CLAUDE COWORK

Enterprise Search Plugin Handbook

The Definitive Guide to Unified Cross-Platform Search in Claude Cowork

VERSION	1.0.0
PLUGIN	enterprise-search

RationalEyes.ai
contact@rationaleyes.ai

February 2026

Table of Contents

The Definitive Guide to Unified Cross-Platform Search in Claude Cowork

Version 1.0.0

Part I: What Is Enterprise Search?

Introduction

The enterprise-search plugin eliminates the daily frustration of hunting for information across disconnected tools. Instead of opening Slack, then Gmail, then Google Drive, then Notion, then Jira — repeatedly searching each platform individually and trying to piece together fragments of information — you ask one question and get one unified answer with complete source attribution.

This is not a web search engine for the internet. This is a **knowledge retrieval system for your company's internal information**, designed specifically for knowledge workers who spend hours every week re-discovering decisions, hunting for documents, tracking down experts, and reconstructing context from scattered conversations.

The Knowledge Fragmentation Problem

Modern knowledge work happens everywhere and nowhere simultaneously. A product decision starts in a Slack thread, continues via email, gets documented in Notion, spawns Jira tickets, and eventually lives in a Google Doc. Six weeks later, when someone asks "Why did we decide to use PostgreSQL instead of MongoDB?", the answer exists — but it exists in pieces, distributed across five systems, requiring someone to remember where each fragment lives.

The cost of knowledge fragmentation:

SYMPTOM	BUSINESS IMPACT
Re-asking the same questions	Duplicate work, wasted expert time
Repeating past mistakes	Failed initiatives get retried because lessons weren't captured
Context loss in handoffs	New team members lack institutional knowledge
Slow decision-making	Hours spent researching background before making calls
Information hoarding	Knowledge stays in individuals' heads because documentation is scattered

Enterprise search treats all your connected tools as a single, unified knowledge base. The boundaries between Slack and email, between Drive and Notion, disappear from the user's perspective. One query, executed in parallel across every source, synthesized into a coherent answer.

How Enterprise Search Works

The Five-Stage Process

```
[User Question]
      ↓
[1. Query Decomposition]
      ↓
[2. Parallel Source-Specific Searches]
      ↓
[3. Result Ranking and Deduplication]
      ↓
[4. Knowledge Synthesis]
      ↓
[5. Attributed Answer Delivery]
```

Stage 1: Query Decomposition

Claude analyzes your natural language question to understand intent, extract entities, identify time constraints, and determine which query type this represents. Is this a decision query ("What did we decide?"), a status query ("What's the current state?"), a document query ("Where's the spec?"), or a people query ("Who knows about X?")?

From this analysis, Claude generates source-specific sub-queries. The same question is automatically translated into the native query syntax of each connected tool:

- Slack: Semantic search for "API redesign decision" + keyword search in `#engineering` after January 1
- Email: Search for "API redesign" in subject lines and body text from Sarah
- Google Drive: File search for "API" + "design" in document names and full text
- Notion: Semantic search across database entries for "API migration decision"
- Jira: Task search for "API" in project "Platform Redesign"

Stage 2: Parallel Execution

All sub-queries execute simultaneously. The total search time equals the slowest single source, not the sum of all sources. If Slack takes 2 seconds, email takes 3 seconds, and Drive takes 4 seconds, the entire search completes in 4 seconds, not 9.

If one source fails or hits a rate limit, the others continue. A failed email search does not block Slack and Drive results.

Stage 3: Ranking and Deduplication

Raw results undergo scoring based on relevance, freshness, authority, and completeness. The same information appearing in multiple places gets merged: a decision discussed in Slack, confirmed via email, and documented in Notion appears as one synthesized item, not three separate results.

Stage 4: Knowledge Synthesis

Claude combines results into narrative answers. Instead of presenting a list of search hits from each source, enterprise search produces coherent explanations that integrate information from wherever it was found.

Stage 5: Attribution

Every claim in the synthesized answer links back to its source. You get the answer immediately, but you can always drill down to the original Slack thread, the email confirmation, or the updated document.

Who Should Use This Plugin

Primary Audiences

Knowledge Managers and Operations Teams

You maintain the company's collective intelligence. Enterprise search amplifies your ability to connect people with the information they need, identify knowledge gaps, and surface buried decisions. Use this for:

- Onboarding new employees (search for "how we handle X" across all documentation and conversations)
- Creating knowledge base entries by synthesizing scattered information
- Identifying subject matter experts by tracking who contributes to which topics
- Auditing decision quality by reviewing how decisions were made

IT Administrators and System Integrators

You connect the tools and manage the infrastructure. Enterprise search consolidates search interfaces across your entire SaaS stack. Use this for:

- Reducing tool sprawl (one search interface instead of six)

- Centralizing authentication and access control visibility
- Monitoring usage patterns across information sources
- Providing unified search capability without building custom integrations

Project Managers and Team Leads

You need to know status, track decisions, and maintain context across workstreams. Enterprise search eliminates the daily ritual of checking every tool individually. Use this for:

- Daily digest generation (what happened across all projects today)
- Decision archaeology (when and why did we decide X)
- Status aggregation (compile status from chat, task trackers, and documents)
- Retrospective preparation (gather all artifacts related to a project phase)

Executive Leadership

You need synthesized understanding without drowning in detail. Enterprise search provides high-level summaries backed by verifiable sources. Use this for:

- Strategic decision review (what did we decide about X initiative)
- Cross-functional visibility (what are engineering, product, and marketing saying about the same topic)
- Knowledge audit (do we have documented positions on key questions)
- Institutional memory preservation (capture lessons before key employees leave)

Secondary Audiences

Sales and Customer Success Teams benefit from instant access to product documentation, pricing decisions, customer conversation history, and internal discussions about features and roadmap.

