Chuan
xu

DATABASE INFRASTRUCTURE FOR AI AGENTS

Chuanxu Whitepaper

Database infrastructure and collaboration governance for enterprise AI agents

Public edition · English

Product baseline: v5.0.4

Software packaged: November 5, 2026

Document date: November 5, 2026

Contents

01 About this paper

02 Executive summary

03 1. From a single answer to ongoing work

04 2. Positioning and design principles

05 3. Architecture and authoritative data model

06 4. People, agents, organizations, and security domains

07 5. Database-native retrieval with multiple signals

08 6. Source-based knowledge intake and publication

09 7. Temporal memory, working notes, and context

10 8. Tasks, Graphs, Loops, and controlled execution

11 9. Work, Handoff, and DB4A2A collaboration

12 10. Tools, Skills, protocols, and frameworks

13 11. Model routing, capacity, usage, and cost

14 12. Security, governance, and compliance responsibilities

15 13. Database adaptation and deployment prerequisites

16 14. Installation, upgrades, operations, and recovery

17 15. Editions, licensing, and learning material

18 16. Illustrative workflows

19 17. Verification approach and evidence boundaries

20 18. Proof of concept, selection, and implementation

21 Appendix A: Glossary

22 Appendix B: Sources and further reading

About this paper

This paper is intended for enterprise technology leaders, architects, database administrators, security and compliance teams, and teams delivering AI applications and agents. It explains Chuanxu's positioning, architecture, principal workflows, deployment requirements, and evaluation approach so that readers can assess its suitability and design a proof of concept.

The paper is based on the current v5.0.4 product, delivery documentation, and verification records. Statements of support depend on the selected edition, database, authorization, configuration, and deployment prerequisites. Protocol adapters cover only the versions and operations explicitly described. Scenarios illustrate the design; they are not customer case studies. Suggested implementation steps do not promise delivery times, performance, regulatory certification, or service levels. Procurement and production use are governed by the applicable software license and commercial agreement.

The Chinese and English editions share the same structure and factual scope. For later releases, consult the actual package's release notes, deployment guides, license, and migration instructions. Appendix B identifies the product documentation used here. This paper does not replace installation instructions or database security configuration guides.

Executive summary

Chuanxu is enterprise AI agent infrastructure that uses the database as its authoritative source of state. It connects human and agent identities, organizations and security domains, knowledge and memory, tasks and collaboration, model and tool calls, and approval and execution evidence. Enterprises can deploy it in their own environments with Oracle, PostgreSQL, or YashanDB as the database backend. [R1]

Chuanxu addresses questions that arise when agents participate in ongoing business work: who is accountable for an agent, what it may read now, which revision supports an answer, where a task stopped, whether a handoff was accepted, whether an external action actually completed, and whether old context remains usable after access is revoked. These questions require inspectable state, versions, and authorization. Conversation history and a model's description of progress are insufficient.

The design has three priorities. First, business entities and execution records remain authoritative in the database; graph relationships and summaries retain their sources and cannot become alternative authorization systems. Second, current permissions are checked when reading, preparing context, sending model requests, calling tools, and disclosing results. Invitations, document references, and agent assertions cannot grant resource access. Third, plans, approvals, dispatch, actual execution, and acceptance are recorded separately. Unknown outcomes remain unknown until evidence supports recovery.

Version 5.0.3 integrates relational attributes, vectors, full text, tags, and bounded graph relationships into knowledge answers and task context. Guided intake captures sources, applicability, and responsibility. Ranking uses database queries and deterministic fusion without adding generation calls. Query embeddings, context, and answer generation still have costs, and retrieval does not establish evidence sufficiency or answer correctness. [R2–R4]

The public product family consists of one Community package offering three database choices and three separate Enterprise editions for Oracle, PostgreSQL, and YashanDB. Community provides the common knowledge, memory, and collaboration foundation. Enterprise adds dedicated approval, audit management, LDAP, Skill Token, orchestration governance, and compliance modules. Enterprise production use requires a commercial license. Selection should consider governance needs, database prerequisites, and actual model capabilities alongside interface features. [R15]

Version 5.0.4 adds official incident sources, evidence-linked incident report drafts, and context assembly evidence, and strengthens final model input and complete output checks. Public incidents establish provider-side facts, not internal enterprise impact, unknown root causes, or remediation acceptance. Preparation measurements enable no query/TTL cache; shared encoding remains disabled research. [R20, R22]

1. From a single answer to ongoing work

A single question may be answered in one request. Ongoing work can span days, people, agents, and external systems. A support task might consult documentation, gather diagnostic evidence, hand the investigation to another agent, and seek human approval for a change. Permission changes, revised documentation, interrupted connections, and retries can each alter whether the next action is permitted.

When state is scattered across chat windows, vector stores, script logs, and framework memory, basic questions become difficult to answer. Which source revision supported the output? Is a previous read still authorized? Can an old worker still publish results? Will recovery repeat an external side effect? A fluent model explanation cannot replace records of these events.

Chuanxu connects these concerns in a common workflow. A human sets the goal and authorization. An agent discovers and reads permitted knowledge. Tasks and Steps record the work. Models and tools perform controlled actions. Work and Handoff records capture responsibility and transfer. Actual receipts determine what may proceed. Experience intended for long-term reuse enters independent review and publication rather than becoming enterprise knowledge merely because it appeared in an answer.

The product is suited to agent applications requiring persistent state, multiple participants, source traceability, and access governance. A direct model API may be sufficient for one-off text generation. Existing application platforms and agent frameworks can connect within explicit protocol and execution boundaries; integration does not imply that every external framework feature is hosted or verified.

2. Positioning and design principles

Chuanxu provides a shared foundation for data, execution, and governance. It can support knowledge answers, long-running tasks, execution graphs, bounded loops, and cross-agent handoffs. The database supplies persistent state and native permissions. Models generate or reason over approved input. Business systems retain responsibility for their own transactions and approvals.

