Add icon Add cover

Omni Partner Platform

Document Control

Captures version history, ownership, approvers, and distribution so changes are auditable.

	Field	Value
1	**Project**	JioOmni
2	**Doc Owner**	Ravindra Chhabra Mujtaba Rasool
3	**Version**	0.1-DRAFT
4	**Last Updated**	
5	**Approvers**	
6	**Distribution**	

1. Overview

The Partner Platform is a unified ecosystem that enables external developers, enterprises, and technology partners to seamlessly integrate with Omni. It provides the tools, workflows, and governance required to create, test, and publish AI-powered **Skills** that can be consumed by Omni users.

Requirements

2.1 Core Objectives

- **Grow Developer Ecosystem** – Provide self-serve tools for building, testing, and deploying Skills.
- **Secure by Design** – Enterprise login, CAPTCHA, 2FA, and governed approval workflows.
- **Seamless Onboarding** – Allow users to join their existing teams or create new ones.
- **Quality Assurance** – Automated validation of MCP servers/OpenAPI/Agent specs and pre-publish testing.
- **Scalable Architecture** – Modular foundation supporting phased rollout and future Agent-to-Agent integration.
- **Collaborative & Transparent** – Dashboards, analytics, and feedback loops for partners and governance teams.

2.2 Non-Functional Requirements (NFRs)

Defines service-quality targets—latency, uptime, DR, accessibility—that shape infra sizing and SLAs.

	Category	Target
1	**Latency (P95)**	
2	**Uptime (SLA)**	
3	**Throughput**	

4	**DR RPO/RTO**	
5	**Accessibility**	

2.3 KPIs

Establishes measurable success metrics (e.g., skill-success %, CSAT) used for post-launch health checks.

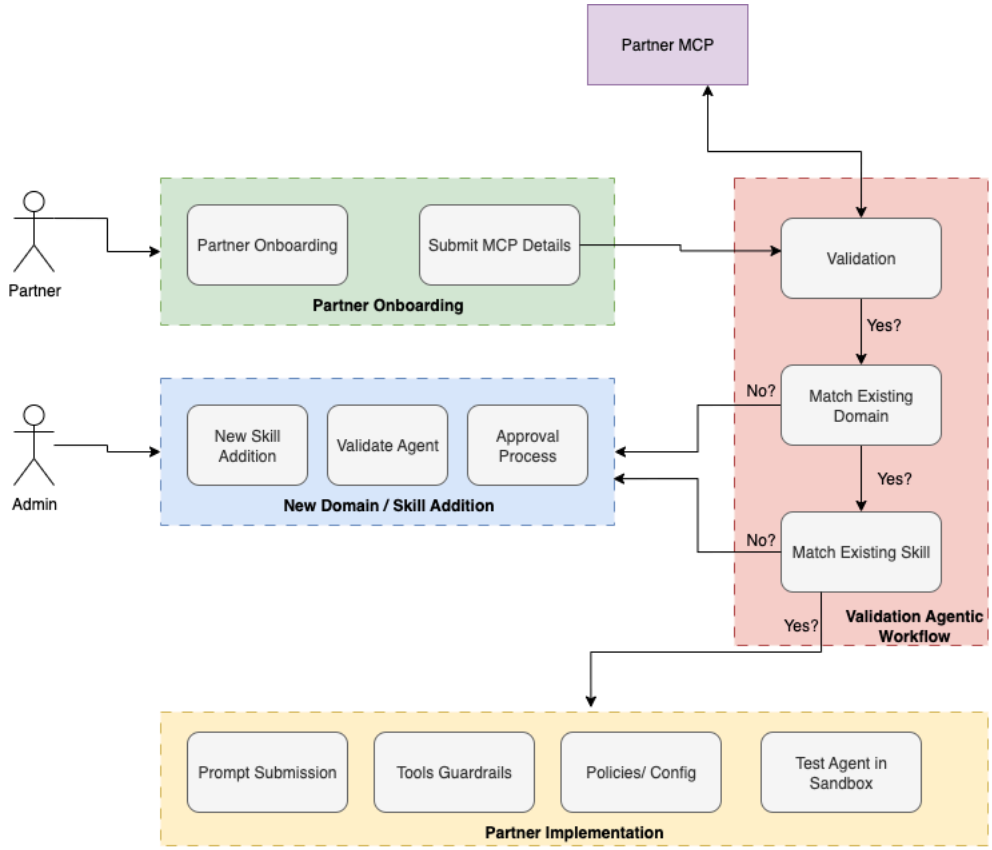
System Architecture

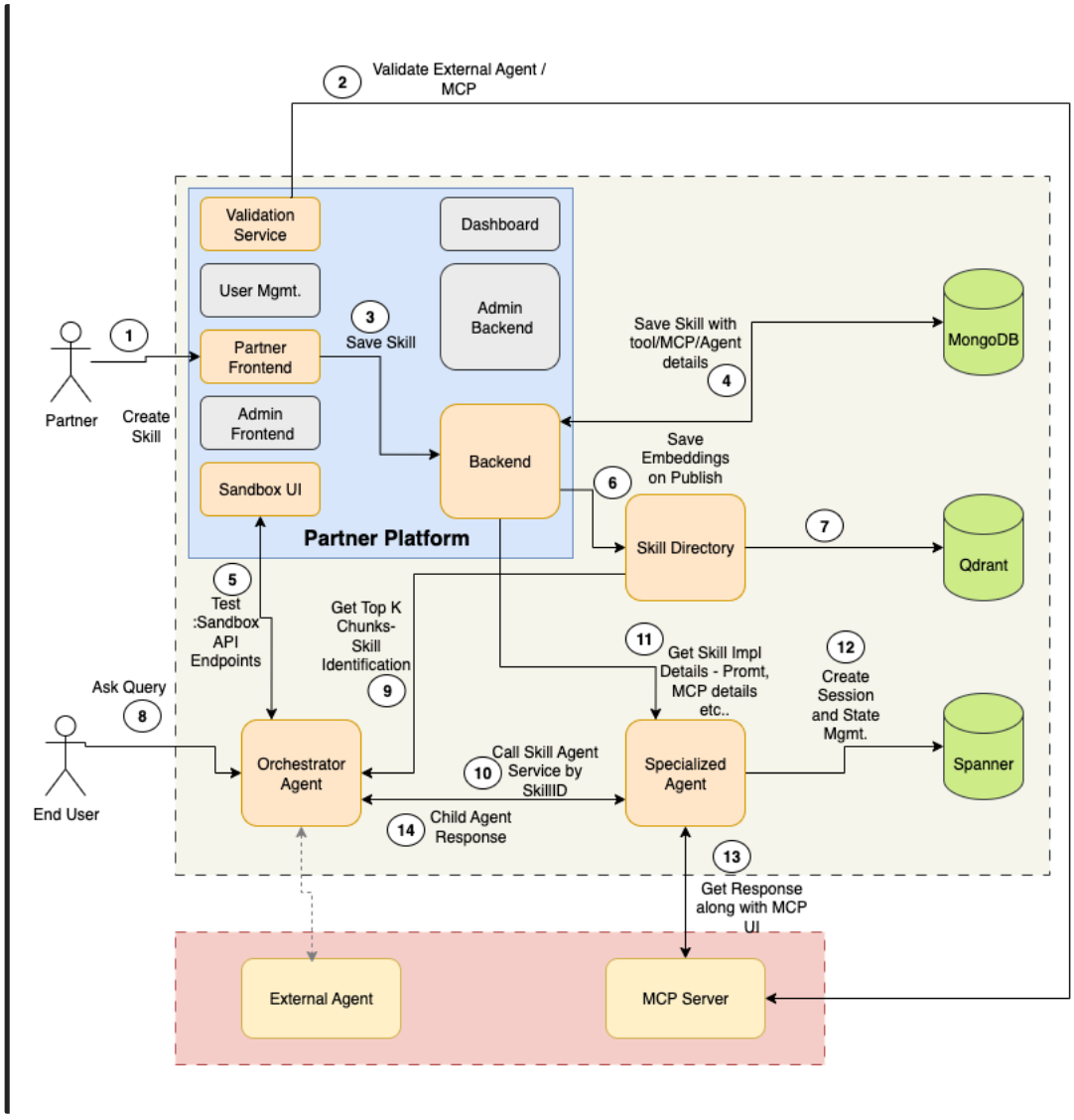
3.0 Diagram Index

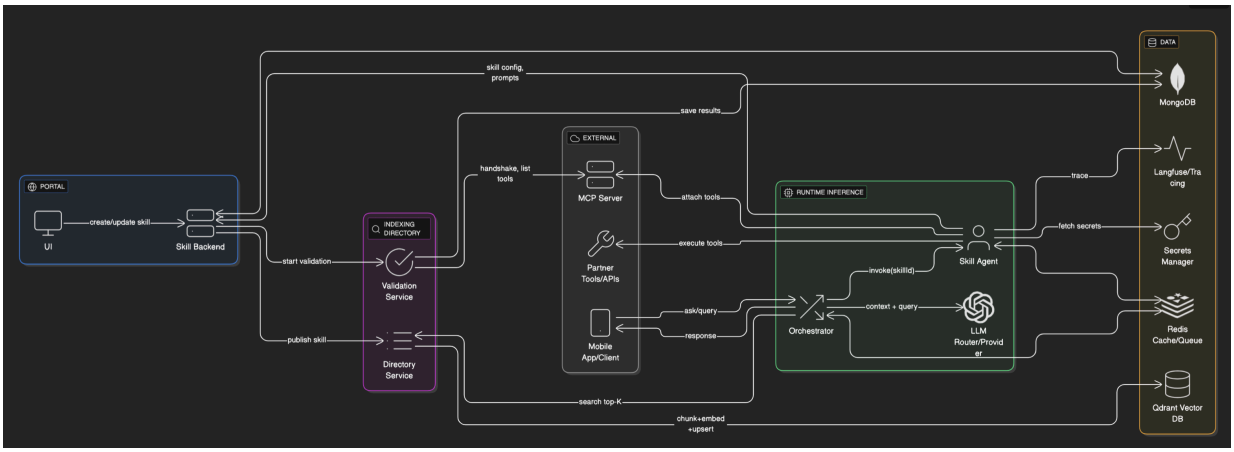
Serves as a table of embedded Loop images so reviewers can find every architecture visual quickly. (*Just put which is important to your team*)