Legal and Compliance Teams use enterprise search to verify what was communicated when, track policy documentation across systems, and perform e-discovery across all connected platforms.

Research and Analysis Teams synthesize information from disparate sources to inform strategic recommendations, competitive analysis, and market research.

Key Concepts and Terminology

Source Categories

The plugin organizes external tools into conceptual categories using the `~category` placeholder system:

CATEGORY	WHAT IT INCLUDES	EXAMPLE TOOLS
`~chat`	Real-time messaging platforms	Slack, Microsoft Teams
`~email`	Email systems	Microsoft 365, Gmail
`~cloud storage`	Document repositories	Google Drive, SharePoint, Dropbox
`~knowledge base`	Wikis and structured knowledge	Notion, Confluence, Guru
`~project tracker`	Task and project management	Jira, Asana, Linear
`~CRM`	Customer relationship management	Salesforce, HubSpot
`~office suite`	Productivity applications	Microsoft 365, Google Workspace

This categorization is intentionally tool-agnostic. Whether your company uses Slack or Teams, Notion or Confluence, the plugin's instructions remain the same.

Query Types

Enterprise search classifies queries to optimize search strategy:

QUERY TYPE	EXAMPLE	SEARCH STRATEGY
Decision	"What did we decide about the API migration?"	Prioritize conversations with conclusion signals, email confirmations, meeting notes
Status	"What's the status of Project Aurora?"	Prioritize task trackers and recent activity, weight freshness heavily
Document	"Where's the onboarding guide?"	Prioritize cloud storage and wikis, search file names and full text
Person	"Who knows about Kubernetes setup?"	Search message authors, task assignees, document collaborators
Factual	"What's our return policy?"	Prioritize official documentation, policy docs, knowledge base articles

QUERY TYPE	EXAMPLE	SEARCH STRATEGY
Temporal	"When did we launch feature X?"	Broad date range search, look for timestamps and announcements
Exploratory	"What do we know about competitor Y?"	Broad search across all sources, synthesize diverse information

Understanding query types helps you formulate better searches and interpret why certain results rank higher than others.

Result Confidence Levels

Not all search results are equally trustworthy. The synthesis engine assesses confidence based on source authority, information freshness, and cross-source agreement:

High Confidence: Multiple recent authoritative sources agree. Example: A decision documented in meeting notes, confirmed via email, and reflected in an updated strategy doc — all from this month.

Moderate Confidence: Single recent source or multiple older sources. Example: A Slack discussion from last month about a technical approach, but no formal documentation.

Low Confidence: Old information, informal source only, or conflicting signals. Example: A casual chat message from six months ago mentioning a potential direction, but no follow-up.

The plugin surfaces confidence levels when appropriate, flagging outdated or uncertain information rather than presenting it as authoritative.

Source Attribution

Every claim in a synthesized answer must be traceable to its source. Attribution includes:

- **Source type**: Which category (chat, email, cloud storage, etc.)
- **Specific location**: Channel name, folder, thread title
- **Timestamp**: When the information was created or last modified
- **Author**: Who wrote, sent, or last updated
- **Document/thread title**: The name of the artifact

Example attribution:

```
Sources:
- ~~chat: #engineering discussion (Jan 14, 2025) — initial decision thread
- ~~email: "API Decision" from Sarah Chen (Jan 15) — formal confirmation
- ~~cloud storage: "API Design Doc v3" last modified Jan 15 — updated specification
```

Part II: Installation and Configuration

Installing the Plugin

Cowork (recommended): Open **Plugin Settings** in the Cowork desktop app, find **Enterprise Search**, and click **Install**. The plugin activates immediately with its default configuration, pre-configured for six common enterprise tools.

> **Note:** *All standard Cowork plugins, including Enterprise Search, are available from Plugin Settings with a single click.*

Installing a customized version: If your IT team has distributed a customized `.plugin` file (for example, pre-configured with your organization's tools), open the `.plugin` file in Cowork. It will show a rich preview of the plugin contents where you can browse the files and accept with a single click.

Claude Code CLI (alternative): If you are using Claude Code in the terminal rather than Cowork, install via:

```
claude plugins add knowledge-work-plugins/enterprise-search
```

Verifying Installation:

After installation, type `/enterprise-search:` in any chat. You should see two command options appear:

- `/enterprise-search:search`
- `/enterprise-search:digest`

If the commands appear, installation succeeded.

Connecting Source Systems

The plugin searches only the tools you explicitly connect via MCP (Model Context Protocol) servers. Out of the box, it supports six common enterprise platforms:

SOURCE	MCP SERVER URL	AUTHENTICATION
Slack	`https://mcp.slack.com/mcp`	OAuth (Sign in with Slack)
Notion	`https://mcp.notion.com/mcp`	OAuth (Sign in with Notion)
Guru	`https://mcp.api.getguru.com/mcp`	OAuth (Sign in with Guru)
Atlassian	`https://mcp.atlassian.com/v1/mcp`	OAuth (Sign in with Atlassian)
Asana	`https://mcp.asana.com/v2/mcp`	OAuth (Sign in with Asana)
Microsoft 365	`https://microsoft365.mcp.claude.com/mcp`	OAuth (Sign in with Microsoft)

Connection Workflow:

1. **Enable the MCP server** in Claude Cowork settings:
   - Navigate to Settings → MCP Servers
   - The six enterprise-search servers should appear in the list
   - Toggle "Enabled" for each source you want to connect
2. **Authenticate** when prompted:
   - The first time you use a command that requires a source, Claude prompts for authentication
   - Click the authentication link
   - Sign in using your company account for that service
   - Grant permissions when requested
   - Claude stores the authentication token securely
3. **Verify connectivity:**
   - Run a simple search: `/enterprise-search:search test`
   - Check which sources appear in the results summary
   - If a source is connected and working, it appears in the "Sources scanned" line

Minimum Recommended Configuration:

For a functional enterprise search experience, connect at least three sources from different categories:

- One messaging platform (Slack or Microsoft Teams via Microsoft 365)
- One document repository (Google Drive, SharePoint via Microsoft 365, Dropbox, or Notion)

- One additional source matching your company's primary tool usage (project tracker, CRM, or knowledge base)

Optimal Configuration:

Connect five or more sources spanning:

- Chat (Slack, Teams)
- Email (Microsoft 365, Gmail)
- Cloud storage (Drive, SharePoint, Dropbox)
- Knowledge base (Notion, Confluence, Guru)
- Project tracker (Jira, Asana, Linear)

The more sources you connect, the more complete your search results.

