EGNYTE

The Context Layer

Turning Enterprise Content Into Context That AI and Agents Can Act On

Table of Contents

Introduction

Large language models (LLMs) provide powerful reasoning and generation capabilities, but they don't inherently understand how an organization's information is structured, related, governed, or used. For example, a new hire spends an entire afternoon hunting for the current contract. They finally find a document that looks right, but aren't sure it is the most current version. If the new employee searched for the document using an AI application, whether built internally by the organization or provided by a third-party service such as Claude or ChatGPT, they could surface the relevant contract in seconds. However, without access to reliable enterprise context, the resulting answer may be incomplete, outdated, or incorrect.

Enterprise AI applications, such as AI Assistants and Agents, therefore require a context layer that transforms distributed enterprise information into structured, connected, and permission-aware context that LLMs, AI applications, and agents can consume.

The gap is widest for autonomous agents and process workflows. An agent evaluating an RFP can't repeatedly scan repositories, identify comparable bids, determine applicable compliance requirements, and reconstruct relationships. Without this context, agents are more likely to hallucinate, consume more tokens, and incur higher latency and costs, delaying AI ROI. Agent performance depends on the quality, completeness, and relevance of context available at runtime.

That's where business context comes in, capturing how information relates to people, processes, entities, and decisions.

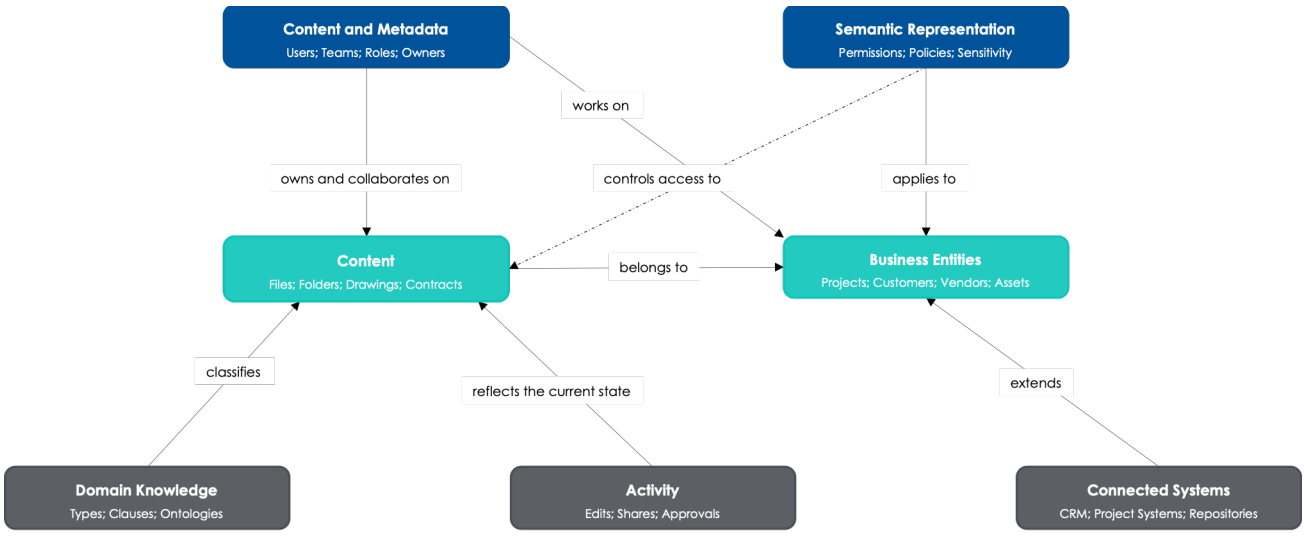
What Makes Up Business Context?

Business context is a machine-readable representation of the information, semantics, relationships, activity, and governance needed to interpret enterprise data and execute tasks correctly. It's assembled from core components, or building blocks of business context, that Egnyte identifies, resolves, and connects across enterprise information sources:

- **Content**: Files, folders, documents, drawings, and contracts, including formats that are difficult for AI systems to interpret directly, such as CAD drawings, BIM models, scanned submittals, images, PDFs, email and chat conversations (Eg; Slack)
- **People**: Users, teams, roles, owners, and collaborators—these attributes establish who created, approved, owns, or is responsible for the information

- **Business entities**: Projects, customers, vendors, contracts, and assets—entities represent the real-world objects described in content and are resolved to consistent identities across sources
- **Relationships**: Typed, directional connections such as belongs to, works on, references, approved by, supersedes, and similar to—these connections establish how content, people, and entities relate
- **Domain knowledge**: Document types, concepts, clauses, classifications, and ontologies that provide industry-specific semantics, such as what constitutes an RFI, submittal, or trade
- **Activity**: Edits, shares, approvals, and interactions that provide temporal signals about what is active, finalized, or no longer relevant
- **Connected systems**: CRM, project management, productivity applications, and other repositories that provide context beyond the file system and evolve independently
- **Governance**: Permissions, policies, sensitivity, retention, and compliance controls that determine what information can be accessed, used, and retained, by whom and under what conditions.