	#	Diagram	Loop Embed / Link
1	1	High-Level Overview	
2	2	Deployment Topology	
3	3	Network Zoning	
4	4	Multi-Modal Flow	
5	5	Sequence Diagram	
6	…	…	

3.1 High-Level Overview (Diagram link)


Partner MCP
Partner
Partner Onboarding
Submit MCP Details
Partner Onboarding
Validation
Yes?
No?
Match Existing Domain
Yes?
No?
Match Existing Skill
Yes?
Validation Agentic Workflow
Admin
New Skill Addition
Validate Agent
Approval Process
New Domain / Skill Addition
Prompt Submission
Tools Guardrails
Policies/ Config
Test Agent in Sandbox
Partner Implementation


2
Validate External Agent / MCP
Validation Service
Dashboard
User Mgmt.
Admin Backend
Partner Frontend
Admin Frontend
Sandbox UI
Backend
Partner Platform
1
Partner
Create Skill
3
Save Skill
Save Skill with tool/MCP/Agent details
4
MongoDB
Save Embeddings on Publish
6
Skill Directory
7
Qdrant
5
Test :Sandbox API Endpoints
Get Top K Chunks- Skill Identification
9
11
Get Skill Impl Details - Promt, MCP details etc..
12
Create Session and State Mgmt.
Spanner
Ask Query
8
End User
Orchestrator Agent
10
Call Skill Agent Service by SkillID
Specialized Agent
14
Child Agent Response
13
Get Response along with MCP UI
External Agent
MCP Server



3.2 Component Catalog

Summarizes each subsystem's role and how it fits into the overall architecture.

	Area	Component	Summary
1	Core	**Supervisor/Orchestrator Service**	Front-door for user queries (SSE/Socket.IO/HTTP). Retrieves top-K skills from Skill Directory, performs LLM routing, invokes Skill Agent Service, streams responses. Enforces retries/fallbacks and budgets.
2	Core	**Skill Agent Service**	Stateless runtime that materializes an Agent per request using skill prompts + binding. Attaches MCP tools, runs plan→act loop, calls external tools, streams results back to Orchestrator.
3	Core	**LLM Router/Provider**	Chooses best skill from retrieved candidates using query + snippets + policies; supports model routing and temperature/budget controls.
4	Core	**Memory/Session Layer**	Conversation/session context store (Spanner or Redis) with user context and tool results cache; optional long-term memory/personalization store.
5	Registries	**Skill Directory Service (SDS)**	Unified indexing & search: chunks skill text, generates embeddings, upserts vectors to Qdrant keyed by skillId/versionId; exposes `/index` & `/search`; applies tenant/visibility filters and health-based boosting.

6	Registries	**Tools/MCP Registry**	Catalog of internal MCP endpoints (baseUrl, transport, protocolVersion, capabilities, health) discovered via Validation Service; referenced by bindings.
7	Partner Portal	**Portal Backend**	Canonical skill metadata in MongoDB (prompts, versions, policies, bindings list); read API for runtime.
8	Validation & Indexing	**MCP Validation Service**	Performs handshake/auth checks, lists tools, captures schemas (and optional `x-*` policy hints), runs smoke tests, writes `validation_runs` + health signals.
9	Data & Storage	**MongoDB**	Source of truth for skills, versions, MCP endpoints, bindings, validation runs, audit trails.
10	Data & Storage	**Qdrant Vector DB**	`skills_v1` collection for retrieval; metadata filters on tenant/visibility/version/state; HNSW tuned for recall/latency.
11	Data & Storage	**Google Secrets Manager/ Redis**	Stores all partner credentials; app holds only `secret_ref`. Short-lived tokens fetched at runtime.
12	Data & Storage	**Blob Storage (GCS/Azure Blob)**	Stores large OpenAPI specs/manifests; referenced by `specRefs`.
13	Security	**Policy/Guardrails**	Prompt linting, tool allowlists, PII redaction; enforced in SAS and at response boundary.
14	Observability	**Langfuse**	Tracing of LLM/tool events with conversation_id/request_id; links validation and runtime spans.
15	Observability	**Log**	System metrics/logs/traces to Databricks SLOs & alerts (p95 search, MCP health, publish pipeline).
16	Messaging & Events	**Redis (Cache/Streams)**	Low-latency cache for runtime skill reads and SDS candidate cache; ephemeral queues and Socket.IO adapter.
17	Client & UX	**Partner Portal (Web UI)**	Skill creation/edit, MCP registration, validation runs, publish/rollback, versioning, RBAC; shows discovered tools & health.

18	Client & UX	**Client Web SDK (Sandbox)**	Sends ASK events, receives streamed responses; carries user/session context and consent flags.
19	Governance	**Audit & Approvals**	Immutable logs of changes (who/what/when), approval workflow for publish, policy exceptions with expiry.

2.1 Partner Web App (Portal UI)

- Create/edit **Skill** (name, tags, domain, visibility, owners).
- Define **prompts** (system/instructions, tool-use guidelines, guardrails), **descriptions**, **test cases**.
- Register **MCP server(s)** (base URL, auth type, headers/keys referenced from Secrets Manager, not raw secrets in browser).
- Trigger **Validate** → displays discovered tools and validation status with logs.
- **Publish** workflow with version notes and rollout (draft → published → deprecated).
- Role-based access control (Partner Owner, Editor, Viewer).

2.2 Portal Backend Service (PBE)

- REST API for Skills, MCP endpoints, versions, validation runs.
- Persist canonical skill models in MongoDB.
- Starts validation jobs (enqueue) and embedding/index jobs.
- On **publish**, calls to **Skill Directory Service**.
- Exposes read API for **Skill Agent Service** (runtime fetch of skill details).

2.3 MCP Validation Service

- Performs connectivity handshake to partner MCP endpoints.
- Discovers tools, pulls tool schemas, runs **smoke tests** (optional sample tool calls with partner-provided test inputs).
- Caches tool list & availability, stores **validation reports** (status, errors, latency stats) in MongoDB.
- Periodic re-validation (cron/Temporal) & health signals for routing.

2.4 Skill Directory Service (SDS — Index & Search)

- Single service responsible for **chunking text**, **generating embeddings**, and **upserting** vectors to Qdrant against `skillId`/`versionId`.
- Exposes `POST /index` for publish/reindex (idempotent), and `POST /search` for runtime retrieval.
- Ranking (hybrid optional): cosine score + availability/health boost (+ optional BM25/text filters).
- Enforces tenant/visibility filters and policy gating (only `published`).
- **Index Map (Mongo)**: maintains mapping of `skillId + versionId → pointIds` for deterministic deletes/rollbacks and audit.

  Indexes: `(tenantId, skillId, versionId)` unique; `active`; `batchId`.

```VisualBasic
//Sample--
// collection: sds_index_map
{
"_id": "ObjectId",
"tenantId": "tenant-jio",
"skillId": "skill-astro-...",
"versionId": "sv-astro-1-0-0", // or semver "1.0.0"
"collection": "skills_v1",
"pointIds": [1200101, 1200102, 1200103], // Qdrant point IDs (int or UUID—pick one and standardize)
"numPoints": 3,
```

```
"model": "text-embedding-3-large",
"chunking": { "algo": "recursive", "maxTokens": 512, "overlap": 64 },
"checksum": { "textHash": "sha256:...", "configHash": "sha256:..." },
"batchId": "idx-2025-09-04-astro-0001",
"active": true, // set false on rollback/delete
"createdAt": "2025-09-04T10:30:00Z",
"updatedAt": "2025-09-04T10:31:00Z",
"indexedBy": "svc-sds"
}
```

- **Why this mapping?**
- Guarantees **precise deletes** even if filters or metadata evolve.
- Enables **rollback** to a previous version by knowing exactly which point IDs belong to each version.
- Supports **auditing** (how many vectors, when, which model/chunking).

Clean delete/rollback options

- **By IDs (preferred when map exists):** `qdrant.points.delete(ids = pointIds)`.
- **By filter (when no map):** filter on `skill_id` + `version_id` payload keys.