Authentication and Permissions

What Permissions Are Required

Each MCP server requests specific OAuth scopes during authentication. These permissions determine what Claude can access on your behalf.

Slack Example Scopes:

- `search:read` — Search messages and files
- `channels:read` — List channels you are a member of
- `users:read` — Identify message authors
- `files:read` — Access shared files

Microsoft 365 Example Scopes:

- `Mail.Read` — Read your email
- `Files.Read.All` — Read files in drives you have access to
- `Sites.Read.All` — Search SharePoint sites

General Principle: The plugin only accesses what **you** can already access. If you do not have permission to view a private Slack channel, enterprise search will not return results from that channel. MCP servers enforce the authenticated user's existing permissions.

Security Model

Authentication tokens are stored locally in Claude Cowork's secure credential store. They are never transmitted to Anthropic servers or shared across users. Each user authenticates individually with their own credentials.

Access Control Considerations:

- Enterprise search respects the access controls of connected systems. A user searching for sensitive information only sees results they already have permission to view.
- Shared results (e.g., pasting search output into a document) do not automatically grant access to underlying sources. Links to Slack threads, Drive files, or Jira tickets still require proper permissions.
- For compliance-sensitive industries, review each MCP server's privacy policy and data handling practices before enabling.

Revoking Access

To disconnect a source:

1. Navigate to Settings → MCP Servers
2. Find the server you want to disconnect
3. Toggle "Enabled" to off, or click "Remove"
4. Optionally revoke the OAuth token directly in the source application:
   - **Slack**: Workspace Settings → Installed Apps → Remove Enterprise Search
   - **Google**: Google Account → Security → Third-party apps → Remove Claude Cowork
   - **Microsoft 365**: Account Settings → Apps & services → Revoke

Verifying Your Setup

Test Search:

Run a broad test query that should match content across multiple sources:

```
/enterprise-search:search test
```

or

```
/enterprise-search:search [your company name]
```

Expected Output:

```
[Synthesized results from multiple sources]

Sources scanned: ~~chat, ~~email, ~~cloud storage, ~~knowledge base

Found results across:
- ~~chat: X relevant messages
- ~~email: X threads
- ~~cloud storage: X documents
- ~~knowledge base: X pages
```

If a connected source does not appear in "Sources scanned," check:

- Is the MCP server enabled in Settings?
- Did you complete OAuth authentication?
- Does your account have access to any content in that system?

Test Digest:

Run a daily digest to verify the plugin can scan recent activity:

```
/enterprise-search:digest --daily
```

Expected Output:

A structured summary of recent activity across all sources, grouped by topic, with action items and mentions highlighted.

If no activity appears despite recent work, verify that:

- You are signed in with the correct account for each source
- The time range is appropriate (try `--weekly` for a broader scan)
- Your account has participated in or has access to recent activity

Part III: User Guide

The Search Command

Syntax:

```
/enterprise-search:search <query>
```

The `<query>` is a natural language question or search phrase. You do not need to specify which sources to search — the plugin searches all connected sources automatically.

Basic Examples:

```
/enterprise-search:search API migration timeline
/enterprise-search:search what did we decide about hiring freeze
/enterprise-search:search where is the brand guidelines doc
/enterprise-search:search who works on authentication
```

How It Works:

1. Claude analyzes your query to determine intent and extract key entities
2. Parallel searches launch across all connected sources
3. Results are ranked by relevance, freshness, and authority
4. Duplicate information from multiple sources is merged
5. A synthesized answer is presented with source attribution

Output Format:

For most queries, you receive a direct answer followed by source attribution:

```
The team decided to proceed with the API migration in phases, starting with
read-only endpoints in Q1 and write endpoints in Q2. The decision was made
during the February 5 engineering all-hands, confirmed in Sarah's email on
February 6, and the timeline is documented in the Q1 Planning doc.

Sources:
- ~~chat: #engineering all-hands discussion (Feb 5, 2025)
- ~~email: "API Migration Timeline Confirmed" from Sarah Chen (Feb 6)
- ~~cloud storage: "Q1 2025 Planning" last modified Feb 6
```

For exploratory queries with many results, you receive a summary with key sources highlighted:

```
Found 47 references to the API migration across all sources.

Key themes:
1. Timeline and phasing (discussed in #engineering, documented in Q1 Planning)
2. Resource allocation (email thread with Sarah and David, Jira epic)
3. Risk mitigation (architecture review doc, security assessment)

Top sources:
- ~~cloud storage: "API Migration Plan" (authoritative)
- ~~chat: #engineering thread from Feb 5 (decision rationale)
- ~~project tracker: "API Migration" epic with 23 subtasks

Would you like me to dig deeper into any specific aspect?
```

The Digest Command

Syntax:

```
/enterprise-search:digest [--daily | --weekly | --since <date>]
```

Flags:

FLAG	MEANING	EXAMPLE
`--daily`	Last 24 hours (default if no flag specified)	`/digest --daily`
`--weekly`	Last 7 days	`/digest --weekly`
`--since <date>`	Custom time range	`/digest --since Monday` or `/digest --since 2025-01-20`

What It Does:

Scans all connected sources for recent activity relevant to you:

- Messages that mention you or appear in your channels
- Emails sent to you or threads you participated in
- Documents you own, collaborate on, or were shared with you
- Tasks assigned to you or projects you follow
- Updates to items you are watching

The digest groups this activity by topic or project rather than by source, making it easy to scan what happened across your entire work ecosystem.