No single component is sufficient on its own. A vector index over content can retrieve a document that mentions concrete foundations, but it can't determine which bid was awarded, who approved it, which specification superseded it, or whether the requester is authorized to access it. Context emerges when these components are connected and continuously updated, and it's why the diagram below represents relationships as edges connecting the other components rather than as a standalone component.



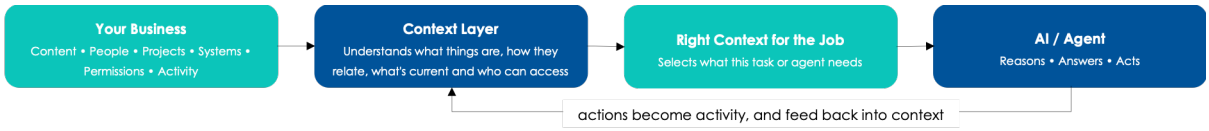
Relationships are represented as edges, not standalone components – belongs to, works on, references, approved by, supersedes, and similar to.

Business context as a relation graph

The Context Layer

We've covered *how* enterprise content becomes structured understanding. The harder question is *what* that understanding is for. Business context is never static. Documents change, permissions shift, projects close, and people change roles.

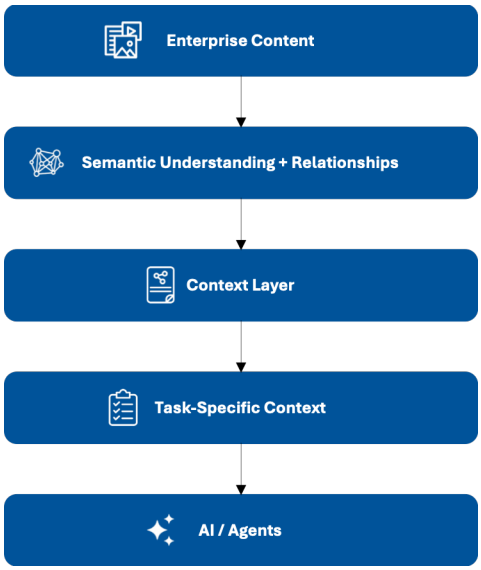
Egnyte's Context Layer is a persistent context system, not a static processing pipeline. It continuously derives context from enterprise content, keeps it current and governed, and makes it available through open APIs and MCP. It gives AI the right business context to work accurately by understanding what things are, what information matters, what's current, and who can access it. It helps AI avoid outdated or irrelevant information, respect permissions, and provide traceable answers.



The same governed context powers internal AI apps, third-party agents, and custom applications, so each doesn't have to independently determine what's relevant, current, and permitted. For example, an agent evaluating a construction RFP can connect the client, requirements, past bids, pricing benchmarks, document relationships, current versions, and permissions, helping it reaches faster, more accurate, and better-informed decisions.

A worked example demonstrates how an RFP evaluation agent uses Egnyte's pre-built Context Layer to compare a new bid with prior engagements, historical pricing, and compliance requirements, improving measured accuracy from 63% to 80% across a 266-document construction corpus (source: Egnyte internal RFP evaluation study).

The Context Layer moves information through five stages:



- The first two stages build context.
- The third holds and maintains it.
- The last two spend it by narrowing the whole to what one task needs, incorporating personalization based on user profile, activity, firm memory, and organizational vocabulary, then delivering it to the model or agent that asked.

Enterprise Content

The layer starts with an inventory, not a query. When a folder or repository is connected, the system creates a structured record of its files (indexes) and metadata before adding deeper context or meaning. This establishes the initial corpus boundary—for example, a construction bid containing 76 files across 25 folders, or an HR repository containing 26,483 files across 1,317 folders.

The inventory captures the scope against which subsequent ingestion, enrichment, entity resolution (different records that refer to the same project/domain), relationship extraction, and indexing. By establishing a complete baseline first, the system can detect missing objects, track processing state, and measure coverage and freshness as the Egnyte Context Layer evolves.

