations and service

- **Service levels:** a 99.9% uptime commitment, first response in 15 minutes, root-cause reports after incidents.
- **People:** a pod of seven roles, account manager, tech lead, backend and frontend developers, UI designer, QA tester and DevOps engineer, in a private channel.
- **Releases:** a staging store per customer; release only after sign-off.
- **Monitoring:** round-the-clock checks; alerts on errors, traffic spikes and failed logins.

17 Case study: a multi-branch retailer, online and offline

Anonymised. A retailer with an online store and several physical branches.

17.1 *The challenge*

Branches had unreliable internet but could not stop billing. Stock had to be consistent between branches and the online store, and head office needed one view of sales, transfers and shifts.

17.2 *What was built*

- Branch billing on a local database with its own key series, synchronising 25 tables with head office in a fixed order.
- Shifts with day-end settlement, indents with automatic raising, inter-branch transfers, part payments and billing by weight.
- A conflict log for review instead of silent overwrites.
- A bridge pushing branch stock to the online store's inventory, so online customers see what branches hold.
- Per-device licensing for billing counters.

17.3 *What it changed*

Billing continues through outages; head office sees branch activity after each synchronisation; and online availability follows what is physically on the shelf.

18 Implementation roadmap

Phase	Outcome
1. Discovery	Store network, catalogue, payments, tax and integration plan
2. Pilot	One brand or region live; migration rehearsed
3. Scale	All stores, outlets and channels; offline branches
4. Operate	Service levels, monthly reviews, peak readiness drills

19 Checklist: 20 questions

1	Is every request bound to one store before any logic runs?
2	Are cross-store access attempts covered by tests?
3	Does search run on a dedicated index past a few hundred thousand products?
4	Are personalised widgets excluded from caches?
5	Are images served in modern formats from a CDN with versioned paths?
6	Does every gateway sit behind one contract?
7	Is every callback verified with the gateway's own method?
8	Is the amount re-read from the order, not the browser?
9	Do exceptions (delay, rejection, return) have their own states?
10	Does every line carry HSN and GST rates?
11	Are inclusive prices back-calculated exactly?
12	Is stock held per outlet with service areas by pincode?
13	Can branches bill offline without key collisions?
14	Are sync conflicts logged rather than overwritten?
15	Do seller changes wait for approval?
16	Are integrations keyed per system and versioned?
17	Is there a parallel run before migration cut-over?
18	Are restores tested regularly?
19	Is there a staging store and a sign-off before release?
20	Who is on call on sale day?

20 Glossary

Store resolution	Binding a request to one store from its host name.
Facet	A filter with counts, such as size or colour.
HSN	Harmonised System of Nomenclature code for goods under GST.
CGST / SGST / IGST	Central, State and Integrated GST.
Key range	A block of identifiers reserved for one branch.
Watermark	The last point synchronised, per table and direction.
Hosted payment page	The gateway's page where card details are entered.

21 References

1. PCI Security Standards Council. *Payment Card Industry Data Security Standard*, v4.0, 2022.
2. Reserve Bank of India. *Tokenisation: Card Transactions*, circulars, 2021–2022.
3. National Payments Corporation of India. *Unified Payments Interface (UPI): Procedural Guidelines*.
4. Government of India. *The Central Goods and Services Tax Act*, 2017.
5. GSTN. *e-Invoice System: Invoice Registration Portal (IRP) Specifications*.
6. The Apache Software Foundation. *Apache Solr Reference Guide*.
7. R. Fielding, M. Nottingham and J. Reschke. *HTTP Caching*. IETF RFC 9111, 2022.
8. Google. *Content API for Shopping*, Merchant Center developer documentation.
9. Meta. *WhatsApp Business Platform: Cloud API*, developer documentation.
10. Google. *WebP image format*; Alliance for Open Media. *AV1 Image File Format (AVIF)*, 2019.
11. OWASP Foundation. *Application Security Verification Standard*, v4.0.

About this report. Prepared by the Cloudnix engineering team from systems we have designed, built and operated. Client and product names are withheld; illustrative values are marked as examples. Cloudnix Software Labs, Bengaluru, India · cloudnix.com. This report may be shared freely in its entirety.

Document control			
Report	CNX-TR-2026-03	Version	1.2
Issued	November 2026	Status	Published
File	cloudnix-enterprise-ecommerce-v1.2.pdf		