Output Structure:

```
# Daily Digest — February 14, 2025

Sources scanned: ~~chat, ~~email, ~~cloud storage, ~~project tracker

## Action Items (3 items)
- [ ] Review Q1 budget projections by Friday — from Finance team, ~~email (Feb 13)
- [ ] Approve PR #847 for API changes — from Alex, ~~chat #engineering (Feb 14)
- [ ] Update onboarding doc with new process — from Sarah, ~~project tracker (Feb 13)

## Decisions Made
- Security review process will now require two approvers instead of one — ~~chat #secu
- Moved launch date to March 15 to allow for additional testing — ~~email from Product

## Project Aurora
- ~~chat: Design review concluded, team chose Option B (#design, Feb 12)
- ~~email: Sarah sent updated spec incorporating feedback (Feb 13)
- ~~cloud storage: "Aurora API Spec v3" updated by Sarah (Feb 13)
- ~~project tracker: 3 tasks moved to In Progress, 2 completed

## Mentions
- You were mentioned in #marketing regarding the new campaign launch timeline
- David referenced your API design proposal in a thread with Product team

## Documents Updated
- "Q1 Budget Projections" — Finance added final numbers (Feb 13)
- "Engineering Roadmap 2025" — You were added as a collaborator (Feb 14)

---
3 action items · 2 decisions · 2 mentions · 2 doc updates
Across 4 sources · Covering last 24 hours
```

When to Use Digest:

- **Every morning**: Catch up on what happened yesterday across all your tools in one read
- **After time off**: Weekly digest after vacation or a long weekend
- **Before key meetings**: Quick scan of related activity to walk in informed
- **End of week**: Review what happened across projects to prepare status reports

Query Syntax and Filters

While natural language queries work for most searches, you can use specific filters to narrow results.

Supported Filters:

FILTER	SYNTAX	EXAMPLE	EFFECT
From	`from:name`	`from:sarah budget`	Only results authored/sent by Sarah
In	`in:location`	`in:engineering API`	Only results in #engineering channel or Engineering folder
After	`after:date`	`after:2025-01-01 migration`	Only results after January 1, 2025
Before	`before:date`	`before:2025-02-01 launch`	Only results before February 1, 2025
Type	`type:content-type`	`type:thread decisions`	Only results of specified type (thread, email, doc, file, task)

Date Formats:

- Absolute: `2025-01-15`, `2025-02-01`
- Relative: `yesterday`, `last week`, `this month`
- Named: `Monday`, `January`

Filter Combination:

Filters apply together (logical AND):

```
/enterprise-search:search from:sarah in:engineering after:2025-01-01 API design
```

Returns results that:

- Are from Sarah
- AND appear in the engineering channel/folder/workspace
- AND are dated after January 1, 2025
- AND mention "API design"

How Filters Are Applied:

Each connected source has different native query syntax. The plugin automatically translates your filters into the appropriate format for each source:

- **Slack:** `from:sarah` becomes `from:@USERID`, `in:engineering` becomes `in:engineering`, dates map to `after:` and `before:`
- **Email:** `from:sarah` becomes sender filter, dates map to sent/received date range
- **Google Drive:** `in:engineering` searches within the "Engineering" folder, dates map to modified date
- **Jira:** `from:sarah` maps to assignee or reporter, dates map to created/updated filters

You write filters once in a standard syntax; the plugin handles source-specific translation.

Common Search Patterns

Finding Decisions

Query Pattern:

```
/enterprise-search:search what did we decide about <topic>
/enterprise-search:search <topic> decision
```

Why It Works: Decision queries prioritize conversations (chat, email) over static documents, weight conclusion signals ("we decided," "let's go with"), and rank recent discussions higher.

Example:

```
/enterprise-search:search what did we decide about remote work policy
```

Typical Result:

```
The company decided to adopt a hybrid model with 3 days in-office and 2 days
remote starting March 1. The decision was made during the February 10 leadership
meeting, announced via email on February 11, and documented in the Employee Handbook.

Sources:
- ~~chat: #leadership meeting notes (Feb 10)
- ~~email: "Remote Work Policy Update" from CEO (Feb 11)
- ~~knowledge base: Employee Handbook updated (Feb 11)
```

Checking Status

Query Pattern:

```
/enterprise-search:search status of <project>
/enterprise-search:search what's happening with <initiative>
```

Why It Works: Status queries prioritize task trackers and recent activity, weight freshness heavily (week-old status is often stale), and synthesize updates from multiple sources.

Example:

```
/enterprise-search:search status of mobile app redesign
```

Typical Result:

```
The mobile app redesign is currently in design review phase. The design team
completed initial mockups last week (Figma file updated Feb 8). Engineering
started technical scoping — 5 of 8 technical tasks are in progress according
to the Jira board. Target launch is still April 15 per the project plan doc.

Sources:
- ~~design: Figma file "Mobile App Redesign v2" (updated Feb 8)
- ~~project tracker: "Mobile Redesign" epic (5/8 tasks in progress)
- ~~cloud storage: "Mobile Redesign Project Plan" (last modified Feb 1)
- ~~chat: #mobile-team status discussion (Feb 12)
```

Locating Documents

Query Pattern:

```
/enterprise-search:search <document name>
/enterprise-search:search where is the <document description>
```

Why It Works: Document queries prioritize cloud storage and knowledge bases, search both file names and full text content, and rank by how recently the document was accessed or modified.