Parsing: Format-specific parsers process enterprise file types including PDF, Microsoft Office, images and OCR, email, and CAD. Content is segmented into structure-aware chunks rather than fixed-length windows, preserving semantic relationships such as clauses with their exceptions and specifications with their parameters. Object-level metadata—including author, dates, path, document type, project ID, and custom attributes—is captured alongside content.

Hard formats: Generic AI pipelines are built for documents. Egnyte goes further, bringing AI understanding to the complex technical content that powers industries such as AEC. DWG, Revit, BIM, CAD, and geospatial files are transformed into AI-readable representations, while images are enriched with captions and classifications. This draws on Egnyte's deep experience with specialized industry content, turning files that AI traditionally struggles to interpret into searchable, connected, and usable business context.

Event-driven processing: Processing is triggered by content changes rather than scheduled repository-wide crawls. A change to a single clause reprocesses the affected document and impacted relationships without reprocessing the entire repository.

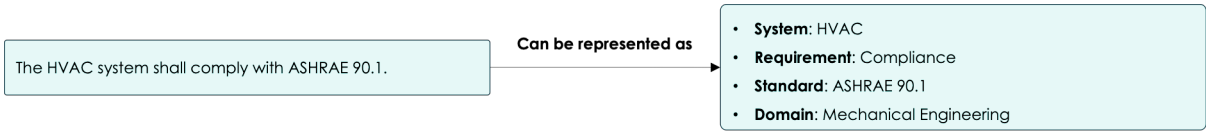
Eg: Specification.pdf → Project: NS Tower → Discipline: Mechanical → Revision: 03 → Author: ABC Engineering

Ingestion, parsing, and metadata capture

Semantic Understanding and Relationships

This stage operates at two levels, determining what content means and establishing what it connects to. Semantic understanding enables relevant retrieval, while relationships transform those signals into context by connecting content to entities, events, people, projects, and other business objects.

Meaning: Embeddings represent the meaning of content, such as documents, sentences, or phrases, helping AI find relevant information even when different words are used. For example, "foundation pricing" can retrieve content describing concrete substructure costs without using the same terms. Classification models identify document types and attributes, while extraction models identify obligations, clauses, dates, values, and technical specifications.

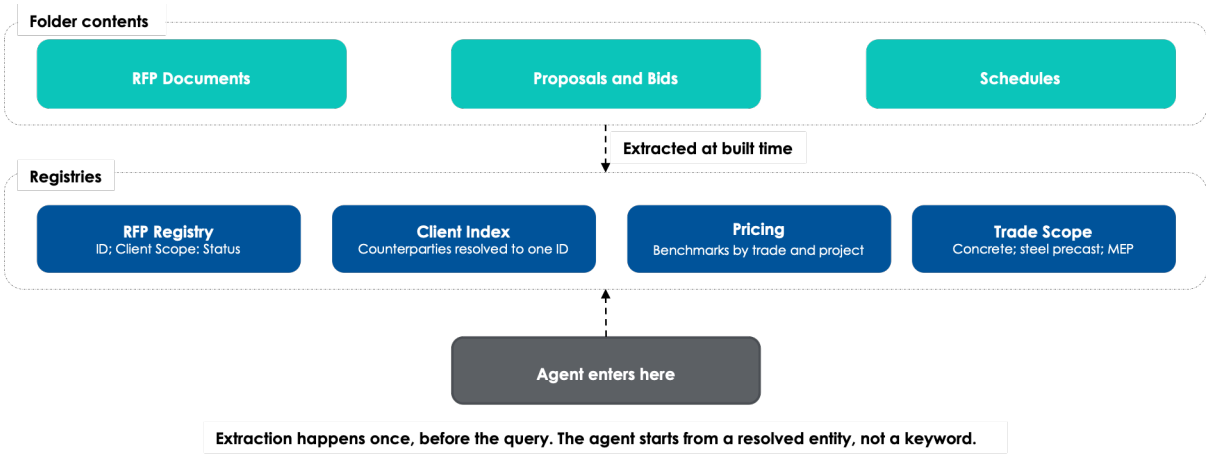


Semantic enrichment. embeddings, classification, and extraction

Entities and resolution: Named entity recognition identifies vendors, project codes, criteria, dates, and other business entities. Entity resolution determines when references point to the same real-world entity. For example, a vendor mentioned in an RFP, scorecard, and signed contract is treated as one vendor, creating a consistent foundation for connecting related information. These resolved entities become stable anchors for relationships and graph traversal.