Principle	Practical meaning	Business implication
Authoritative database entities	Identities, resources, tasks, versions, and receipts are explicit entities; JSON carries bounded content and protocol data	Actual state can be inspected without relying on conversational completion claims
Human accountability	Business agents have an active human primary owner	Responsibility is traceable, and access follows current organization and authorization
Authorization at use	Discovery, reads, sends, and disclosure check current permission	Old references and cached context do not extend revoked access
Exact versions and sources	Content, plans, and context bind to specific revisions and digests	Historical evidence can be distinguished from current facts
Bounded autonomy	Execution depends on authorization, leases, budgets, stop conditions, and receipts	Recovery and retries cannot silently expand capabilities
Explicit uncertainty	Unknown transport, missing usage, and unavailable aggregates have separate states	Unknown outcomes are not reported as success or zero cost

These principles do not make a model automatically correct. They provide the conditions needed to inspect its input, constrain actions, diagnose errors, and assign responsibility. People retain the responsibilities appropriate to factual judgment, business risk, and final acceptance.

3. Architecture and authoritative data model

The application layer exposes Web and machine access through FastAPI. The React Dashboard supports administration and work operations; the Portal supports user interaction. Business services reuse common identity, resource, task, and authorization logic. Backend adapters handle database-specific SQL, indexes, native permissions, and retrieval behavior. [R1]

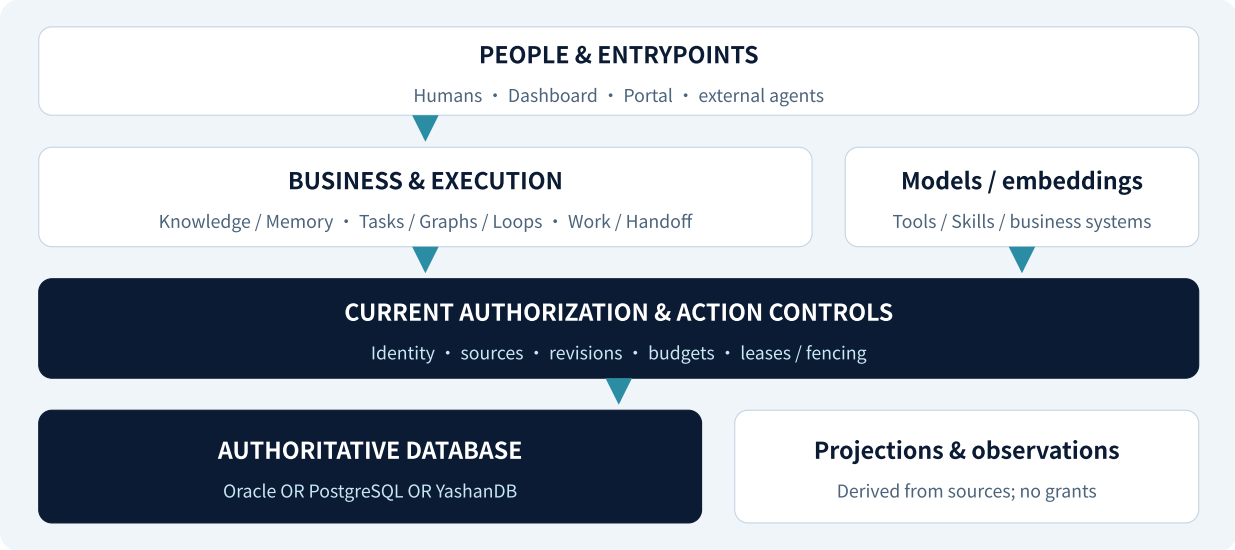


Figure 1: Chuanxu's logical architecture. The database holds authoritative state. Models, tools, and external agents participate through current authorization and execution controls. These layers describe responsibilities, not mandatory deployment as separate services.

Authoritative objects include principals and organization relationships, security domains and channels, Knowledge and Memory revisions, Tasks and Steps, workspaces and branches, Graph plans and runs, Loops, Work and handoffs, candidates, reviews, publications, leases, and audit records. Relationships between explicit objects express state changes, responsibility, and access boundaries more precisely than a natural-language summary alone.

A property graph describes relationships between business objects and supports bounded relationship retrieval and visualization. An execution graph describes nodes, dependencies, barriers, and runtime results. These are distinct uses. Property-graph projections are rebuildable and neither override source entities nor grant permission. Execution graphs bind to specific compiled and published plans.

Each application process selects its backend before importing database drivers and rejects conflicting configuration before connection. The Community package offers three deployment choices, with each installation using only the selected backend and dependencies. Moving between backends requires a separate target, installation, and migration verification. The product does not transparently federate three databases into one online transaction system.

4. People, agents, organizations, and security domains

Humans, platform-native agents, external agents, and database Owners have different responsibilities. Agent registration, activation, runtime readiness, and resource grants are separate states. A business agent must have an active Human primary owner. Registration alone grants no model, tool, knowledge, or execution permission. [R5]

Organizations express administrative affiliation. A Human has one primary organization and may have temporary additional relationships. The organization hierarchy and its effective closure participate in current access decisions. A security domain expresses a collaboration authorization boundary; a channel provides conversation and work communication. Channel membership does not grant access to every resource available to other participants or to database rows.

A knowledge-assisted task typically involves both the requesting Human and the executing agent. Usable sources must satisfy the intersection of their current permissions, together with resource, domain, lifecycle, and purpose constraints. An executor cannot use the database Owner, another user, or a replacement token to bypass restrictions on the original request.

External agents enroll through controlled registration and identity proof, using a one-time secret or Ed25519 proof, and receive short-lived tokens bound to an instance. Execution records link credentials, instances, leases, and fencing. Administrators can distinguish an agent description from verified identity and from permission for that particular instance to act now.

Deployments should define procedures for departures, organization changes, disabled owners, and leaked credentials. Revocation should affect subsequent reads and execution. Historical records preserve what happened, but a historical snapshot cannot restore current access.