API Contracts (concise) : WS **Wagisha Singh** **can add for Skill Directory**

2.6 Orchestrator Service (Supervisor)

- Receives `ASK` from clients (Socket.IO/SSE/HTTP), manages conversation context.
- Calls **Skill Directory Service** for top-K candidate skills.
- LLM-based **routing**: feeds retrieved skill snippets + user query + policy constraints to decide target skill.
- Calls **Skill Agent Service** with `skill_id` (and optionally version pinning), passes condensed query & user context (optional).
- Streams partial/full responses to client; handles retries/fallbacks.

2.7 Skill Agent Service (SAS)

- Stateless runtime that **materializes an Agent on demand**:
  - Fetches skill prompts/config from PBE- Portal backend.
  - Fetches secrets (e.g., auth tokens) from Google Secret manager by secret refs only.
  - Connects to MCP server(s), registers exposed tools as function/tool adapters.
  - Runs the agent loop (planning → tool calls → synthesis), with timeouts, rate limits, guardrails.
  - Streams tokens/results back to Orchestrator.
- Traces all LLM/tool events to **Langfuse**.
- Enforces per-partner quotas, concurrency, and per-tool policy.

2.8 Data Layer

- **MongoDB**: canonical skill data, versions, MCP endpoints, validation runs, publication history, policy flags, audit logs.
- **Qdrant**: vector embeddings for retrieval; collection `skills_v1` with metadata for tenant/skill/version/state/tags.
- **VAULT**: all secrets (MCP auth, API keys). Backend stores only references.
- **Redis**: low-latency caches (skill read cache), streaming buffers, and queues if needed.

2.9 Observability & Safety

- **Langfuse** for tracing of LLM & tool calls (link with conversation_id, request_id).
- Safety/guardrails: prompt linting, allow/deny tool list per skill, response redaction.

3) Key Sequences

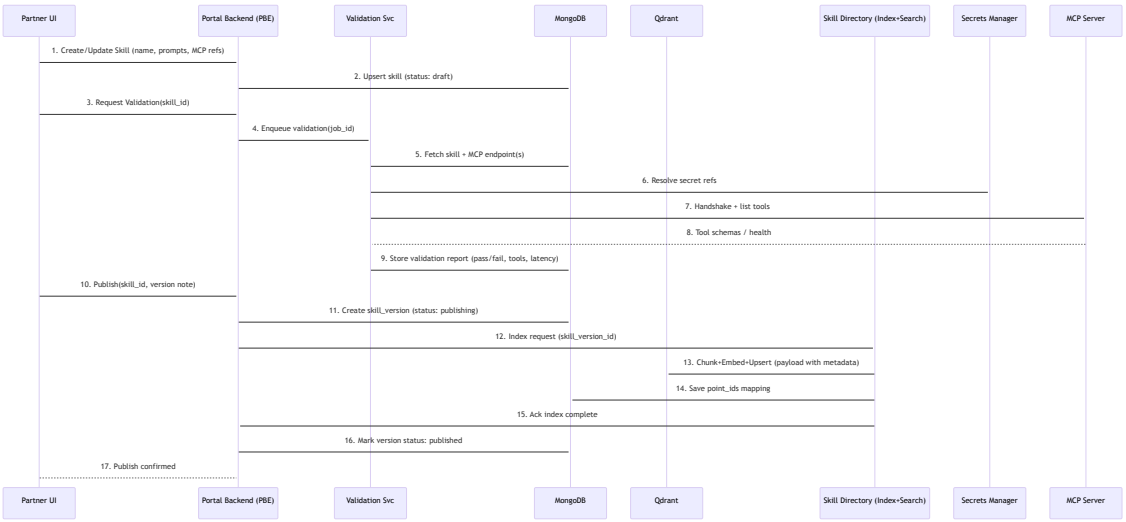
3.1 Partner: Create → Validate → Publish Skill

Mermaid

```
sequenceDiagram
participant U as Partner UI
participant B as Portal Backend (PBE)
participant V as Validation Svc
participant DB as MongoDB
participant Q as Qdrant
participant S as Skill Directory (Index+Search)
participant Va as Secrets Manager
participant M as MCP Server

U->>B: 1. Create/Update Skill (name, prompts, MCP refs)
B->>DB: 2. Upsert skill (status: draft)
U->>B: 3. Request Validation(skill_id)
B->>V: 4. Enqueue validation(job_id)
V->>DB: 5. Fetch skill + MCP endpoint(s)
V->>Va: 6. Resolve secret refs
V->>M: 7. Handshake + list tools
M-->>V: 8. Tool schemas / health
V->>DB: 9. Store validation report (pass/fail, tools, latency)
U->>B: 10. Publish(skill_id, version note)
B->>DB: 11. Create skill_version (status: publishing)
B->>S: 12. Index request (skill_version_id)
S->>Q: 13. Chunk+Embed+Upsert (payload with metadata)
S->>DB: 14. Save point_ids mapping
S->>B: 15. Ack index complete
B->>DB: 16. Mark version status: published
U-->>B: 17. Publish confirmed


```

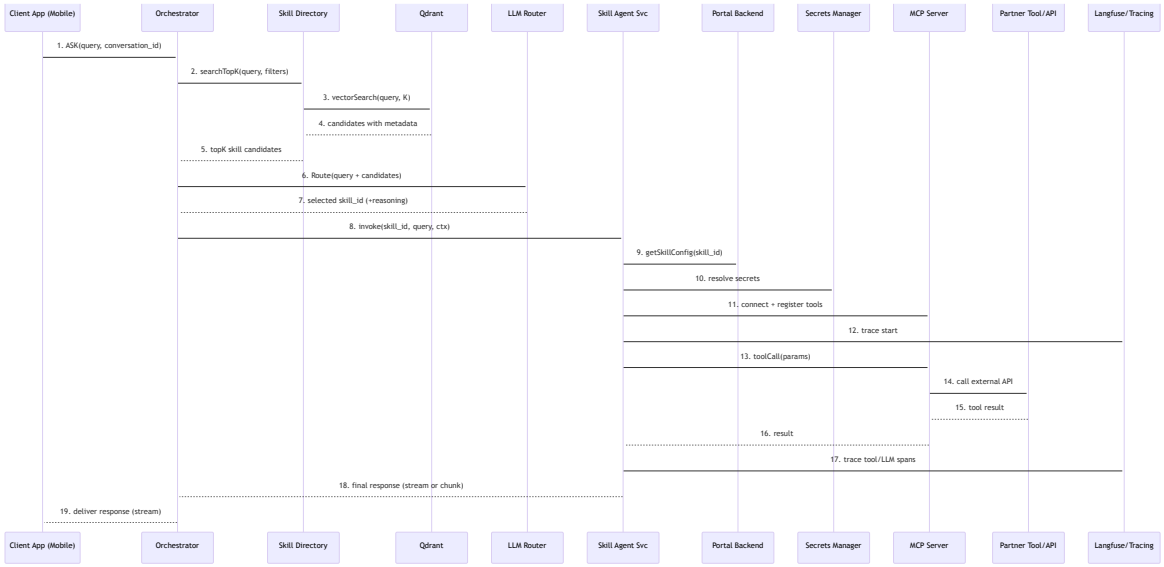


3.2 End-User: Query → Route → Invoke Skill → Tools → Answer

Mermaid

```
sequenceDiagram
participant C as Client App (Mobile)
participant O as Orchestrator
participant S as Skill Directory
participant Q as Qdrant
participant L as LLM Router
participant A as Skill Agent Svc
participant B as Portal Backend
participant V as Secrets Manager
participant M as MCP Server
participant T as Partner Tool/API
participant F as Langfuse/Tracing
```

```
C->>O: 1. ASK(query, conversation_id)
O->>S: 2. searchTopK(query, filters)
S->>Q: 3. vectorSearch(query, K)
Q-->>S: 4. candidates with metadata
S-->>O: 5. topK skill candidates
O->>L: 6. Route(query + candidates)
L-->>O: 7. selected skill_id (+reasoning)
O->>A: 8. invoke(skill_id, query, ctx)
A->>B: 9. getSkillConfig(skill_id)
A->>V: 10. resolve secrets
A->>M: 11. connect + register tools
A->>F: 12. trace start
A->>M: 13. toolCall(params)
M->>T: 14. call external API
T-->>M: 15. tool result
M-->>A: 16. result
A->>F: 17. trace tool/LLM spans
A-->>O: 18. final response (stream or chunk)
O-->>C: 19. deliver response (stream)
```