Precomputed registries: The Context Layer organizes extracted data into structured business context during processing, rather than assembling it for each query. This gives agents direct, query-ready access to key business objects. For a construction bid folder, these can include:

- **RFP registry**: ID, client, scope, and status for each solicitation
- **Proposal registry**: Bid, value, outcome, and scope across submissions
- **Client index**: Counterparties resolved to a single identity
- **Trade scope taxonomy**: Concrete, steel, precast, MEP (mechanical, electrical, and plumbing), automation, and civil
- **Pricing benchmarks**: normalized by trade and project type
- **Timeline patterns**: mobilization, build, closeout, and bid-to-award intervals



Entity extraction: Raw folders are resolved into registries the agent can navigate

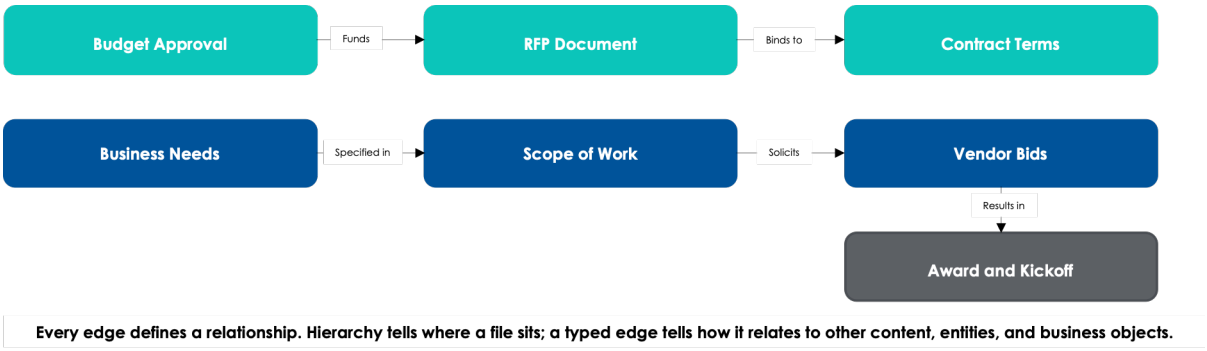
Typed relationships: Folder hierarchies provide both structural and semantic context, such as AEC project folders or client folders in wealth management. The system builds on these existing signals to identify deeper relationships, such as which vendor response addresses an RFP, which evaluation references specific compliance criteria, and which schedule supports a plan.

These relationships are established by analyzing content, metadata, document structure, and other signals, then validated against available evidence. Each relationship carries confidence and

provenance, showing why it was identified and where the supporting evidence came from. This allows agents to use connected information while providing a traceable basis for decisions.

In a construction project, these relationships are represented as typed edges with explicit semantics:

- A project budget constrains contracts and agreements
- Contracts authorize the work plan, which is tracked by the project dashboard
- Design drawings are detailed by specifications and coordinate with mechanical and electrical engineering
- Specifications form the basis of bidding, while addenda initiate updates to construction administration

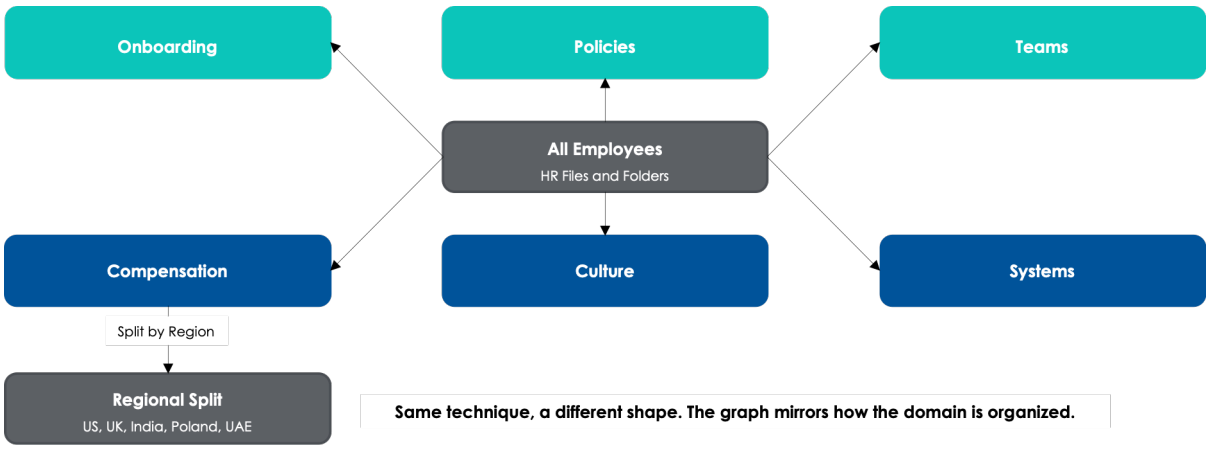


RFP project folder: Edges carry meaning, not just hierarchy

Apply the same method to an HR repository, and the resulting structure is entirely different:

- Content clusters around the employee lifecycle, including joining, employment, compensation, culture and communications, and systems
- Regional patterns recur across benefits, payroll, and time off, spanning the U.S., U.K., India, Poland, and UAE
- Policies form a central node that governs these areas

The method is domain-agnostic, but the resulting graph reflects the domain's entities, relationships, and business processes.