5. Database-native retrieval with multiple signals

Version 5.0.3 combines five logical signals in knowledge answers and task context: relational attributes, vector similarity, full text, tags, and bounded graph relationships. Relational attributes handle exact product, version, responsibility, and time conditions. Vectors assist semantic recall; full text preserves names and terminology; tags express classification; graph relationships contribute bounded connections. Actual indexes, queries, and available signals depend on the backend and current configuration. [R2]

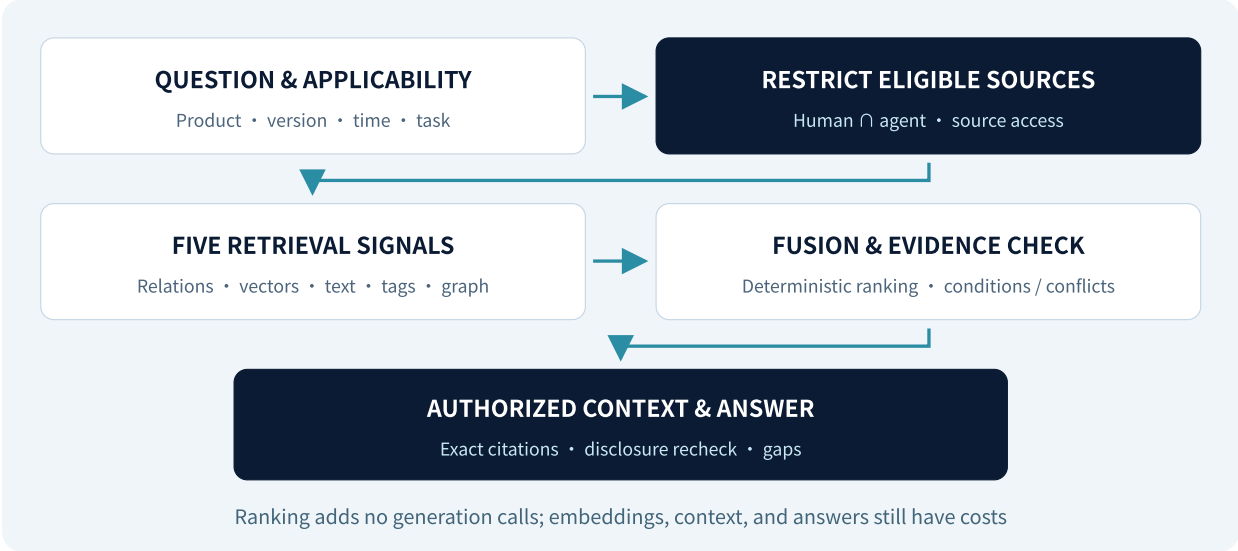


Figure 2: From a question to authorized evidence. Authorization and applicability precede candidate truncation. Candidate loading, model disclosure, and citation reads continue to check exact sources.

Retrieval first restricts the population by principal, resource type, lifecycle, applicable version, and time. Recall, fusion, and candidate limits then operate within that authorized population. Fetching a global top set and removing unauthorized records afterwards can both expose information and displace usable results. Ranking scores cannot override permission or applicability.

Database queries and deterministic rank fusion add no generation call. Query embeddings may be reused within the same contract. Duplicate material can be consolidated while retaining distinct scopes, conditions, conflicts, and exact identities. Query embeddings, context input, and final generation still consume resources; the design does not make an entire answer workflow token-free.

If an embedding or retrieval service is unavailable, fallback is limited to an explicitly configured, authorized text path, with the actual strategy and reason recorded. An access denial cannot trigger a broader fallback. Full match, partial match, no match, and ambiguity are assessed separately; a high-ranked candidate is not automatically sufficient evidence.

Answers can distinguish knowledge-grounded content, knowledge extracts, model supplementation, mixed sources, clarification requests, and insufficient knowledge. Knowledge text may be sent only to a model Profile approved to receive it. Without disclosure approval, the service can return extracts the user is authorized to see. General model supplementation requires policy permission and separate labeling. Facts, inference, and conditional advice remain distinguishable. [R3]

6. Source-based knowledge intake and publication

Reliable use of enterprise knowledge begins at intake. Chuanxu's guided workflow captures purpose, source location, responsible actor, topic, applicable product and version, time, classification, and intended audience. Chunk previews and quality feedback follow. Table headers, code, units, negations, and applicability conditions should remain with the content so that chunking does not change its meaning. [R4]

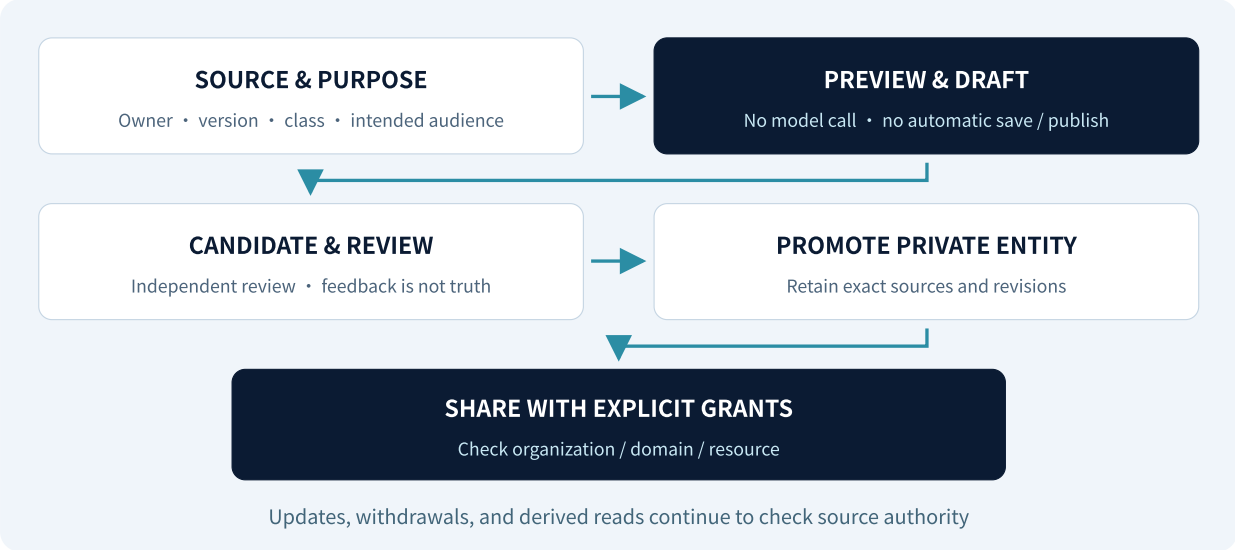


Figure 3: The lifecycle from a user source to sharing. Promotion after review creates a private native entity. Broader sharing still requires separate, explicit authorization.