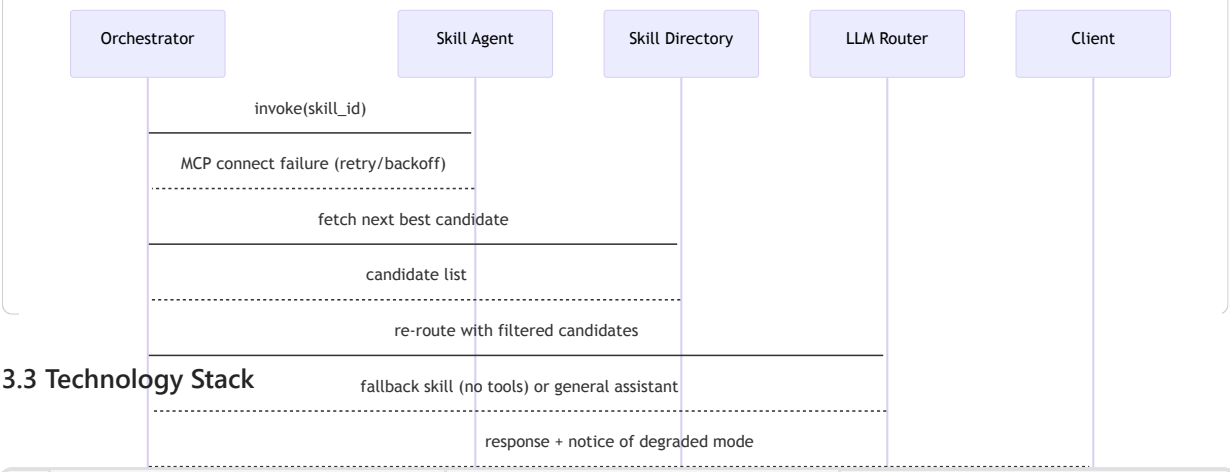


3.3 [WIP] Failure: MCP Unavailable → Graceful Degradation (To be Explored more)

Mermaid

```
sequenceDiagram
participant O as Orchestrator
participant A as Skill Agent
participant S as Skill Directory
participant L as LLM Router

O->>A: invoke(skill_id)
A-->>O: MCP connect failure (retry/backoff)
O->>S: fetch next best candidate
S-->>O: candidate list
O->>L: re-route with filtered candidates
L-->>O: fallback skill (no tools) or general assistant
O-->>Client: response + notice of degraded mode
```



3.3 Technology Stack

	Layer	Technology	Rationale
1	Validation Service	**Go/Java**	Efficient concurrent network probes (handshake, list_tools, smoke), small binaries, predictable latency.
2	Cache / Realtime	**Redis** (cache + pub/sub/streams)	Low-latency bootstrap caches, SDS candidate cache, SSE/WebSocket adapters.
3	Messaging / Events	*GCP Pub/Sub*	Decouples validation/indexing/publish pipelines; durable fan-out and replay for audits.
4	Secrets	**Google Secret Manager**	Store only `secret_ref` (no plaintext); rotation, audit trails.
5	LLM Provider Interface	**Vendor-agnostic HTTP adapter (LiteLLM) - Router**	Swap models/providers behind a stable interface; record tokens/latency.
6	Observability — Tracing	**Langfuse**	End-to-end traces for LLM/tool spans; correlate validation ↔️ runtime.
7	Observability — Metrics/Logs	**Google Logs**	p95 search latency, MCP health, publish pipeline SLOs; centralized logs.
8	Frontend — Partner Portal	**React + TypeScript (Next.js)**	Rich admin UX, SSR performance, component library to render MCP-UI (card/table/form).
9	Frontend — Mobile	*React Native*	SSE/WebSocket streaming, consent & policy UI, offline caches.

3.4 Reference Deployment Topology

Shows region distribution, cluster layout, and node pools for dev, staging, prod.

> *Diagram / table of regions, VNETs, node pools*

3.5 Network & Security

Details ingress/egress, WAF, service-mesh, mTLS, and subnet isolation for zero-trust.

- Ingress paths, WAF, mTLS
- Zero-trust zones diagram

API and Data Models

Data Model :

1. skill_implementations

GraphQL

```
{
  "_id": { "$oid": "66f101a0a1b2c30100000201" },
  "skillId": "skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f",

  "tenantId": "tenant-jio", // NEW for multi-tenant isolation
  "userId": "ravindra.singh@ril.com",

  "skillName": "Astrology",
  "skillDescription": "Provides daily/weekly horoscopes, detailed birth charts (Kundali), and relationship compatibility analysis.",

  "partnerName": "AstroVision",
  "partnerDescription": "AstroVision offers accurate astrology computations and curated interpretations.",

  "vectorSkillId": "vec-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f",

  "prompts": { // NEW: agent construction at runtime
    "system": "You are an expert Vedic astrology assistant. Be precise but empathetic. Explain results in simple language and avoid deterministic predictions.",
    "toolUseGuidelines": "If birth details are provided use get_birth_chart; otherwise get_horoscope. For relationship queries call match_compatibility.",
    "safety": "Never output exact birth time publicly; mask to nearest 5 minutes unless user has consent."
  },

  "tags": ["astrology","horoscope","kundali","compatibility"],  // NEW: improves SDS retrieval & filters

  "policy": { // NEW
    "visibility": "public",
    "rate_limits": { "rpm": 120 }
  },

  "mcpBindings": ["bind-astro-primary-v1"], // NEW: binds this skill to an MCP endpoint via bindings

  "specRefs": { // NEW: move big specs out of the doc
    "openapi": {
      "blobRef": "gs://partner-specs/astrovision/openapi@sha256:cafebabe1234",
      "hash": "sha256:cafebabe1234",
      "model": "3.0.1"
    }
  },

  "createdBy": "ravindra.singh@ril.com",
  "updatedBy": "ravindra.singh@ril.com",
  "createdAt": { "$date": "2025-07-06T09:24:00Z" },
  "updatedAt": { "$date": "2025-07-06T09:30:00Z" },

  "status": "APPROVED",
  "markForApproval": false,
  "version": 3,
  "approvalRemarks": "",
  "_class": "com.rjil.adminPortal.model.SkillImplementation"
```

```
}
```

	Field	Type	Example / Allowed Values	Description & Significance
1	`_id`	ObjectId	`66f101a0a1b2c30100000201`	Mongo primary key.
2	`skillId`	string	`skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f`	Stable identifier used across services and in retrieval payloads.
3	`tenantId`	string	`tenant-jio`	Multi-tenant isolation; included in all reads/search filters.
4	`userId`	string (email/ID)	`ravindra.singh@ril.com`	Last editor/owner for audit.
5	`skillName`	string	`Astrology`	Display name; used in directory UI & text search.
6	`skillDescription`	string	`Provides daily/weekly horoscopes…`	Embedding source + directory description.
7	`partnerName`	string	`AstroVision`	Partner brand shown in portal.
8	`partnerDescription`	string	`AstroVision offers accurate…`	Longer partner bio for UI/context.
9	`vectorSkillId`	string	`vec-astro-7c9e-…`	Logical grouping for embeddings (useful for reindex & cleanup).
10	`prompts.system`	string	`You are an expert Vedic astrology assistant…`	Base system prompt to construct the agent.
11	`prompts.toolUseGuidelines`	string	`If birth details are provided use get_birth_chart…`	Sequencing/selection hints that improve tool reliability.
12	`prompts.safety`	string	`Never output exact birth time publicly…`	Guardrails, masking, and policy reminders.

13	`tags`	array<string>	`["astrology","horoscope","kundali","compatibility"]`	Improves Skill Directory filtering & ranking.
14	`policy.visibility`	enum	`public` | `tenant` | `private`	Controls who can discover/use the skill via search.
15	`policy.rate_limits.rpm`	integer	`120`	Soft per-skill throttle hint; can inform routing/backpressure.
16	`mcpBindings`	array<string>	`["bind-astro-primary-v1"]`	References into **`skill_mcp_bindings`** (endpoint, per-tool policies, runtime budgets).
17	`specRefs.openapi.blobRef`	string (URI)	`gs://partner-specs/astrovision/openapi@sha256:cafebabe1234`	Pointer to large OpenAPI spec stored out-of-band.
18	`specRefs.openapi.hash`	string	`sha256:cafebabe1234`	Integrity check for the referenced spec blob.
19	`specRefs.openapi.model`	string	`3.0.1`	OpenAPI version of the referenced spec.
20	`createdBy`	string (email/ID)	`ravindra.singh@ril.com`	Audit field.
21	`updatedBy`	string (email/ID)	`ravindra.singh@ril.com`	Audit field.
22	`createdAt`	ISO datetime	`2025-07-06T09:24:00Z`	Audit field.
23	`updatedAt`	ISO datetime	`2025-07-06T09:30:00Z`	Audit field.
24	`status`	enum	`DRAFT` | `PENDING_APPROVAL` | `APPROVED` | `REJECTED` *(example shown:* `APPROVED`*)*	Workflow state; used to gate publish and search exposure.
25	`markForApproval`	boolean	`false`	Signals item needs reviewer attention.

26	`version`	integer	`3`	Incremented on updates; helpful for review diffs.
27	`approvalRemarks`	string	`""`	Approver/reviewer comments.
28	`_class`	string *(optional)*	`com.rjil.adminPortal.model.SkillImplementation`	Framework/ORM class name (legacy/optional).