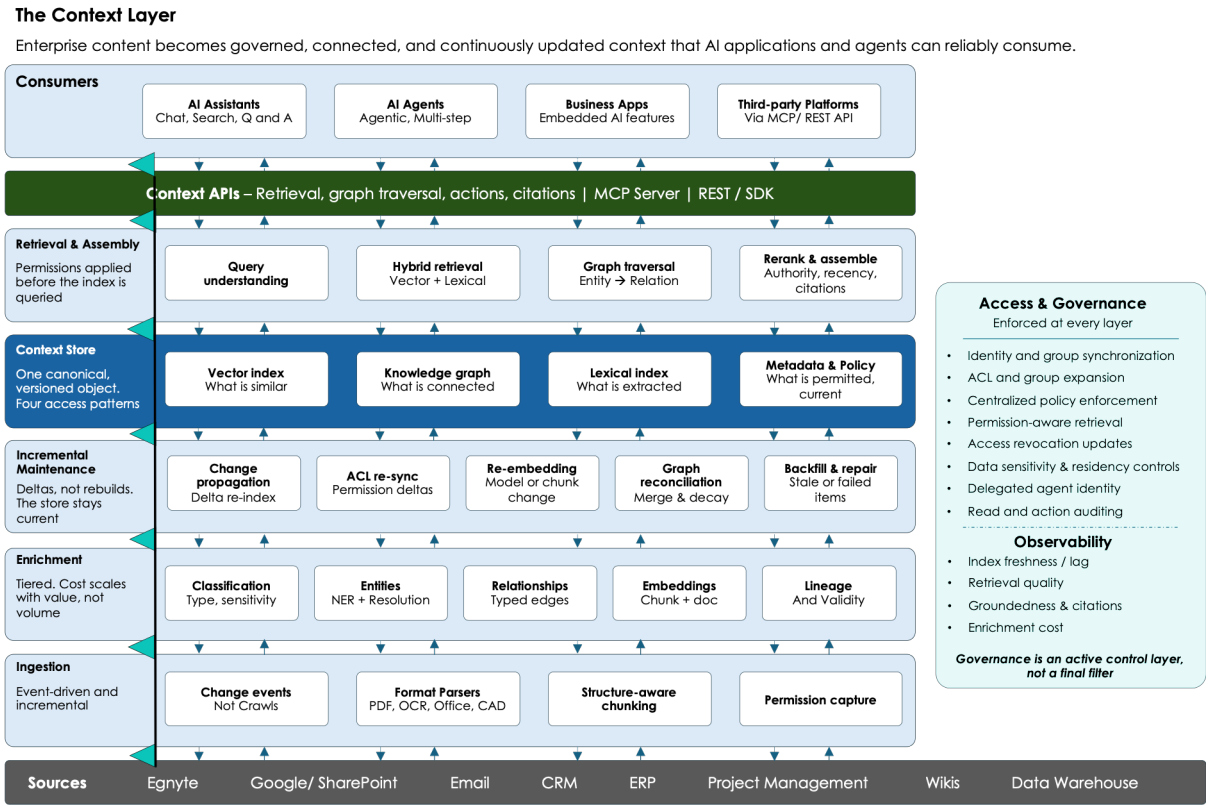


The same method over an HR repository—a different shape from the same technique

The Context Store

The graph and its indexes are persistent and continuously maintained. They're updated incrementally as content changes, allowing agents to navigate an existing context structure rather than performing a full scan for each interaction. Different retrieval needs require different index types, all maintained against a canonical model, a shared representation of the underlying content, entities, relationships, metadata, and permissions. This keeps each index aligned to the same underlying objects and relationships:

- Vector indexes for semantic similarity
- Knowledge graph for entities and typed relationships such as references, approved by, supersedes, and works on
- Lexical indexes for exact terms and identifiers such as project codes and drawing numbers
- Metadata and policy stores for state, version, and access
- Memory for persistent user and team context



The context store

The **canonical model** is the architectural center of the context layer, providing a common representation that keeps the graph, vect₹ors, indexes, metadata, policies, and memory aligned. It also tracks how information was established, where it came from, and how it should be interpreted across different systems. **Lineage** shows how a fact was derived, including the transformations and relationships that produced it, while provenance links it back to its original source, document, system, or event. **Temporal state** tracks when information is valid and whether it's draft, approved, executed, superseded, or archived.