Preview invokes no model and neither saves nor publishes knowledge. Incomplete material may remain a draft. Missing-field, duplicate, and conflict feedback helps improve submissions but does not establish factual truth. Candidates use the existing independent review and publication chain. Reviewers cannot approve their own submissions; content promotion and sharing rights remain separate decisions.

Promotion creates a private native entity. The requested audience expresses intent, not an access grant. Organization sharing or cross-domain distribution requires additional authorization. Community retains the common candidate and independent review workflow, which should be distinguished from Enterprise-specific governance approvals.

Web, API, MCP, CLI, curated import, and source synchronization reuse server-side rules. Personal private notes retain their existing direct-write path. Direct creation of shared knowledge is directed into governed intake. Controlled source synchronization creates candidates; file changes do not automatically publish content, and source code is not executed.

Published knowledge retains exact revisions, digests, source locations, products, versions, and time. Updates, withdrawals, and conflicts affect subsequent use. Summaries, answer history, and derived content must recheck source permissions when read or disclosed. Built-in product learning material has a separate trusted, source-bound publication lifecycle. That mechanism does not authorize automatic publication of arbitrary user content.

Official incidents and report drafts

Version 5.0.4 collects bounded records from official GitHub and Cloudflare status APIs, retaining raw snapshots, event times, exact revisions, and idempotency records through the existing Source, Task, and candidate review chain. Successful collection records external material; it neither publishes knowledge nor executes remediation. [R22]

Incident reports use an attributed PagerDuty open-template structure for facts, timelines, evidence, gaps, and advice awaiting confirmation, retaining the applicable license. Public sources cannot establish a particular enterprise's internal impact or fill an unpublished root cause. Teams add internal logs, accountable owners, and action receipts before independent review and explicit promotion. Example reports remain drafts.

7. Temporal memory, working notes, and context

Memory captures events, observations, and factual relationships needed across tasks. Event time, observation time, ingestion time, and applicability time have different meanings. Late-arriving data cannot automatically replace an earlier fact simply because it was ingested later. Missing time or timezone information remains unknown rather than becoming an invented precise timeline. [R6]

Confirming a conflict relationship and publishing content are different actions. Restoring historical content creates a new successor revision and preserves history. It does not move a current pointer backwards or lower classification through restoration. Readers can trace when a judgment was recorded, revised, and supported by particular evidence.

Working notes retain findings, rejected approaches, validation results, open questions, and next actions. Notes and sources have immutable revisions and remain subject to expiry, Work, and Handoff access checks. Intermediate ideas do not automatically become formal knowledge. Even a generated retrospective first produces a pending candidate for independent review and publication.

Context assets follow metadata-first discovery and exact-version loading. Bound sources and permissions are rechecked. Budgets reserve space for system instructions, tool protocols, and output. Conditions, negations, conflicts, and mandatory evidence cannot be silently truncated to fit. An insufficient budget blocks dispatch so that the task can be narrowed or an approved configuration adjusted. [R7]

The system distinguishes counts from a known tokenizer from conservative estimates. It also distinguishes prepared context from evidence that the actual input was delivered. Diagnostics can therefore investigate whether material entered that particular model request, rather than merely showing that the server once constructed it.

Version 5.0.4 assembly evidence shows exact source revisions, representations, budgets, and reasons for selection, omission, or fallback. Preparation grants no source access and proves no dispatch. Use rechecks sources, expiry, and current authorization; historical records cannot continue disclosing revoked content. The interface explains assembly decisions and does not provide independent transport-byte proof. [R7, R22]

8. Tasks, Graphs, Loops, and controlled execution

Tasks and Steps provide stable work identities, while attempts record individual execution tries. A lease identifies the worker currently entitled to execute. Fencing prevents an expired executor from publishing results, and idempotency identifiers help detect repeated requests. Persistent records support diagnosis and recovery after process restarts. [R8]

A Graph compiles nodes and dependencies into a specific plan version. Once publication is authorized, runs bind to that exact plan. Nodes may involve agents, models, Skills, tools, databases, humans, events, and subgraphs; attempts, events, checkpoints, and results are recorded separately. Barriers retain participant snapshots and arrival records without holding worker execution leases while waiting.

A Loop supplies bounded iteration within the same execution system, with iteration, token, time, and stop limits. Evaluations may use tests, differences, model judgment, manual review, specifications, or aggregation. An evaluation method does not change the evidentiary strength of its result. A Harness organizes goals, environment, and acceptance requirements into a run blueprint; it grants no model or tool permission.

Logical deduplication cannot guarantee exactly-once side effects in arbitrary remote systems. If a response is lost after creating a ticket, initiating a payment, or changing an external system, the actual outcome must be checked, with compensation or human intervention where needed. Offline replay is diagnostic and does not automatically repeat external actions. Migration preflight does not imply support for moving arbitrary active runs online.

Processing budgets cannot be reset through a restart, new attempt, or Graph fork. Recovery requires human authorization for an exact target, version, and bounded budget. Accepting a recovery request, dispatching it, executing it successfully, and approving the business result are separate events. BLOCKED, DEGRADED, and UNKNOWN retain distinct meanings.