Example:

```
/enterprise-search:search onboarding guide
```

Typical Result:

```
Found "New Hire Onboarding Guide" in Google Drive (Engineering folder, last
updated Jan 25, 2025). The guide covers first-week setup, account provisioning,
and team introductions.

Also found:
- ~~knowledge base: "Onboarding Checklist" in Notion (Sales team version)
- ~~chat: #people-ops has an onboarding thread with additional resources

Main document: [Link to "New Hire Onboarding Guide"]
```

Finding Experts

Query Pattern:

```
/enterprise-search:search who knows about <topic>
/enterprise-search:search who works on <system/project>
```

Why It Works: People queries search message authors, task assignees, and document collaborators. Frequency and recency of contributions score higher.

Example:

```
/enterprise-search:search who knows about Kubernetes infrastructure
```

Typical Result:

```
Based on message history and document authorship, Alex and Priya are your
go-to people for Kubernetes.

Evidence:
- Alex authored the "K8s Cluster Setup Guide" and has 47 messages in #infrastructure
  about Kubernetes topics over the past 3 months
- Priya is assigned to 6 Kubernetes-related tasks in Jira and participated in
  the cluster migration discussion in January

Key resources:
- ~~knowledge base: "K8s Cluster Setup Guide" by Alex
- ~~chat: #infrastructure (Alex and Priya are frequent contributors)
```

Researching Topics

Query Pattern:

```
/enterprise-search:search everything about <topic>
/enterprise-search:search what do we know about <topic>
```

Why It Works: Exploratory queries search broadly across all sources, prioritize synthesis over individual results, and group findings by theme.

Example:

```
/enterprise-search:search everything about competitor analysis
```

Typical Result:

```
Found 62 references to competitor analysis across all sources.

Key themes:
1. **Competitive pricing** — analyzed in "Market Research Q4 2024" doc,
   discussed in #product and #sales channels
2. **Feature comparison** — tracked in "Competitor Feature Matrix" spreadsheet,
   updated monthly by Product team
3. **Win/loss analysis** — CRM reports show reasons for competitive wins/losses

Top resources:
- ~~cloud storage: "Market Research Q4 2024" (comprehensive)
- ~~cloud storage: "Competitor Feature Matrix" (updated monthly)
- ~~CRM: Win/loss reports (real customer feedback)
- ~~chat: #competitive-intel channel (ongoing discussions)

Would you like me to focus on a specific competitor or aspect?
```

Interpreting Results

Understanding Source Attribution

Every search result includes source attribution showing where information came from. This serves two purposes:

1. **Verification**: You can click through to the original source to see the full context
2. **Trust calibration**: Authoritative sources (official docs, formal emails) carry more weight than informal ones (casual chat messages)

Source Types and Authority Levels:

SOURCE TYPE	AUTHORITY LEVEL	WHEN IT MATTERS MOST
~~knowledge base (wiki)	Highest	Factual queries, policy questions
~~cloud storage (final docs)	High	Specifications, strategy, planning
~~email (announcements)	High	Formal decisions, official communications
Meeting notes	Moderate-High	Decision rationale, discussion context
~~chat (thread conclusions)	Moderate	Real-time decisions, informal agreements
~~chat (mid-thread messages)	Lower	Ongoing discussions, preliminary thoughts
~~project tracker (task comments)	Contextual	Implementation details, task-specific info

Reading Confidence Signals:

High-confidence answers are presented as direct statements:

```
The team decided to use REST for the API redesign.
```

Moderate-confidence answers include qualifying language:

```
Based on the discussion in #engineering last month, the team was leaning
toward REST. This may have evolved since then.
```

Low-confidence answers explicitly flag uncertainty:

```
I found a reference to an API discussion from three months ago, but no formal
decision document. The information may be outdated — you might want to check
with the team for current status.
```

When Results Are Incomplete

If enterprise search cannot find what you are looking for, or results seem partial, check:

1. Is the relevant source connected?

If your answer lives in Confluence but you only have Slack and email connected, the search cannot find it. The "Sources scanned" line tells you which sources were checked.

2. Do you have permission to access the information?

Enterprise search only returns what you can already access. If the answer lives in a private channel or restricted folder you do not have permission to view, it will not appear.

3. Is the query too specific or too broad?

Very narrow queries (exact phrases, multiple filters) may miss relevant results phrased differently. Very broad queries may return so many results that the most relevant ones get buried.

Try Query Refinement:

- **Too few results**: Broaden terms, remove date filters, try synonyms
- **Too many results**: Add filters, specify time range, include more specific keywords
- **Irrelevant results**: Add negations (`migration NOT email` to exclude email migration discussions), specify source type

Example Refinement Sequence:

```
Query 1: /enterprise-search:search postgres migration january
Result: Too specific, found only 2 results

Query 2: /enterprise-search:search postgres migration
Result: Better, found 15 results but includes very old discussions

Query 3: /enterprise-search:search postgres migration after:2024-11-01
Result: Optimal, found 8 relevant results from recent discussions
```

Handling Conflicting Information

When sources disagree, enterprise search surfaces the conflict explicitly:

```
I found conflicting information about the launch date:
- The project plan doc (updated Jan 10) lists March 1
- The #product channel discussion (Feb 5) mentions pushing to March 15
- The Jira milestone is still set to March 1

The chat discussion is more recent but the formal documents have not been
updated yet. You may want to confirm which date is current.
```

Do not assume the most recent source is always correct. Sometimes:

- A casual chat message proposes a change that was not ultimately approved
- An email announcement is sent but documents have not been updated yet
- Different teams have different information and alignment is needed

When you see conflicts, investigate rather than picking one source arbitrarily.