2. mcp_endpoints

GraphQL

```
{
  "_id": { "$oid": "66f101a0a1b2c30100000101" },
  "endpointId": "mcp-astrovision-primary",

  "tenantId": "tenant-jio",
  "partnerId": "astrovision",

  "name": "AstroVision MCP",
  "baseUrl": "wss://mcp.api.astrovision.com/mcp",
  "transport": "ws",
  "protocolVersion": "mcp-1.0",
  "serverVersion": "1.0.0",

  "auth": {
    "type": "bearer",
    "secret_ref": "secrets/partners/astrovision/mcp_token"   // never store raw secret
  },

  "capabilities": {
    "tools": true,
    "resources": false,
    "prompts": false,
    "ui": {                                  // NEW (optional MCP UI support)
      "supported": true,
      "spec": "mcp-ui-0.1",
      "components": ["card","table","form"],
      "maxPayloadKb": 64
    }
  },

  "discoveredTools": [                       // captured by Validation Service
    { "name": "get_horoscope", "schemaHash": "sha256:5f2c..." },
    { "name": "get_birth_chart", "schemaHash": "sha256:ab44..." },
    { "name": "match_compatibility", "schemaHash": "sha256:d1e9..." }
  ],
  "discoveredUi": [
    { "component": "card", "schemaHash": "sha256:1111...", "lastSeenAt": "2025-09-05T06:10:00Z" },
    { "component": "form", "schemaHash": "sha256:2222..." }
  ],

  "health": {
    "status": "up",
    "avgLatencyMs": 210,
    "lastChecked": "2025-07-06T09:35:00Z",
    "uptime7dPct": 99.7
  },

  "createdAt": "2025-07-06T09:20:00Z",
  "updatedAt": "2025-07-06T09:35:00Z"
}
```

	Field	Type	Example / Allowed Values	Description & Significance
1	`_id`	ObjectId	`66f101a0a1b2c301000 00101`	Mongo primary key. **[System-managed]**
2	`endpointId`	string	`mcp-astrovision-primary`	Stable endpoint identifier referenced by bindings (`endpointRef`). **[System-managed]** (generated), visible in portal.
3	`tenantId`	string	`tenant-jio`	Multi-tenant isolation; requests must match tenant. **[Partner input]** (or prefilled by portal context).
4	`partnerId`	string	`astrovision`	Groups endpoints by partner for RBAC, reporting, quotas. **[Partner input]** (selected from partner org).
5	`name`	string	`AstroVision MCP`	Human-readable label for ops/portal. **[Partner input]**
6	`baseUrl`	URL	`wss://mcp.api.astro vision.com/mcp`	MCP connection address (ws/http). **Critical for dialing. [Partner input]**
7	`transport`	enum	`ws` | `http` | `stdio`	Selects MCP client/transport. **[Partner input]**
8	`protocolVersion`	string	`mcp-1.0`	MCP protocol claimed by server; we verify on handshake. **[Partner input]**, **validated [Auto-discovered]**
9	`serverVersion`	string	`1.0.0`	Vendor build/semver shown in ops. **[Partner input]** (optional), may be **confirmed [Auto-discovered]**
10	`auth.type`	enum	`bearer` | `oauth2` | `mtls`	Auth scheme to use. **[Partner input]**

11	`auth.secret_ref`	string	`secrets/partners/astrovision/mcp_token`	Reference to token/secret in Vault/KMS. We never store plaintext. **[Partner input]** (captured via secure form; stored as ref).
12	`auth.oauth2.token_url`	URL	`https://idp.example.com/oauth/token`	OAuth2 token endpoint (if `oauth2`). **[Partner input]**
13	`auth.oauth2.client_id_ref`	string	`secrets/.../client_id`	Secret ref for OAuth2 client id. **[Partner input]**
14	`auth.oauth2.client_secret_ref`	string	`secrets/.../client_secret`	Secret ref for OAuth2 client secret. **[Partner input]**
15	`auth.oauth2.scopes`	array<string>	`["mcp.read","mcp.write"]`	Requested OAuth scopes. **[Partner input]**
16	`capabilities.tools`	boolean	`true`	Server implements MCP tools. **[Partner input]** (declared) and **verified [Auto-discovered]** via `list_tools`.
17	`capabilities.resources`	boolean	`false`	Exposes MCP resource catalog. **[Partner input]** (optional) and **verified [Auto-discovered]**
18	`capabilities.prompts`	boolean	`false`	Exposes MCP prompts catalog. **[Partner input]** (optional) and **verified [Auto-discovered]**
19	`capabilities.ui.supported`	boolean	`true`	Server can emit UI frames (MCP-UI extension). **[Partner input]** (optional) and **probed [Auto-discovered]**
20	`capabilities.ui.spec`	string	`mcp-ui-0.1`	UI spec version the client understands. **[Partner input]** (optional)
21	`capabilities.ui.components`	array<enum>	`["card","table","form"]`	Component types the server may emit. **[Partner input]** (optional), **validated [Auto-discovered]**
22	`capabilities.ui.maxPayloadKb`	integer	`64`	Payload cap to enforce client-side. **[Partner input]** (optional, policy hint)

23	`discoveredTools[].name`	string	`get_horoscope`	Tool name from `list_tools`. Source of truth for available tools. **[Auto-discovered]** (Validation writes it).
24	`discoveredTools[].schemaHash`	string	`sha256:5f2c...`	Hash of input schema for drift detection. **[Auto-discovered]**
25	`discoveredUi[].component`	enum	`card` | `table` | `form`	UI component observed during validation. **[Auto-discovered]**
26	`discoveredUi[].schemaHash`	string	`sha256:1111...`	UI schema hash for drift/change detection. **[Auto-discovered]**
27	`discoveredUi[].lastSeenAt`	ISO datetime	`2025-09-05T06:10:00Z`	Last time this UI component was seen. **[System-managed]** (set by Validation Service).
28	`health.status`	enum	`up` | `degraded` | `down`	Latest health from probe; used for routing/boosts. **[Auto-discovered]** and **[System-managed]**
29	`health.avgLatencyMs`	number	`210`	Moving average latency for endpoint. **[Auto-discovered]** and **[System-managed]**
30	`health.lastChecked`	ISO datetime	`2025-07-06T09:35:00Z`	Freshness of health signal. **[System-managed]**
31	`health.uptime7dPct`	number	`99.7`	7-day availability; can boost ranking. **[System-managed]**
32	`createdAt`	ISO datetime	`2025-07-06T09:20:00Z`	Audit field. **[System-managed]**
33	`updatedAt`	ISO datetime	`2025-07-06T09:35:00Z`	Audit field. **[System-managed]**

3. `skill_mcp_bindings` (how the skill uses the endpoint: policies + runtime + UI)

GraphQL

```
{
  "_id": { "$oid": "66f101a0a1b2c30100000301" },
  "bindingId": "bind-astro-primary-v1",

  "tenantId": "tenant-jio",
  "skillId": "skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f",
  "skillVersion": "1.0.0",                    // pin to the prompts/policies set

```

```
  "endpointRef": "mcp-astrovision-primary",

  "toolsAllowlist": ["get_horoscope","get_birth_chart","match_compatibility"],

  "toolPolicies": {
    "get_horoscope": {
      "timeoutMs": 4000,
      "rateLimit": { "rpm": 200 },
      "paramDefaults": { "timezone": "Asia/Kolkata", "lang": "en" },
      "paramRequirements": ["sign","period"],
      "privacy": { "maskBirthTimeToNearestMinutes": 5 },
      "source": {                              // optional provenance metadata
        "timeoutMs": "server:x-timeout-ms",
        "rateLimit": "server:x-rate-limit-rpm",
        "paramDefaults": "schema.properties.*.default",
        "paramRequirements": "schema.required"
      },
      "lastSyncedAt": "2025-07-06T09:33:00Z"
    },
    "get_birth_chart": {
      "timeoutMs": 8000,
      "rateLimit": { "rpm": 120 },
      "paramRequirements": ["dob","tob","lat","lon"]
    },
    "match_compatibility": {
      "timeoutMs": 7000,
      "rateLimit": { "rpm": 100 },
      "paramDefaults": { "model": "vedic" }
    }
  },

  "runtime": {                                 // agent runtime budgets for this binding
    "maxSteps": 4,
    "maxTokens": 2048,
    "temperature": 0.1
  },

  "uiAllowlist": ["card","table","form"],      // optional, if MCP UI is enabled
  "uiPolicies": {
    "maxInteractiveSteps": 3,
    "allowExternalLinks": false,
    "sanitizeHtml": true,
    "maxPayloadKb": 64
  },

  "validation": {
    "status": "passed",
    "lastRunId": "val-astro-2025-07-06-001",
    "latencyP95Ms": 620
  },

  "createdAt": "2025-07-06T09:26:00Z",
  "updatedAt": "2025-07-06T09:34:00Z"
}
```

	Field	Type	Example / Allowed Values	Description & Significance
1	`_id`	ObjectId	`66f101a0a1b2c301000 00301`	Mongo primary key. **[System-managed]**
2	`bindingId`	string	`bind-astro-primary-v1`	Stable join key referenced by skills and bootstrap API. **[System-managed]** (generated)