9. Work, Handoff, and DB4A2A collaboration

Work records the goal, responsibility, version, and acceptance conditions. Handoff records the offer, acknowledgment, start, outcome, and read evidence. In the default single-recipient workflow, responsibility transfers on acceptance. Explicit parallel handoffs can be bounded to 2–16 recipients, with the coordinator retaining responsibility. This is a workflow boundary, not a capacity benchmark. [R9]

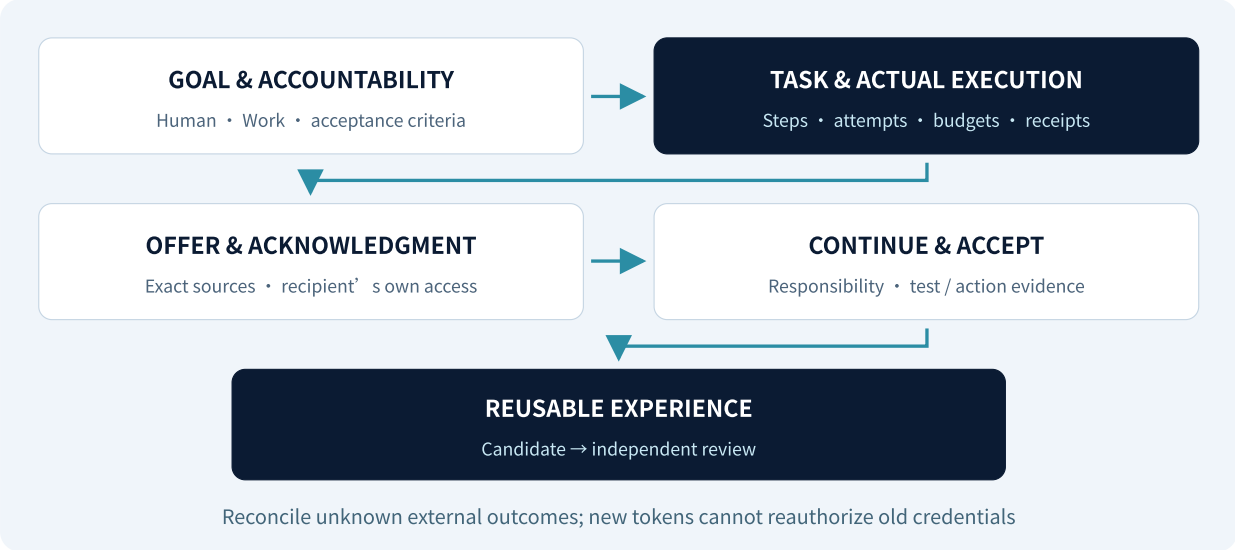


Figure 4: Continuing work and transferring responsibility. Each step links actual versions and receipts. A generated summary cannot replace acknowledgment or execution evidence.

DB4A2A is Chuanxu's collaboration pattern for sharing context through a database. Small task or status envelopes carry `task_id`, `context_ref`, `snapshot_digest`, `expected_version`, `scope_ref`, and `branch_policy`. The recipient authenticates as itself and reads the exact context it is currently authorized to access. A reference grants no permission. [R10]

Reads do not modify the source by default. Creating a derived work branch requires explicit authorization and consistent source context, dispatch, and audit records. Merging into the original branch remains a separate action. Task, Graph, and collaboration snapshots describe state at capture time. Reading them rechecks source permissions; they do not establish current runtime status.

DB4A2A can reduce repeated transport of the same context and version drift after copying. Benefits depend on shared-data arrangements, access paths, context size, and model use. They do not establish a universal latency or token-saving percentage. DB4A2A is not a replacement for the A2A standard. Where no shared authorized data plane exists, A2A payload or artifact transport is still needed.

Unknown external sends remain `UNOBSERVED` until the actual original outcome is reconciled. A newly issued token cannot replace a revoked original credential to retroactively authorize an old operation. A handoff therefore carries more than a next action: it retains its evidence, the authority to continue, and whether the previous step actually completed.

10. Tools, Skills, protocols, and frameworks

Chuanxu separates capability discovery from execution. An agent can inspect Skill or tool metadata and then load exact content as needed. Execution still depends on current registration contracts, credentials, security domains, source access, and leases. Skill instructions and tool output cannot elevate themselves into system security instructions. [R11]

Integration	Explicit current coverage	Boundary
MCP	Versioned Streamable HTTP adapters for 2025-06-18 and 2025-11-25	Use the operations and contracts of each version; arbitrary MCP server compatibility is not claimed
Later MCP adapter	Bounded 2026-07-28 `server/discover`, `tools/list`, and read-only `tools/call`	Complete results and a bounded schema subset; no subscriptions, tasks, MRTR, `input_required`, or other unopened operations
A2A	Bounded 1.0.1 HTTP JSON task workflows and limited task-status SSE	Does not imply support for every SDK, gRPC, or push transport
Framework runtime	Fixed LangGraph 1.2.12, digest-locked dependencies, and a verified Linux rootfs	Requires isolated execution evidence; not arbitrary framework or version hosting
Observability export	Committed trace metadata through OTLP/HTTP JSON 1.9.0	Excludes prompt and answer bodies; not every OTLP transport implementation

Discovered capabilities require explicit approval and exposure. Rediscovery may change a contract and invalidates old approval; execution must use a matching current version. A descriptor-conversion adapter alone does not demonstrate end-to-end certification of a named external product’s native versions.

Release and Skill packages can undergo server-side signature verification when an independently trusted public key is available. Receiving a package and approving its activation are separate states. A cooperative execution wrapper controls participating processes; it cannot promise termination of arbitrary remote processes. Before connecting their own tools, customers should establish idempotency, timeout, audit, rollback, and actual side-effect reconciliation mechanisms.

11. Model routing, capacity, usage, and cost

A model Profile stores specific configuration and revisions. Actual probes distinguish text, streaming, structured output, tools, images, reasoning parameters, usage, and cancellation capabilities. A model name or an address resembling `/v1` proves none of these. Server-validated `TEXT_JSON` and `NATIVE_STRICT` tested against an exact business schema are separate contracts. [R12]

A controlled read-only tool round illustrates deliberately bounded execution: one currently registered read-only GET tool, human registration and per-action authorization, at most two model requests, one tool call, no repair attempts, and one overall deadline. GET alone does not prove that a tool is read-only; its semantics need review. This scope does not establish that arbitrary autonomous tool loops are safely enabled.

Model gateway and direct routes may coexist by Profile. A shared inference resource needs one authoritative capacity coordinator, with registered participant identities, endpoint aliases, and Profile versions. Independent locks in separate databases do not create a global queue. The originating side checks business-source authority; the coordinator checks central capacity and upstream permission. Neither grants access to another database's private sources. [R13]

A client disconnect establishes local cancellation, not that the Provider stopped computing. An unknown request cannot release capacity and be resent merely because its lease expired. Its actual completion must first be reconciled. Complete transport does not establish business acceptance; refusals, truncation, and validation failures remain separate outcomes.