Part IV: Administrator Guide

Deployment Planning

As an administrator deploying enterprise search to your organization, consider these factors before rollout.

Scope and Phasing

Pilot Phase (2-4 weeks):

- Deploy to a small group (10-20 users) representing diverse roles
- Connect 3-4 core sources (chat, email, primary document repository)
- Gather feedback on search quality, relevance, and missing sources

Expansion Phase (1-2 months):

- Roll out to departments sequentially (start with knowledge-intensive teams)
- Add additional sources based on pilot feedback
- Monitor usage patterns and adjust configurations

Organization-Wide Rollout:

- Deploy to all users after successful expansion
- Establish support channels and documentation
- Schedule training sessions for new user cohorts

Recommended Pilot User Mix:

ROLE TYPE	WHY INCLUDE
Knowledge managers	Will identify gaps in coverage and source priority issues
Executives	Will test synthesized high-level summaries and cross-functional visibility
Project managers	Will stress-test multi-source searches and digest functionality
Individual contributors	Will reveal everyday use cases and common query patterns

Source Selection Strategy

Not all organizations need all sources. Prioritize based on where your company's knowledge lives:

Essential Sources (connect first):

- Primary chat platform (Slack, Teams)
- Email system (Microsoft 365, Gmail)
- Main document repository (Google Drive, SharePoint, Dropbox)

High-Value Secondary Sources (connect if widely used):

- Knowledge base (Notion, Confluence, Guru) — if you have a wiki culture
- Project tracker (Jira, Asana, Linear) — if project status queries are common
- CRM (Salesforce, HubSpot) — for customer-facing teams

Specialized Sources (connect for specific teams):

- Design tools (Figma, Canva) — for design and marketing teams
- Product analytics (Amplitude, Mixpanel) — for product and growth teams
- Code repositories (GitHub, GitLab) — for engineering teams via custom MCP servers

Decision Framework:

For each potential source, ask:

1. Do 50%+ of users access this tool weekly?
2. Does it contain knowledge not duplicated elsewhere?
3. Is search within this tool currently a pain point?
4. Is there an available MCP server for it?

If yes to all four, connect it. Otherwise, defer.

Source Priority and Configuration

Different query types benefit from searching certain sources first. While enterprise search searches all sources in parallel, result ranking is influenced by source priority weights.

Configuring Priority by Query Type

The plugin automatically adjusts source priority based on detected query type, but administrators can tune these weights for organization-specific needs.

Default Priority Weights (1.0 = baseline):

QUERY TYPE	~~CHAT	~~EMAIL	~~CLOUD STORAGE	~~KNOWLEDGE BASE	~~PROJECT TRACKER	~~CRM
Decision	1.3	1.2	1.0	0.9	0.8	0.7
Status	0.9	0.8	1.0	0.9	1.4	0.9
Document	0.7	0.8	1.5	1.4	0.7	0.6
Person	1.2	0.9	1.0	0.8	1.1	1.0
Factual	0.8	0.9	1.2	1.5	0.7	0.7

Tuning Recommendations:

For **documentation-heavy cultures** (consulting, legal, research):

- Increase `~knowledge base` and `~cloud storage` weights across all query types
- Decrease `~chat` weight for factual queries (encourage documented knowledge)

For **fast-moving startups** (tech, agencies):

- Increase `~chat` weight for decision and status queries (decisions happen in real-time)
- Increase `~project tracker` weight (Jira/Asana is source of truth for status)

For **customer-centric organizations** (sales, customer success):

- Increase `~CRM` weight across all query types
- Add customer name entity recognition to route customer-related queries to CRM

Implementing Custom Weights:

Custom priority configuration is advanced. For most deployments, the defaults work well. To customize, you would create a complementary skill that adjusts ranking logic (see Part VI for extension techniques).

Rate Limiting Management

MCP servers impose rate limits to protect their infrastructure. Enterprise search handles limits gracefully, but administrators should understand the constraints.

Common Rate Limits by Source

SOURCE	TYPICAL LIMIT	SCOPE	WHAT HAPPENS WHEN EXCEEDED
Slack	20 requests/minute	Per user	HTTP 429, retry after 60 seconds

SOURCE	TYPICAL LIMIT	SCOPE	WHAT HAPPENS WHEN EXCEEDED
Microsoft 365	Varies by endpoint	Per user	Throttling, exponential backoff
Notion	3 requests/second	Per integration	HTTP 429, retry after delay
Google Drive	1000 requests/100 seconds	Per user	403 or 429, quota reset after window
Asana	150 requests/minute	Per user	429, retry after indicated time

Enterprise Search Rate Limit Handling:

1. **Automatic Retry**: When a source returns a rate limit error, the plugin notes the failure and continues with other sources. It does not block the entire search.

2. **User Notification**: Results include a note:

```
Note: ~~email was temporarily rate limited during this search.
Results above are from ~~chat, ~~cloud storage, and ~~knowledge base only.
```

3. **Exponential Backoff**: If a user triggers rate limits repeatedly, the plugin increases the delay before retrying that source.

Preventing Rate Limit Issues:

- **Spread digest generation**: Avoid having all users run `/digest` at the exact same time (e.g., all at 9 AM). Stagger by team or individual schedule.
- **Monitor high-volume users**: Identify users making many searches per minute and provide guidance on query refinement.
- **Use targeted searches**: Encourage specific queries over very broad exploratory searches to reduce API load.

For High-Volume Deployments:

Organizations with hundreds of concurrent users may need:

- **Service accounts with higher rate limits**: Some MCP servers offer enterprise tiers with increased quotas
- **Caching layer**: Implement a caching proxy for frequently accessed results (requires custom development)
- **Query throttling**: Limit searches per user per minute at the organization policy level

User Training and Adoption

Successful enterprise search adoption requires more than technical setup. Users must understand when and how to use the plugin effectively.