Together, these capabilities ensure agents work from consistent, current, explainable, and traceable context while reducing ambiguity, duplication, and conflicting interpretations.

Lineage Record Transformation Path

Source document → Text extraction → Entity extraction → Entity resolution → Relationship inference →Derived business fact

Provenance connects a generated fact back to its supporting source. For example:

Fact: HVAC specification = Revision 03

Source: Specification.pdf; Page 42; Revision field; Modified: July 18, 2026

Lineage, provenance, and temporal state

The Context Layer is designed to remain current, maintainable, and transparent as enterprise information changes. Two capabilities support this: incremental maintenance and observability.

Incremental maintenance: When a document changes, only affected content, entities, and relationships are recomputed rather than rebuilding the entire repository. This makes continuous updates practical at repository scale.

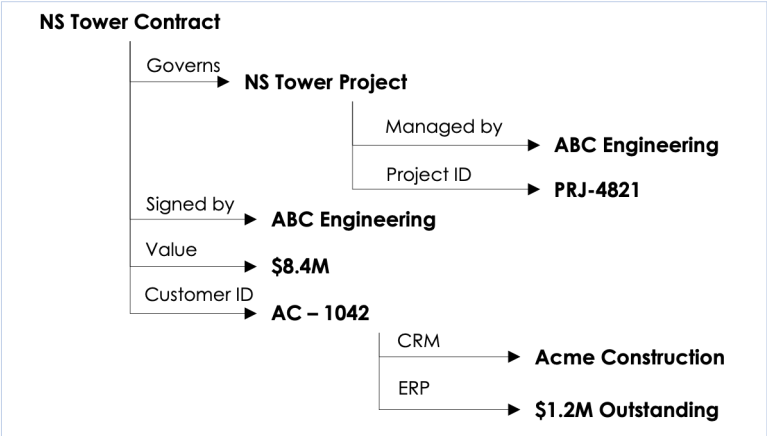
Observability: Context is built and maintained ahead of queries, making it possible to inspect, measure, and improve. Emerging capabilities will administrators visibility into what AI knows, highlights gaps or incomplete context, and shows the sources behind it. Administrators will be able to enrich, classify, extract, or link content to improve the context. This creates a transparent, measurable foundation for AI, rather than leaving context as a black box.

Task-Specific Context

The Context Layer contains far more information than any single task or agent needs. The key isn't creating more context, but assembling the right context for the job. Based on the task, the layer selects the relevant content, people, relationships, and business knowledge to create a focused context map for the agent. This gives each agent the information it needs to perform its specific job without overwhelming it with irrelevant context.

Task scoping: The system interprets the agent's instructions and cross-references them with repository content, shaping the working set around the task rather than the query terms alone. An RFP evaluation agent and a compliance audit agent working from the same folder retrieve different context.

Permission-aware retrieval: Effective permissions are applied before retrieval begins, so search and AI work only with content the user is authorized to access. This avoids retrieving broadly and filtering results afterward, which can remove relevant results and weaken ranking. Pre-filtering produces a complete ranking from content the user is authorized to access. Egnyte keeps permissions connected to the business context used by AI and agents, maintaining a consistent security boundary from retrieval through action.



Example: permission-aware retrieval; filtering before and not after ranking.

Relationship trraversal: Starting with the most relevant results, the system follows related content and entities to find information that may not use similar words. For example, it can connect a specification to the contractor responsible for it, the addendum that changed it, and the earlier bid used to price it. This finds connections that a standard search may miss.

Temporal filtering: The system tracks when information was valid and whether it was draft, approved, or superseded. This lets it answer questions like which specification was in effect when a project was bid, rather than simply returning the latest version.

Memory and personalization: Memory retains useful context from interactions, such as resolved entities, decisions, and corrections, so agents can build on what's already been established rather than starting from scratch. This context can be scoped to an individual, team, or organization, depending on the use case. Memory helps prioritize relevant information but doesn't change access rights. Permissions are always checked when content is retrieved, ensuring memory can't expose information the user is no longer authorized to access.

The output is a compact, ordered set of relevant evidence, with its sources and validity attached. This is the context delivered to the model or agent.