The usage ledger retains provenance, request digests, Provider-reported input and output, and available cache or reasoning dimensions. Timing distinguishes queueing, first content, request duration, and total duration. Missing usage is incomplete rather than zero. Unknown prices or incomplete observations leave cost unknown. Platform estimates are not supplier invoices; Enterprise allocation requires explicit scope and complete allocation percentages. [R14]

Metadata ledgers do not contain prompts, answers, or hidden reasoning bodies. Explicitly configured business-content, evaluation, or encrypted replay stores require separate management. Read-only dashboards show authorized agent, session, Task, and Loop information and token and cost summaries over a bounded period. Unavailable aggregates are null or degraded, not invented zero consumption. External direct traffic without trusted adapter evidence may remain outside coverage.

12. Security, governance, and compliance responsibilities

Final transport and output boundaries

The v5.0.4 common native transport checks the complete text of final Provider JSON and sends the same serialized bytes. Complete output validation continues. This reduces differences between checked input and transmitted input. Hosted-image pixel semantics and independent DispatchTicket byte auditing retain explicit limits. These checks establish neither comprehensive DLP nor equivalent coverage across all adapters. [R22]

Chuanxu combines application checks with database-native boundaries. Sessions, CSRF protection, short-lived agent credentials, resource authorization, security domains, and source checks serve different purposes. Web source-IP rules cover IPv4 and IPv6; emergency loopback login still requires authentication. Secrets stay in server-side encrypted configuration or restricted files. Clients cannot issue themselves authority. [R5]

PostgreSQL uses a restricted Owner, dedicated login roles, identity mapping, and forced RLS, rejecting bypass through superuser access, BYPASSRLS, or Owner fallback. Oracle depends on native End User, context, Data Role, and Data Grant prerequisites, including in Community. YashanDB uses separate users, object grants, restricted projections, and session identity rather than copying Oracle security semantics. [R16]

Enterprise adds dedicated approvals, audit management views, LDAP, governance, and compliance modules for responsibilities such as separation of duties, policy, retention, legal hold, and scoped export. Common paths still write necessary audit records; Community should not be described as having no audit. Common independent review of knowledge candidates is also distinct from Enterprise approvals. Reserved type names such as OIDC or SCIM are not delivered connectors.

Runtime isolation depends on actual host evidence: dedicated identities, namespaces, cgroup v2, resource limits, seccomp, bubblewrap, LSM, a read-only root, and network controls. Execution is refused when requirements are missing. Default-deny egress has been verified; selective outbound allowlisting remains refused where implementation and evidence are absent. Reference container or VM lifecycle adapters return unverified status rather than demonstrating automated creation and verification. [R17]

Application-process isolation does not cover attackers with host-root or infrastructure-administrator privileges. Database clusters, operating systems, certificates, key custody, backup, networks, and incident response remain shared customer and implementation responsibilities. A compliance agent may propose actions and evidence but inherits no administrator authority. Technical controls do not constitute regulatory authorization or independent certification.

13. Database adaptation and deployment prerequisites

The three backends follow common business-object and authorization goals through native adapters. Selection should consider existing database capabilities, licensing, DBA experience, data scale, and governance requirements. Similar logical interfaces do not imply identical indexes, execution plans, or throughput. [R1, R16]

Backend	Typical isolation and preparation	Deployment checks
Oracle	Dedicated PDB, application Owner, native identity and grant preparation	Required versions and capabilities such as Text and vectors; DBA scripts; restored objects, constraints, and grants
PostgreSQL	Dedicated database, non-superuser Owner, login roles, and forced RLS	Extensions and indexes, session identity, Owner privileges, and negative authorization cases
YashanDB	Dedicated PDB, users and object grants, restricted data projections	Matching clients and drivers, retrieval capabilities, session identity, and native permission differences

The packaged runtime baseline is x86_64 Linux with glibc 2.34 or later, using RHEL / Oracle Linux 9.8 or later or separately verified compatible combinations. Deploy with CPython 3.14 matching the ABI of the packaged wheels and database drivers. A higher Python version does not automatically provide binary compatibility. [R18]

Packages can include offline dependencies and database clients. Community installs only the selected backend's dependencies. A DBA performs privileged preparation before runtime; SYS or root credentials do not belong in application runtime configuration. Narrow account-status views required for Oracle and YashanDB are explicitly prepared by the DBA, rather than replaced with broad system-catalog grants. Configuration and key files use restricted permissions and are preserved during replacement.

Models, embedding services, and external tools may reside at customer-approved endpoints. Whether data may leave the deployment boundary depends on the actual Profile, source-disclosure policy, network, and contractual arrangements. Private installation alone does not establish that every model call is local. Actual routes and send evidence must be checked.

14. Installation, upgrades, operations, and recovery

Fresh installation initializes an explicitly empty target using packaged SQL. It does not rely on model-generated schemas or automatically enable sensitive capabilities. Before use, verify the actual product, database, runtime, and configuration, together with database identity, migration state, and negative authorization cases. [R18]

Before upgrading, back up and confirm recoverability. Then perform preflight, execute the supported migration chain, and verify existing data, constraints, permissions, and application behavior. Applied historical SQL and checksums remain immutable; corrections use new migrations. Partial-execution recovery must follow exact migration journals. Schema comparison aids diagnosis but cannot replace an actual upgrade, and a terminal completion message alone is insufficient.

`/api/health` reports process liveness; `/api/ready` checks database and control-plane readiness. They serve different purposes. Operational observations should link Tasks, attempts, model requests, tool receipts, leases, and migration status. Unknown execution, invalid sources, insufficient budgets, and database unavailability need separately explainable states and procedures.

Beyond database backups, protect encrypted configuration, keys, approved Skills or runtime images, and migration and signature-verification records. Recovery exercises should verify identities, exact knowledge revisions, authorization, task state, and external side effects rather than merely opening the restored database. Recovery point and recovery time objectives depend on the actual architecture and exercise results.