Training Program Structure

Initial Onboarding (30 minutes):

1. **The Problem** (5 min): Demonstrate the pain of searching across six tools individually
2. **The Solution** (5 min): Show the same search executed via `/search`, returning unified results
3. **Core Commands** (10 min): Walk through `/search` and `/digest` with real examples from your company
4. **Query Patterns** (10 min): Teach the five most common search patterns for your organization

Weekly Tips (ongoing):

Send one tip per week via your company communication channel:

- Week 1: "Use filters to narrow results: `from:sarah after:2025-01-01`"
- Week 2: "Run `/digest --daily` every morning to catch up in one read"
- Week 3: "Find experts by searching for contributors: `who knows about X`"

Office Hours (monthly):

Hold monthly drop-in sessions where users can ask questions, share use cases, and request features.

Common User Questions and Answers

Q: Do I need to learn different search syntax for each tool?

A: No. Enterprise search uses one query syntax. The plugin automatically translates your query into the native syntax for Slack, Gmail, Drive, Jira, etc.

Q: Can I search for something that I saw but do not have access to anymore?

A: No. Enterprise search only returns results you currently have permission to view. It respects all access controls from connected sources.

Q: How do I know which sources were searched?

A: Every result includes a "Sources scanned" line listing which sources were checked. If a source you expected is missing, it may not be connected or may have been rate-limited.

Q: What if I find conflicting information?

A: Enterprise search will explicitly surface conflicts when detected. In these cases, check the timestamps and source authority levels, and verify with the source owners if needed.

Q: Can other people see my searches?

A: No. Searches are private. Only you see your query and results. However, if you share or paste search results into a document or chat, that shared content becomes visible to others.

Q: What happens if one source is down?

A: Searches continue with the remaining sources. A note appears indicating which source was unavailable.

Monitoring and Usage Analytics

Track adoption and identify opportunities for improvement through usage monitoring.

Key Metrics to Monitor

METRIC	WHAT IT TELLS YOU	HOW TO TRACK
Active users per week	Adoption rate	Count unique users running `/search` or `/digest`
Queries per user per week	Engagement depth	Average searches per active user
Sources per query	Coverage breadth	How many sources return results on average
Zero-result queries	Search effectiveness	Queries that returned no results from any source
Digest adoption	Proactive usage	% of users running `/digest` at least weekly

Interpreting Metrics:

- **High active users, low queries per user**: Users try it once but do not adopt. Improve training or query relevance.
- **High zero-result queries**: Users searching for information not in connected sources, or queries poorly formed. Review common failed queries.
- **Low digest adoption**: Users do not see value in proactive summaries. Showcase digest use cases.
- **Low sources per query**: Queries are too narrow or sources are poorly connected. Check MCP configurations.

Feedback Collection

Implement a lightweight feedback mechanism:

In-Product Feedback: After presenting search results, include:

```
Was this helpful? [Yes] [No] [Provide feedback]
```

Quarterly Surveys: Ask users:

1. How often do you use enterprise search? (Daily / Weekly / Monthly / Rarely)
2. Which query types do you use most? (Decision / Status / Document / People / Other)
3. Which sources do you wish were connected that are not currently?
4. What would make enterprise search more useful?

Usage Pattern Analysis: Review anonymized query logs to identify:

- Most common query patterns (adapt training to emphasize these)
- Frequently searched topics (consider creating dedicated skills or shortcuts)
- Common filters (e.g., if everyone filters `from:ceo`, create a shortcut)

Part V: Advanced Capabilities

Understanding the Three-Skill Architecture

Enterprise search's intelligence comes from three interconnected skills working in concert:

1. **search-strategy**: Query decomposition, source-specific translation, and result ranking
2. **source-management**: Available source detection, priority ordering, and rate limit handling
3. **knowledge-synthesis**: Result deduplication, confidence scoring, and narrative generation

Understanding how these skills work helps you use enterprise search more effectively and troubleshoot when results do not match expectations.

Multi-Source Query Decomposition

The search-strategy skill transforms a single natural language query into parallel, source-specific searches.

The Decomposition Process

Step 1: Query Type Classification

The first analysis determines what type of question this is:

QUERY	DETECTED TYPE	WHY
"What did we decide about X?"	Decision	Contains decision-seeking language
"What's the status of Y?"	Status	Contains status indicators
"Where's the Z doc?"	Document	Seeking a specific artifact
"Who knows about W?"	Person	Asking for expertise identification
"What's our policy on V?"	Factual	Seeking established information

Query type determines which sources get prioritized and how results are ranked.

Step 2: Entity and Constraint Extraction

From the query, the skill extracts:

- **Keywords**: Core terms that must appear ("API", "migration", "timeline")

- **Entities**: People ("Sarah"), projects ("Project Aurora"), teams ("engineering")
- **Time constraints**: "this week", "last month", "after January 1"
- **Source hints**: "in Slack", "that email", "the Notion doc"
- **Filters**: `from:`, `in:`, `after:`, `before:`, `type:`

Step 3: Source-Specific Query Generation

For each connected source, a targeted query is generated using that source's native capabilities:

Example Decomposition:

User query: `"What did we decide about the API migration timeline after January?"`

SOURCE	GENERATED QUERY	RATIONALE
Slack	Semantic: "API migration timeline decision" Keyword: "API migration" in:#engineering after:2025-01-01	Semantic for conceptual match, keyword for exact match, filtered to relevant channel and date
Email	Search: "API migration timeline" in subject or body, after:2025-01-01	Email subject lines often contain decision language
Google Drive	File search: "API migration" in name or full text, modified after:2025-01-01	Formal decisions often documented in Drive
Notion	Semantic: "API migration timeline decision rationale"	Notion databases good for decision documentation
Jira	Text: "API migration", workspace: Engineering, modified after:2025-01-01	Tasks reflect implementation of decisions

Each query is optimized for its target source's search capabilities and content characteristics.