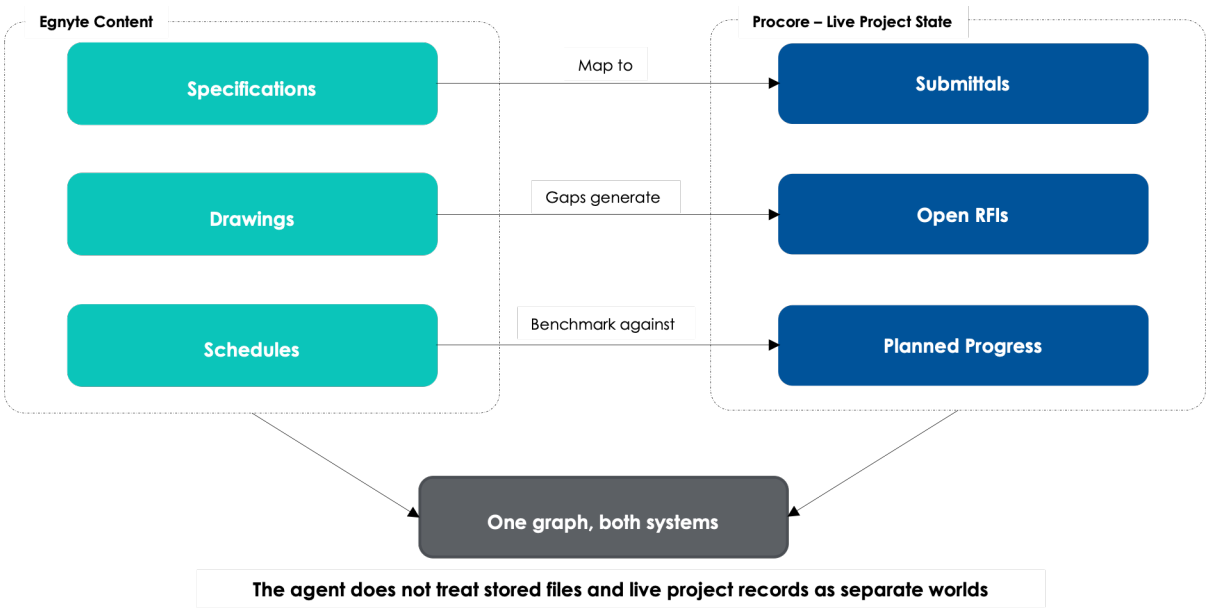
AI and Agents

The Context Layer makes enterprise context available to AI applications and Agents to understand information in relation to the people, projects, entities, and business processes they support. This context can be accessed by Egnyte AI as well as external applications through open interfaces.

Delivery: Context is delivered to Egnyte AI and external applications through open APIs and MCP. Applications consume an existing, maintained representation of business context instead of rebuilding relationships for every interaction.

Governed action: Agents operate with delegated identities tied to the user's permissions, with authorization checked at execution time. An agent cannot gain access beyond what the user can access. Reads remain permission-filtered, sensitive actions can require approval, and agent activity is logged for audit and governance. Agents have distinct identities that make their actions identifiable and auditable, while access to content and actions is governed by applicable permissions. Authorization is checked when an agent performs an action, preventing it from accessing or changing information beyond its allowed scope.

Connected systems: When an agent connects to systems such as Procore, Salesforce, or ServiceNow, the Context Layer extends beyond files to live business records. Egnyte content can be connected to projects, RFIs, submittals, meetings, and incidents, allowing relationships to surface across systems. A drawing can be linked to an open RFI, a specification to its submittal list, or a schedule to planned project progress. The agent can then reason across documents and live system data as one connected context rather than as separate sources.