Application multiprocess execution and lease recovery do not provide database-cluster high availability. Replication, failover, load balancing, certificate renewal, capacity planning, and disaster recovery require explicit deployment design. An old executor that has lost its lease cannot publish success. Unknown external operations must be reconciled before retries so that recovery does not duplicate business actions.

15. Editions, licensing, and learning material

Product	Database choices	Common capabilities	Enterprise modules	License
Chuanxu Community	Oracle / PostgreSQL / YashanDB; one per deployment	Knowledge, memory, tasks, collaboration, common execution controls	Excluded from the package	Apache-2.0
Chuanxu Oracle Enterprise	Oracle	Common capabilities and native adaptation	Included	BSL-1.1; commercial license for production
Chuanxu PostgreSQL Enterprise	PostgreSQL	Common capabilities and native adaptation	Included	BSL-1.1; commercial license for production
Chuanxu YashanDB Enterprise	YashanDB	Common capabilities and native adaptation	Included	BSL-1.1; commercial license for production

Enterprise-specific modules are approvals, audit, ldap, skill_token, orchestrator, governance, and compliance. Community excludes them during the build. Common execution controls include shared Task, Graph, and Loop capabilities. The word “orchestration” alone should not be taken to mean that all graph execution is Enterprise-only. Community still requires the applicable native database identity and authorization prerequisites. [R15]

Version 5.0.3 provides nine complete bilingual platform learning units. Community retains learning entrypoints and four brief product topics but does not bundle or automatically install the nine complete units. An authorized administrator may explicitly import product-, version-, and backend-matched JSON companions delivered outside the software archive. Enterprise bundles and automatically deploys the full material. Upgrades preserve installed knowledge, historical snapshots, and user edits rather than silently deleting them. [R19]

Learning uses role- and goal-oriented catalogs. Agents discover metadata first and load exact bodies by digest. Reading guidance or a Skill grants no execution permission. JSON digest consistency is not independent publisher-signature authentication.

Community is governed by Apache-2.0, including applicable notices and third-party licenses. Enterprise use follows the delivered BSL-1.1 and commercial contract. The current license specifies May 28, 2028 as the Change Date and Apache-2.0 as the Change License. Database, model, client, and dependency licenses require separate review. This paper adds no support obligations and does not present possible future delivery formats as existing capabilities.

16. Illustrative workflows

The following are explanatory scenarios to be verified with a customer’s own material, models, and permissions.

Technical support and operational knowledge. A Human submits a question with product and version. An agent retrieves applicable manuals, approved changes, and incident records. Its answer retains names, versions, source locations, and prerequisites, seeking clarification when evidence conflicts. Diagnostic tools begin with controlled read-only actions; production changes follow separate approval and external execution. Evaluation can examine time spent finding sources, evidence coverage, and human review effort.

Continuity across a development investigation. An initial agent creates a Task and records reproduction, rejected approaches, and unresolved issues. A Handoff passes exact sources and approved decisions to another agent, which reads under its own identity and continues after acceptance. Work is accepted against actual tests and action receipts. Reusable conclusions enter knowledge candidates; unconfirmed inference remains in working notes.

Policy and procedure questions across teams. Organizations may share an entrypoint while retrieving only material within their permissions. Previous policy versions, effective dates, and exceptions remain explicit. Channel membership does not expand policy access. Conclusions concerning compliance interpretation or rights and obligations require designated human confirmation, supported by inspectable evidence and action records.

Controlled automation with human participation. Existing workflows become Tasks or versioned Graphs. Each tool, model, database, and human node has defined input, approval, budget, and result criteria. Unknown outcomes from non-idempotent actions prompt external-state checks before an authorized person decides on recovery or compensation. Automation can expand gradually from observable, reversible actions.

The common value hypothesis is less repeated information transfer and status checking, with more inspectable responsibility, permissions, and sources. Actual cost and efficiency improvements must be established against a real business baseline and pilot results.

17. Verification approach and evidence boundaries

Chuanxu evaluates platform data correctness separately from model semantics. Permissions, source versions, time, task state, actual input, citations, and publication are controls that can be checked individually. A model can still omit conditions or infer incorrectly from correct input. Successful transport cannot replace semantic acceptance. [R20]

Version 5.0.4 verification covers incident sources, report candidates, context assembly evidence, final transport and output checks, and fourteen fresh-install and real-upgrade paths across five artifacts and seven database targets. Dependency review retains applicable retrieval, intake, learning, and authorization results, with affected paths checked separately. This does not mean every final file underwent every test again.

Same-turn preparation measurements cover explicit-source and mixed-retrieval paths while retaining authorization checks at use. Local timings establish no Token, cost, or general performance benefit and enable no new query/TTL cache. Shared encoding has an experiment design and remains disabled. Generated summaries still need source-by-source review: structural validity establishes neither facts, root causes, nor business acceptance. [R22]

Retrieval checks use frozen questions and distractor sets of different sizes. They examine whether required source text is recalled, conditions reach bounded context, citations are exact, and unauthorized material remains undisclosed. Results establish behavior for the selected corpus and configuration, not coverage of all customer documents or arbitrary capacity. Misses in the keyword-only comparison and semantic failures and omissions in model answers remain recorded.

Evidence category	What it can establish	What it does not establish
Native permission and business-path checks	Allows, denials, and state changes for selected identities and workflows	Independent security audit, complete threat coverage, or regulatory certification
Frozen retrieval and context evaluation	Source, condition, citation, and authorization behavior on specified inputs	Correctness for every question or equivalent backend performance
Fresh installs and stable-state-copy upgrades	Actual behavior for selected sources, migration chains, objects, and grants	Direct upgrade from every historical customer database or full physical PDB equivalence
Automated browsers and credential scripts	Executable interface paths, identities, and action receipts	Human novice research or autonomous end-to-end agent validation
Model request and output checks	Actual inputs, transport, and specific outputs for the tested configuration	Comprehensive production-model evaluation or complete semantic acceptance

The current release conclusion holds within the approved scope and retains model errors and unevaluated items. Customers must verify their own database versions, permissions, data, models, concurrency, and operating conditions. This paper promises no universal latency, throughput, token-saving percentage, or SLA. Internal verification observations should not become procurement performance commitments.