3	`tenantId`	string	`tenant-jio`	Tenant isolation for runtime reads and edits. **[Partner input]** (or set by portal context)
4	`skillId`	string	`skill-astro-7c9e-…`	Parent skill this binding belongs to. **[System-managed]** (derived when creating binding)
5	`skillVersion`	string	`1.0.0`	Pins prompt/policy snapshot used at runtime. **[Partner input]** (or selected during publish)
6	`endpointRef`	string	`mcp-astrovision-primary`	References `mcp_endpoints.endpointId` to attach the server. **[Partner input]** (chosen from endpoints)
7	`toolsAllowlist`	array<string>	`["get_horoscope","get_birth_chart","match_compatibility"]`	Only these tools can be called by the agent. **[Partner input]** (informed by discovered tools)
8	`toolPolicies.<tool>.timeoutMs`	integer (ms)	`4000`	Per-tool timeout budget; agent enforces. **[Partner input]** (may be prefilled from server hints)
9	`toolPolicies.<tool>.rateLimit.rpm`	integer	`200`	Per-tool RPM; enforced by agent/gateway. **[Partner input]** (may be prefilled)
10	`toolPolicies.<tool>.paramDefaults`	object	`{ "timezone": "Asia/Kolkata", "lang": "en" }`	Defaults injected when inputs omit them. **[Partner input]** (prefilled from schema defaults **[Auto-discovered]**)
11	`toolPolicies.<tool>.paramRequirements`	array<string>	`["sign","period"]`	Required params; validation before tool call. **[Auto-discovered]** from JSON Schema `required`, editable by partner
12	`toolPolicies.<tool>.privacy`	object	`{ "maskBirthTimeToNearestMinutes": 5 }`	Privacy/PII rules applied in agent. **[Partner input]** (may be hinted via `x-privacy` **[Auto-discovered]**)

13	`toolPolicies.<tool>.source`	object	`{ "timeoutMs":"server:x-timeout-ms", "rateLimit":"server:x-rate-limit-rpm", "paramDefaults":"schema.properties.*.default", "paramRequirements":"schema.required" }`	Provenance of policy values (server hints vs schema vs manual). **[System-managed]** (written by Validation/merge)
14	`toolPolicies.<tool>.lastSyncedAt`	ISO datetime	`2025-07-06T09:33:00Z`	When policy was last auto-populated from server hints. **[System-managed]**
15	`runtime.maxSteps`	integer	`4`	Max plan→act loops for the agent. **[Partner input]** (platform-owned policy)
16	`runtime.maxTokens`	integer	`2048`	Generation budget. **[Partner input]** (platform-owned policy)
17	`runtime.temperature`	number	`0.1`	Determinism vs creativity. **[Partner input]** (platform-owned policy)
18	`uiAllowlist` *(optional)*	array<enum>	`["card","table","form"]`	UI components allowed (if MCP-UI enabled). **[Partner input]**
19	`uiPolicies.maxInteractiveSteps` *(optional)*	integer	`3`	Cap UI back-and-forth loops per turn. **[Partner input]**
20	`uiPolicies.allowExternalLinks` *(optional)*	boolean	`false`	Block off-domain links in UI frames. **[Partner input]**
21	`uiPolicies.sanitizeHtml` *(optional)*	boolean	`true`	Enforce HTML sanitization on client. **[Partner input]**
22	`uiPolicies.maxPayloadKb` *(optional)*	integer	`64`	Max UI payload size per message. **[Partner input]**
23	`validation.status`	enum	`passed` | `failed` | `pending`	Gate for publish/runtime. **[System-managed]** (set by Validation Service)
24	`validation.lastRunId`	string	`val-astro-2025-07-06-001`	Link to latest validation run. **[System-managed]**

25	`validation.latencyP95Ms`	number	`620`	Performance SLO marker for this binding. **[System-managed]**
26	`createdAt`	ISO datetime	`2025-07-06T09:26:00Z`	Audit. **[System-managed]**
27	`updatedAt`	ISO datetime	`2025-07-06T09:34:00Z`	Audit. **[System-managed]**

4. `validation_runs` (immutable audit of checks)

GraphQL

```
{
  "_id": { "$oid": "66f101a0a1b2c30100000401" },
  "runId": "val-astro-2025-07-06-001",

  "tenantId": "tenant-jio",
  "skillId": "skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f",
  "skillVersion": "1.0.0",
  "endpointRef": "mcp-astrovision-primary",

  "status": "passed",

  "checks": [
    { "check": "handshake", "ok": true, "latencyMs": 120 },
    { "check": "listTools", "ok": true, "found": 3 },
    { "check": "toolSchema:get_horoscope", "ok": true },
    { "check": "toolSmoke:get_horoscope", "ok": true, "latencyMs": 260 },
    { "check": "ui:card", "ok": true, "latencyMs": 48 },               // optional, if UI probed
    { "check": "authScope", "ok": true }
  ],

  "metrics": { "latencyP95Ms": 620, "latencyAvgMs": 310 },
  "errors": [],
  "langfuseTraceId": "lf-astro-001",

  "createdAt": "2025-07-06T09:32:00Z"
}
```

	Field	Type	Example / Allowed Values	Description & Significance
1	`_id`	ObjectId	`66f101a0a1b2c30100000401`	Mongo primary key. **[System-managed]**
2	`runId`	string	`val-astro-2025-07-06-001`	Stable validation job id used for audit and joins from bindings. **[System-managed]** (generated by Validation Service)

3	`tenantId`	string	`tenant-jio`	Tenant isolation for validation context. **[System-managed]** (from portal/session)
4	`skillId`	string	`skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f`	Skill under validation. **[System-managed]** (resolved from request)
5	`skillVersion`	string	`1.0.0`	Version being validated (draft or published). **[System-managed]** (derived when job starts; partner may choose version in UI, but record is written by system)
6	`endpointRef`	string	`mcp-astrovision-primary`	MCP endpoint tested. **[System-managed]** (copied from binding at job start)
7	`status`	enum	`passed` | `failed` | `pending`	Overall outcome of the validation job. **[System-managed]**
8	`checks[].check`	string	`handshake`, `listTools`, `toolSchema:get_horoscope`, `toolSmoke:get_horoscope`, `ui:card`, `authScope`	Name of each probe executed (handshake, discovery, schema verify, smoke calls, optional UI probe). **[System-managed]** with results **[Auto-discovered]**
9	`checks[].ok`	boolean	`true/false`	Pass/fail per probe. **[System-managed]**
10	`checks[].latencyMs` *(optional)*	number	`120`	Latency of a probe (ms). **[System-managed]**
11	`checks[].found` *(optional)*	number	`3`	Count metric for a probe (e.g., tools discovered). **[System-managed]**
12	`checks[].details` *(optional)*	object	`{ "tools": ["get_horoscope","…"] }`	Small structured payload for extra context (kept lightweight). **[System-managed]** with values **[Auto-discovered]**
13	`metrics.latencyP95Ms`	number	`620`	Aggregated p95 latency across smoke calls (SLO signal). **[System-managed]**

14	`metrics.latencyAvgMs`	number	`310`	Aggregated average latency across probes. **[System-managed]**
15	`errors[]`	array<string>	`["auth failed: 401 during handshake"]`	Top-level errors, if any, for triage. **[System-managed]**
16	`langfuseTraceId` *(optional)*	string	`lf-astro-001`	Correlates the run with traces/spans in Langfuse. **[System-managed]**
17	`createdAt`	ISO datetime	`2025-07-06T09:32:00Z`	When the job record was created. **[System-managed]**

5. `sds_index_map` (lets SDS delete/rollback by exact IDs)

GraphQL

```
{
  "_id": { "$oid": "66f101a0a1b2c30100000601" },

  "tenantId": "tenant-jio",
  "skillId": "skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f",
  "versionId": "sv-astro-1-0-0",
  "environment": "prod",               // "prod" | "sandbox" (if/when sandbox returns)

  "collection": "skills_v1",
  "pointIds": [1200101, 1200102, 1200103],
  "numPoints": 3,

  "model": "text-embedding-3-large",
  "chunking": { "algo": "recursive", "maxTokens": 512, "overlap": 64 },

  "checksum": { "textHash": "sha256:...", "configHash": "sha256:..." },
  "batchId": "idx-2025-09-04-astro-0001",
  "active": true,

  "createdAt": "2025-09-04T10:30:00Z",
  "updatedAt": "2025-09-04T10:31:00Z"
}


```

	Field	Type	Example / Allowed Values	Description & Significance
1	`_id`	ObjectId	`66f101a0a1b2c30100000601`	Mongo primary key. **[System-managed]**
2	`tenantId`	string	`tenant-jio`	Tenant isolation for vector ownership and deletes. **[System-managed]** (copied from skill)

3	`skillId`	string	`skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f`	Skill whose content was embedded. **[System-managed]**
4	`versionId`	string	`sv-astro-1-0-0`	Exact skill version for these vectors (enables rollback). **[System-managed]**
5	`environment`	enum	`prod` | `sandbox`	Indexing environment/namespace. `sandbox` used for draft testing. **[System-managed]**
6	`collection`	string	`skills_v1` *(or `skills_v1_sandbox`)*	Qdrant collection name where vectors were written. **[System-managed]**
7	`pointIds`	array<int or string>	`[1200101,1200102,1200103]`	Exact Qdrant point IDs for this version. Enables deterministic delete/rollback. **[Auto-discovered]** (returned by Qdrant on upsert)
8	`numPoints`	integer	`3`	Count of vectors written for this version. **[System-managed]**
9	`model`	string	`text-embedding-3-large`	Embedding model used (audit, reproducibility). **[System-managed]** (from SDS config)
10	`chunking.algo`	string	`recursive`	Chunking strategy used. **[System-managed]** (from SDS config)
11	`chunking.maxTokens`	integer	`512`	Max tokens per chunk. **[System-managed]**
12	`chunking.overlap`	integer	`64`	Token overlap between chunks. **[System-managed]**
13	`checksum.textHash`	string	`sha256:...`	Hash of concatenated indexed text; detects drift/regression. **[System-managed]**
14	`checksum.configHash`	string	`sha256:...`	Hash of model/chunking config; ensures reproducibility. **[System-managed]**
15	`batchId`	string	`idx-2025-09-04-astro-0001`	Indexing batch/job identifier (join to logs). **[System-managed]**