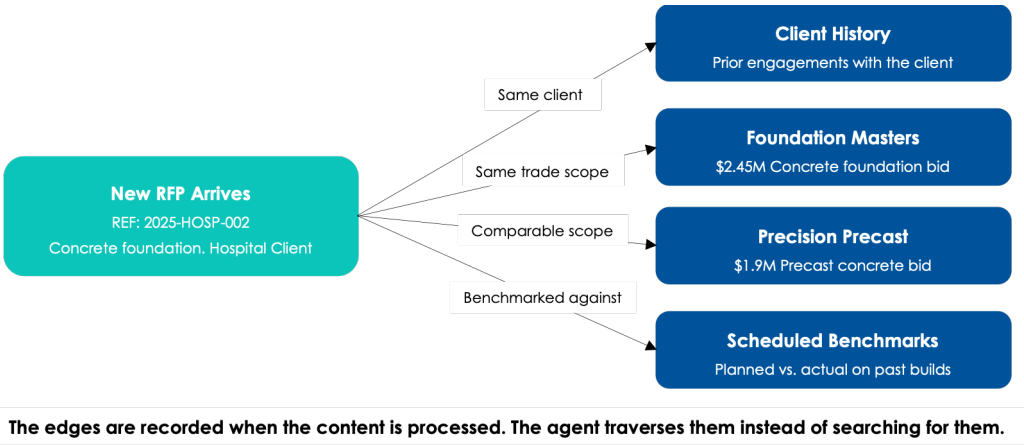


One graph across Egnyte content and live Procore project state

Worked Example: RFP Evaluation

A construction firm receives a high volume of RFPs. Evaluating each one has historically taken days, requiring teams to cross-reference requirements, verify compliance language, and compare vendors using information spread across spreadsheets and prior bid folders.

Before any question is asked, the bid folder is connected and mapped, with 66 files across 25 folders containing active solicitations, client folders, historical proposals, templates, and schedule PDFs. The semantic processing stage resolves entities such as solicitation IDs, clients, trade scopes, and bid values, linking references to the same entities across documents. It then builds registries and relationships, connecting vendor responses to solicitations, historical bids to matching trade scopes, and schedule PDFs to the plans they benchmark.



The pre-built relationship graph an RFP evaluation agent navigates

When a new RFP arrives: The agent starts from a known anchor rather than an empty search. From the bid request, it can traverse to the client's prior engagements, historical bids for the same trade scope, and the pricing benchmarks established by those bids. A $2.45M concrete foundation bid for one hospital client and a $1.9M precast bid for another are not simply similar documents. They're comparable pricing outcomes for known scopes, retrieved through recorded relationships rather than matching words.

What the agent returns: The agent produces a scored evaluation with comparable bids, the compliance criteria mapped to each requirement, the pricing benchmark used for comparison, and a source for every claim. All retrieved content remains within the requesting user's existing permissions.

Extended across systems: With Procore connected, the evaluation can incorporate live project state, including open RFIs against drawings, submittal status, and incident history from comparable work. The agent can reason across file content and project data as one connected context.

Measured Impact

In an internal benchmark of 266 construction documents, the same AI achieved **80%** answer accuracy using Egnyte's pre-built context, compared with **63%** using raw files. It also made **17%** fewer API calls per query.

The improvement comes from building context before the query. The model can use resolved entities, relationships, and current versions instead of reconstructing them from retrieved content. Work done once during context building does not need to be repeated for every query, making retrieval both more accurate and efficient.

How the Study Was Designed

We evaluated multiple knowledge-worker scenarios using the same construction corpus, with each question asked in two conditions: with Egnyte's pre-built context layer and without it—using the underlying raw files. The questions, source corpus, model, and evaluation criteria were kept consistent across both conditions, allowing the impact of pre-built context to be measured directly.

The scenarios represented practical questions a construction professional might ask, including:

- Which previous bids are most comparable to this concrete foundation project, and how does the current price compare?
- Which RFP requirements aren't addressed in the submitted proposal?
- What changed between the current specification and the previous version?
- Which vendors have previously worked on similar projects for this client?
- What's the latest approved schedule, and which documents support it?

How Answers Were Scored

Each response was evaluated against predefined success criteria covering the information required to answer the question accurately and completely. Responses were classified as Complete, Partial, Failed, or Not Possible. Only complete responses counted toward the reported accuracy. A response that

returned an answer but required additional stitching, inference, or cleanup was classified as partial and therefore counted as a non-completion.

Efficiency Was Measured With Accuracy

For each query, we also captured API calls and token consumption. This measures not only whether the AI reached the right answer, but how efficiently it reached it. Token savings were calculated only for scenarios completed under both conditions, avoiding an artificial efficiency advantage from a query that failed early without retrieving the necessary information. Together, the results show the impact of pre-built context on both dimensions that matter for enterprise AI.

Conclusion

The difference between an AI that retrieves and an AI that acts isn't model capability alone. It's whether the structure the model needs, including identity, relationships, currency, and permissions, exists before the question is asked.

Egnyte's Context Layer builds this structure from content organizations already own, keeps it current as content and permissions change, makes coverage visible to administrators, and exposes context through open APIs and MCP. Applications and agents can consume living enterprise context instead of reconstructing it one interaction at a time.

Contact Us

+1-650-968-4018
1350 W. Middlefield Rd.
Mountain View, CA 94043
www.egnyte.com