18. Proof of concept, selection, and implementation

A proof of concept should begin with a scenario that has an accountable owner, available source material, and verifiable outcomes. Agree acceptance criteria before configuring the product and environment. The ability to hold a conversation does not establish operational suitability. The following phases are recommendations, not a fixed schedule. [R21]

Phase	Main work	Suggested exit condition
Scenario and data preparation	Goals, owners, classification, authorization matrix, baseline, and question set	Usable material and licenses; agreed expected outcomes and denial cases
Environment and control checks	Selected backend, identities, RLS or native permissions, model routing, and revocation	Unauthorized reads denied; model disclosure and task evidence traceable
Complete business workflow	Intake, answers, Tasks, handoffs, human approvals, unknown outcomes, and recovery	Sources, responsibility, results, and exception handling are verifiable beyond the answer
Capacity and production review	Actual scale and concurrency; upgrades, backup, failure, and recovery exercises	Agreed quality, capacity, operational, and licensing criteria met

Measure retrieval recall, exact citations, condition retention, authorization denials, semantic correctness, and human review effort. Performance measurements should separate database queries, embeddings, queueing, first content, complete answers, and external tool time. Fix the corpus, questions, version, and model configuration; report distributions, failures, and timeouts rather than only the best run.

Total cost includes database and software licenses, compute and storage, models and embeddings, operations, knowledge governance, and human review. Ranking without additional generation is a design fact. Return on investment still depends on actual workload, quality, and risk changes. Shorter text alone does not establish equal-quality cost savings.

The enterprise owns business goals, data legitimacy, personnel and resource authorization, risk acceptance, and production decisions. Implementers own agreed adaptation, deployment, migration, and operational verification. DBAs and security teams own native boundaries and infrastructure; business reviewers confirm knowledge and outcomes. Selection should confirm Enterprise governance needs, backend prerequisites, upgrade sources, support scope, and contracts. A demonstration cannot replace production review.

Appendix A: Glossary

Term	Meaning in this paper
Human / Principal	A human user; a Principal is the shared identity concept for authentication and authorization
Agent	An executing principal linked to responsibility, an instance, and current authorization
Security Domain / Channel	A collaboration authorization boundary / a conversation space; neither substitutes for the other
Knowledge / Memory	Governed reusable knowledge / source-preserving temporal memory entities
Task / Step / attempt	Work, a step, and a particular execution attempt
Work / Handoff	Responsibility and acceptance object / receipted transfer of responsibility
Graph / Loop / Harness	Versioned execution graph / bounded iteration / execution and acceptance blueprint
Property-graph projection	A relationship view derived from authoritative entities; it grants no resource access
Profile	A versioned model configuration with actual capability evidence
Lease / fencing	A time-bounded execution right / rejection of writes from expired executors
DB4A2A	Collaboration through database references to authorized exact context
MCP / A2A	Tool and context integration / inter-agent task protocols, supported within explicit versions
Digest / signature	Content consistency identifier / provenance verification requiring a trusted key
RLS / PDB	Row-level security / pluggable database, used as appropriate to the backend
UNKNOWN / UNOBSERVED	Unknown outcome / absent trusted observation; neither implies success

Appendix B: Sources and further reading

The following materials are organized by subject. Consult the applicable files in the selected v5.0.4 product package. Chinese guides generally add `_zh` before `.md`. Filenames help offline readers locate delivered documentation without requiring access to development systems.

ID	Material	Purpose
R1	`architecture.md`	Architecture and authoritative data and permissions
R2	`channel-knowledge.md`	Retrieval, context, and source checks
R3	`answer-quality.md`	Evidence sufficiency, source categories, and answer boundaries
R4	`knowledge-intake.md`, `curated-knowledge.md`	Guided intake, candidates, independent review, and publication
R5	`security.md`	Human and agent identities, sessions, domains, and access
R6	`memory-working-notes.md`	Temporal memory, notes, revisions, and retrospectives
R7	`agent-context-assets.md`, `workspace.md`	Exact context, budgets, and branches
R8	`graph-engineering.md`, `loop-engineering.md`, `harness.md`, `processing-recovery.md`	Graphs, loops, execution contracts, and recovery
R9	`continuity-operations.md`	Work, handoffs, and current source authorization
R10	`db4a2a.md`	Collaboration using database context references
R11	`agent-extensions.md`, `mcp-versioned-interoperability.md`, `framework-agent-gateway.md`, `release-signing.md`	Extensions, protocols, frameworks, and signature boundaries
R12	`model-workflows.md`	Profiles, structured output, and controlled tool rounds
R13	`model-endpoint-capacity.md`	Shared capacity, request lifecycle, and unknown outcomes
R14	`model-usage-and-wallboard.md`	Usage, cost, and observability coverage
R15	Selected package's `LICENSE`, edition and licensing documentation	Community and Enterprise capability and usage scope

ID	Material	Purpose
R16	`minimum-privileges.md`	Native permissions and DBA preparation
R17	`runtime-isolation.md`	Host requirements, isolation evidence, and refusal behavior
R18	`deployment.md`, `python-runtime.md`, `linux-platform-compatibility.md`	Installation, upgrades, dependencies, and platform prerequisites
R19	`platform-learning-delivery.md`	Learning entrypoints, separate JSON, and upgrade preservation
R20	v5.0.4 release notes and delivery-scope documentation	Current features, verification approach, and retained boundaries
R21	`poc-readiness.md`	Scenario, deployment, and acceptance preparation
R22	Current incident and context documentation; separate research material in `docs/v5.0.4/enterprise-incidents/`, including the template, ATTRIBUTION, and PagerDuty license	Official sources, report drafts, assembly evidence, measurements, and final transport limits; separate research is not bundled software content

Licenses, third-party notices, and commercial agreements have their respective scope. Oracle, PostgreSQL, YashanDB, LangGraph, and other names belong to their respective rights holders. Describing compatibility does not imply endorsement or certification by those organizations.