16	`active`	boolean	`true/false`	`true` while version is live; set `false` on rollback/delete. **[System-managed]**
17	`createdAt`	ISO datetime	`2025-09-04T10:30:00Z`	When indexing map was created. **[System-managed]**
18	`updatedAt`	ISO datetime	`2025-09-04T10:31:00Z`	Last update time (e.g., reindex). **[System-managed]**
19	`expiresAt` *(optional)*	ISO datetime	`2025-09-19T10:31:00Z`	TTL for sandbox entries; rarely used in prod. **[System-managed]**

6. `skill_versions` (optional, if in case we keep version docs)

GraphQL

```

{
  "_id": { "$oid": "66f101a0a1b2c30100000501" },
  "versionId": "sv-astro-1-0-0",
  "skillId": "skill-astro-7c9e-4c3a-9c1a-1b2a3c4d5e6f",

  "semver": "1.0.0",
  "changelog": "Initial public release with horoscope, birth chart, and compatibility.",
  "publishedBy": "ravindra.singh@ril.com",

  "status": "published",
  "embedding": { "model": "text-embedding-3-large", "point_ids": [1200101, 1200102] },
  "metadata": { "retrieval_weight": 1.0 },

  "createdAt": "2025-07-06T09:28:00Z"
}


```

Indexes :

GraphQL

```
// skill_implementations
db.skill_implementations.createIndex({ tenantId: 1, skillId: 1 }, { unique: true });
db.skill_implementations.createIndex({ tenantId: 1, skillName: 1 });
db.skill_implementations.createIndex({ skillName: "text", skillDescription: "text", tags: "text" });

// mcp_endpoints
db.mcp_endpoints.createIndex({ endpointId: 1 }, { unique: true });
db.mcp_endpoints.createIndex({ tenantId: 1, partnerId: 1 });
db.mcp_endpoints.createIndex({ "health.status": 1, updatedAt: -1 });

// skill_mcp_bindings
db.skill_mcp_bindings.createIndex({ skillId: 1, skillVersion: 1 });
db.skill_mcp_bindings.createIndex({ endpointRef: 1 });
db.skill_mcp_bindings.createIndex({ "validation.status": 1 });

// validation_runs
db.validation_runs.createIndex({ runId: 1 }, { unique: true });
db.validation_runs.createIndex({ skillId: 1, skillVersion: 1, endpointRef: 1, createdAt: -1 });
```

```
19
20 // sds_index_map
21 db.sds_index_map.createIndex({ tenantId: 1, skillId: 1, versionId: 1 }, { unique: true
   });
22 db.sds_index_map.createIndex({ environment: 1, active: 1 });
23
24
25
```

This section covers all internal/external APIs. Define interfaces and Data models (DB Schemas/internal POJO/ DTO / DAO / JSON etc...)

System Flow Summary

Step-by-step narrative of a request—from user input through orchestration to response. Captured above in Key Sequence section

---

Observability & Instrumentation

States what metrics, logs, and traces are collected and how alerts are triggered.

	Signal	Tool	Notes
1	**Logs**	Databricks Dashboard	Structured logs with correlation IDs: `ts, level, service, requestId, tenantId, skillId, bindingId, endpointRef, event` (e.g., `MCP_CALL_START`, `SDS_SEARCH`, `BOOTSTRAP_READ`). Keep payloads summarized; mask PII; secrets never logged. Retention: **7d hot, 30–90d warm** (configurable).
2	**Alerts**	Alertmanager → **PagerDuty**	SLO/SLA-driven paging with runbooks. Examples: (1) **SDS p95** > **100ms** for 10m; (2) **Orchestrator 5xx** > **1%** for 5m; (3) **MCP endpoint health** = `down` 5m; (4) **Validation failure rate** > **10%** over 30m; (5) **Kafka lag** > **10k** for 10m; (6) **Qdrant error rate** > **0.5%** for 5m; (7) **Vault secret fetch failures** > **5/min**; (8) **LLM cost burn** > daily budget. Alerts include Grafana + Langfuse deep links.

Assumptions & Constraints

Lists what the design presumes to be true (e.g., cloud region, budget) and any hard limits (e.g., data-residency).

Assumptions

- **Identity & Secrets**
  - OIDC (tenant-aware) for users; JWT/mTLS for services.
  - Partners provide credentials once; the portal stores **only** `secret_ref` (no plaintext secrets in DB).
- **Protocol / Interop**
  - MCP protocol **mcp-1.0** is used by partners; servers implement `list_tools` and tool invocation consistently.
  - Optional **MCP-UI** frames may be emitted by some partners; clients support a safe subset (card/table/form).
- **Validation Gate**
  - Skills/bindings must **pass validation** before they are eligible for publish and runtime (`strict=true`).
- **Time & Locale**
  - Timestamps stored in UTC; examples use **Asia/Kolkata**; localization handled at the client/UI layer.
- **Sandbox**
  - Sandbox/testing workflow is **deferred**; current scope focuses on production runtime.

Constraints

- **Security & Compliance**
  - **No raw secrets** stored anywhere—only `secret_ref` /KMS references. Token resolution happens at runtime.
  - PII/inputs to tools are **redacted/summarized** in traces; no chain-of-thought persisted.
- **Access Control**
  - All reads/writes are **tenant-scoped**; cross-tenant data access is prohibited.
  - Only **allowlisted tools** in a binding can be invoked at runtime.
- **Validation & Publication**
  - Binding `validation.status` must be `passed` for bootstrap/use; otherwise service returns **409** (when `strict=true`).
  - Tool policy provenance is recorded; platform values **override** partner hints.
- Capacity & Limits (guardrails)
  - **MCP-UI** frames: only **allowlisted components**, payload ≤ **64 KB**, **no external links** unless allowed by policy.
- **Change Management**
  - Publishing creates a new `versionId` ; SDS reindex is **idempotent**; rollback uses `sds_index_map` .
  - Backward-incompatible schema changes require versioned APIs or migrations before rollout.
  - **Third-Party Dependencies**
    - MCP endpoints must provide stable SLAs and notify on schema changes; breaking changes without notice may lead to **temporary unlisting**.

Security Considerations

Catalogues auth, RBAC, malware scanning, audit logging, and key threat mitigations across the Partner Portal → Validation → SDS → Runtime (Orchestrator & Skill Agent Service) path.

1) Identity, Auth & RBAC (Need to discuss further)

- **User AuthN (Portal): OAuth2 / OIDC via JioID** issuing **JWT** (short-lived access tokens, refresh via PKCE).
- **Service AuthN: mTLS** between services; **JWT** for intra-API calls with `aud/iss/exp` checks and key rotation.
- **AuthZ (RBAC):** Tenant-scoped roles — **Partner Owner / Editor / Viewer** — plus Platform roles (**Ops**, **Validator**).
- **Skill invocation RBAC:** Only published skills with `policy.visibility` permitted; **binding.validation.status=passed** required when `strict=true` .
- **ABAC:** Enforce `tenantId` on every read/write; **cross-tenant access is denied** by default.
- **Principle of Least Privilege:** SAS only receives the **binding** + **secret_ref** it needs for the chosen skill.

2) Secrets & Keys

- **Secret storage:** No plaintext secrets in Mongo; store `secret_ref` only. Backed by **Google Secrets Manager**.
- **Retrieval:** SAS resolves secret refs **at call time**; results held in memory only; **never logged**.
- **Rotation:** Partner tokens rotated per policy (e.g., 90 days). Validation re-runs on rotation.
- **Key management:** KMS-managed keys for envelope encryption (optional field-level encryption for sensitive prompts).

3) Data Protection (At Rest & In Transit)

- **Transport:** TLS 1.2+ externally; **mTLS** internally. WS (MCP), HTTP, and gRPC all terminate at gateway/ingress with WAF.
- **At rest:** Disk encryption for Mongo/Qdrant; object storage for specs is KMS-encrypted.
- **Field-level protections:** Optional encryption for prompt fields; PII masking per policy.

4) Malware & Content Scanning

- **Uploads:** Any partner-uploaded files (e.g., OpenAPI specs, sample payloads) are **scanned before LLM access** and before persisting references.
- **Runtime attachments:** If MCP or tools return file links, enforce **allowlisted MIME types**, size caps, and scan before exposing to clients.
- **Blocking:** Quarantine & reject on detection; notify partner and log an audit event.

5) Input/Output Safety (LLM + Tools)

- **Prompt-injection hardening:**
  - Route only to **allowlisted tools** (`toolsAllowlist`), **never** open-ended code exec.
  - **Param validation** against tool JSON Schemas; reject unknown/extra keys.
  - Budget guardrails from binding `runtime` (maxSteps, tokens, timeoutMs per tool).
- **Data exfil prevention:**
  - No secrets in prompts/responses; redact credentials/PII in traces and logs.
  - SAS **egress restriction** to declared **MCP** `baseUrl` (DNS pin / IP allowlist) to mitigate SSRF.
- **MCP-UI safety (if enabled):**
  - Enforce `uiAllowlist` and `uiPolicies` (payload ≤ N KB, **no external links** unless explicitly allowed, strict HTML sanitization).
  - Clients ignore unknown components; degrade to text.

6) Network, Platform & Supply Chain

- **Gateway/WAF:** Rate limiting, IP allowlists, header normalization, request/response size limits, bot/DDoS protections.
- **Kubernetes hardening:** Non-root containers, read-only root FS, seccomp/AppArmor, NetworkPolicies (east-west), image signing (Cosign), SBOM + SCA scans.
- **Dependency hygiene:** Automated vulnerability scans; critical CVEs block deploy until patched.
- **Change control:** GitOps (Argo CD) with reviews; infrastructure as code (Terraform).
- **Multi-region:** Data residency honored per tenant; geo-fencing of compute/DB if required.

7) Logging, Tracing & Audit

- **Structured logs:** PII-safe JSON with correlation IDs (`requestId`, `tenantId`, `skillId`, `bindingId`, `endpointRef`).
- **Traces:** OpenTelemetry + **Langfuse** with redacted attributes (no raw tool inputs, no secrets).
- **Audit trail:** Append-only events for **who changed prompts/bindings**, **who validated/published**, **who invoked** which skill/tool.
- **Tamper resistance:** Write-once storage tier (or hash-chaining) for sensitive audit records; time sync via NTP.

8) Alerts & Incident Response (IR)

- **Alerting:** PagerDuty on SLO/RBA violations (auth anomalies, MCP health down, SDS latency spikes, secret fetch failures).

- **IR playbooks:** Threat-specific runbooks (token leak, SSRF, prompt-injection incident, schema-drift rollback), with auto-capture of relevant traces/logs.
- **Access reviews:** Quarterly RBAC audits; break-glass accounts gated and logged.

---

9) Threats & Mitigations (Quick Map)

	Threat	Mitigations
1	**Prompt injection / tool hijacking**	Tool **allowlist** at binding; strict JSON Schema validation; runtime **maxSteps/tokens**; strip/system-prompt hardening; no chain-of-thought logging.
2	**Cross-tenant data leak**	Tenant-scoped queries everywhere; SDS/Qdrant payload filters (`tenant_id`); authZ checks at gateway and service; per-tenant encryption keys (optional).
3	**Credential exfiltration**	Only `secret_ref` stored; secrets fetched just-in-time; redaction in logs/traces; egress pinned to MCP host; no reflective prompts.
4	**SSRF / egress abuse**	SAS egress restricted to **allowlisted MCP baseUrl**; DNS pinning / IP allowlist; proxy with ACLs; block file://, 169.254/metadata.
5	**Schema drift / breaking MCP changes**	Validation job detects **schemaHash** changes; bindings require re-approval; route down-weights degraded endpoints automatically.
6	**DoS / cost overrun**	Rate limits (gateway + tool RPM); concurrency caps; LLM budgets; circuit breakers; back-pressure; alert on **LLM cost burn**.
7	**Malware via uploads or MCP content**	Pre-ingest **AV scanning**, MIME/type allowlists, size caps, quarantine + alert; disallow executable artifacts.
8	**Malicious MCP-UI frames**	`uiAllowlist`, strict sanitization, payload caps, disallow external links unless policy allows; client ignores unknown components.
9	**MITM / token theft**	TLS 1.2+ everywhere; mTLS internally; token bound to `aud/iss/exp`; short-lived creds; Vault access via workload identity.
10	**Unauthorized publish or policy change**	RBAC + dual control for publish; audit log with immutable storage; notification on policy changes; require **validation.status=passed**.

---

10) Defaults & Guardrails (recommended)

- **JWT TTLs:** 15–30 min access, 24 h refresh; clock skew ≤ 60 s.

- **Rate limits:** Per-tenant global RPM + per-tool RPM from `toolPolicies`.
- **Payload limits:** Request/response ≤ 1–5 MB (service-specific); MCP-UI ≤ 64 KB (configurable).
- **Timeouts:** Tool `timeoutMs` defaults (e.g., 4–8s); Orchestrator E2E soft ≤ 8s, hard ≤ 20s.
- **Retention:** Logs 7d hot/30–90d warm; traces 30–90d; audit ≥ 1 year (per policy).
- **Publish gate:** `validation.status=passed` + health not `down` in last 10 min.

Degradation & Fallback Strategy

Defines graceful-degradation tactics—**read-only mode**, **cached responses**, and **feature flags**

Modes & What Changes

	Mode	When it Activates	What Changes (Scope)
1	**Normal**	All SLOs healthy	Full capabilities: SDS top-K retrieval, LLM routing, SAS planning, MCP tools (e.g., `get_horoscope`, `get_birth_chart`), optional MCP-UI.
2	**Degraded (Routing/Tools)**	MCP endpoint health=`down/degraded` or tool error-rate↑; **SDS p95 > 120 ms** for 10 min	Orchestrator re-routes to next healthy candidate; SAS limits planning (≤ `maxSteps_degraded`), reduces K, disables non-essential tools, suppresses MCP-UI frames.
3	**Reduced-Cost Mode**	Daily LLM budget ≥ threshold or traffic surge	Switch to cheaper model, lower `maxTokens`, disable reranker/reasoning; enable aggressive caches.
4	**Read-Only Mode (ROM)**	Mongo/Vault **writes** failing, or publish queue backlog high	Portal: edits/publish disabled; SDS indexing paused (503); runtime reads allowed; services served from cache.
5	**Maintenance / Safe Halt**	Multiple critical deps failing	Serve cached apology/"status" responses; health endpoints only.

Trigger → Action Matrix

	Trigger (from Observability)	Automatic Actions	Operator Playbook
1	**MCP endpoint** `down` 5 min or p95 > 800 ms	Orchestrator retries/backoff → pick next candidate; SAS disables affected tools; endpoint weight ↓	Notify partner; run validation; if persistent, temporarily unlist endpoint.

2	**SDS p95 > 120 ms** 10 min	Lower `retrieval.k`, enable candidate cache, pre-warm Redis; degrade reranker	Inspect Qdrant health/HNSW params, recent publish volume; scale SDS/Qdrant.
3	**LLM cost burn > daily budget (Can be handled in LLM Router)**	Switch `llm.model` to cost tier; cut `maxTokens`; disable reasoning; cap concurrency	Review traffic; apply tenant-level quotas; tighten binding budgets.
4	**Mongo write errors > 5%**	Enter **ROM**	Investigate primary/replica health; failover; clear ROM after recovery.
5	**Vault secret fetch failures** > 5/min	Deny new MCP connects; serve from cached service	Check Vault health/permissions; rotate tokens if needed.
6	**Validation failure rate > 10%** 30 min	Block publish; keep runtime unaffected	Check schema drift; contact partners; roll back last binding changes.

Panic / Kill-Switch Mode

A hard stop designed to **contain risk instantly** (security incident, runaway cost, partner compromise) by halting risky paths, preserving forensics, and providing a safe user experience.

Objectives

- **Protect users & data** (immediately cut risky execution paths).
- **Bound blast radius** (scope by tenant/skill/endpoint/region or global).
- **Preserve evidence** (freeze audit/traces/logs for IR).
- **Offer clear UX** (status banners / static responses).
- **Enable controlled recovery** (gated re-enable with smoke checks).

Activation Scopes & Levels

	Scope	Examples	Level	What it means
1	**Global**	Platform-wide event (e.g., provider breach)	**Halt**	Stop all runtime invocations, MCP egress, publish/validate/index.
2	**Region**	Cloud zone outage	**Halt / Read-only**	Freeze writes in region; route reads to healthy region (if allowed).
3	**Tenant**	Single-tenant incident or legal hold	**Halt / Read-only / Tool-less**	Block that tenant's runtime & publish; optionally allow read-only portal.
4	**Skill / Binding**	Compromised prompts/policies, schema drift	**Halt / Tool-less**	Disable that skill or binding; other skills work.
5	**Endpoint**	Partner MCP compromised or unstable	**Halt / Egress-block**	Block SAS egress to that endpoint; re-route to alternates.

Compliance & Regulatory

Maps design choices to GDPR, RBI, ISO-27001, SOC-2, or other mandates.

- Data residency requirements
- Consent flows / audit log retention

Scalability & Extensibility

Describes autoscaling, sharding, and plug-in mechanisms for models, skills, tools.

	Layer	Strategy
1	Core services	Horizontal autoscale
2	Vector DB	Sharding per tenant
3	LLM Router	Pluggable providers

Future Roadmap

- Multi-modal extensions (video, AR)
- Edge LLM serving

Glossary

	Acronym	Meaning
1	**MCP**	Model Context Protocol
2	**RAG**	Retrieval-Augmented Generation
3	**MyHub**	JioOmni in-app user action center

Open Issues / Risks