Step 4: Parallel Execution

All source-specific queries execute simultaneously:

```
┌─────────────────┐
│   User Query    │
└────────┬────────┘
         │
         ↓
    [Decompose]
         ↓
    ┌────┴─────────────────────┐
    ↓         ↓         ↓      ↓
 [Slack]  [Email]  [Drive] [Notion]  ← Parallel execution
    ↓         ↓         ↓      ↓
    └────┬─────────────────────┘
         │
         ↓
  [Merge & Rank]
         ↓
 [Synthesized Answer]
```

Total time = max(Slack time, Email time, Drive time, Notion time), not the sum.

Query Broadening Strategy

When initial queries return too few results, the search-strategy skill automatically broadens the search:

Broadening Sequence:

```
Original query: "PostgreSQL migration Q2 timeline decision"

Iteration 1 (original):
  Keywords: ["PostgreSQL", "migration", "Q2", "timeline", "decision"]
  Results: 2 (too few)

Iteration 2 (remove time constraint):
  Keywords: ["PostgreSQL", "migration", "timeline", "decision"]
  Results: 8 (better)

If still too few, Iteration 3 (remove domain-specific terms):
  Keywords: ["PostgreSQL", "migration"]
  Results: 15

If still too few, Iteration 4 (broaden to category):
  Keywords: ["database", "migration"]
  Results: 47
```

The skill stops broadening when sufficient relevant results are found or when further broadening would produce too much noise.

Knowledge Synthesis and Deduplication

The knowledge-synthesis skill combines raw results into coherent answers.

Cross-Source Deduplication

The same information often appears in multiple places. The synthesis skill identifies and merges duplicates.

Deduplication Signals:

SIGNAL	EXAMPLE
Text similarity	Slack message and email contain nearly identical text
Same author, close timestamps	Sarah posted in Slack Tuesday, sent email Wednesday with same content
Cross-references	Email says "as discussed in Slack #engineering thread"
Entity overlap	Both mention the same decision, people, and dates

Deduplication Example:

Raw results:

```
Slack (#engineering, Jan 15): "Team decided on REST over GraphQL for API v2"
Email (from Sarah, Jan 16): "Following yesterday's discussion, confirming REST for API
Notion (API Decisions doc, Jan 16): "Decision: REST for API v2 (per eng discussion)"
Jira (API-123, Jan 17): "Implement REST endpoints per team decision"
```

After deduplication:

```
The team decided to use REST over GraphQL for API v2. The decision was made
in the January 15 engineering discussion, confirmed by Sarah via email on
January 16, and documented in the API Decisions page. Implementation is
tracked in Jira ticket API-123.

Sources:
- ~~chat: #engineering discussion (Jan 15) — initial decision
- ~~email: "API v2 Decision Confirmation" from Sarah (Jan 16)
- ~~knowledge base: "API Decisions" page updated Jan 16
- ~~project tracker: API-123 "Implement REST endpoints" (created Jan 17)
```

Four separate results merged into one narrative with complete source attribution.

Authority Hierarchy

When the same information appears in sources with different authority levels, the synthesis skill weights them appropriately.

Authority by Source Type:

For **factual/policy queries:**

```
Wiki/knowledge base > Published documents > Email announcements > Chat conclusions > C
```

For **decision queries:**

```
Meeting notes > Thread conclusions > Email confirmations > Chat messages > Task commen
```

For **status queries:**

```
Task tracker > Recent chat > Status documents > Email updates > Old documentation
```

Authority Example:

Query: "What's our remote work policy?"

Results:

- Slack #general (last month): "Thinking about going full remote"
- Email from HR (last week): "New remote work policy: 3 days in-office, 2 remote"
- Employee Handbook (updated last week): "Hybrid policy: 3 days in-office, 2 remote, effective March 1"

Synthesis prioritizes Employee Handbook (official policy document) over email (announcement) over Slack (discussion). The answer leads with the handbook, notes the email announcement, and ignores the speculative Slack message as pre-decision discussion.

Confidence Scoring

The synthesis skill assesses how confident it is in each answer based on freshness, authority, and source agreement.

Confidence Factors

FACTOR	HIGH CONFIDENCE	MODERATE CONFIDENCE	LOW CONFIDENCE
Freshness	Last 7 days	Last 30 days	Older than 30 days
Authority	Official docs, knowledge base	Formal email, meeting notes	Casual chat messages
Source agreement	3+ sources agree	2 sources agree	Single source only
Completeness	Full context provided	Partial information	Brief mention only

Confidence Expression in Answers:

High confidence (direct statement):

```
The launch date is March 15, 2025.
```

Moderate confidence (qualified statement):

```
Based on the project plan from last month, the launch date is March 15,
though this may have been updated since.
```

Low confidence (explicit uncertainty):

```
I found a reference to a March 15 launch date in a Slack message from
two months ago, but I couldn't find recent confirmation. You should
verify this is still current.
```

Handling Temporal Degradation

Information loses confidence as it ages, especially for status and decision queries.

Freshness Half-Life by Query Type:

QUERY TYPE	INFORMATION STAYS FRESH FOR
Status	1 week (status changes frequently)
Decision	1 month (decisions are more stable but may be revisited)
Factual/Policy	6 months (official information changes slowly)

QUERY TYPE	INFORMATION STAYS FRESH FOR
People	3 months (team structures and expertise evolve)

A status update from three weeks ago is flagged as potentially outdated. A policy document from six months ago is still considered authoritative unless contradicted by newer information.

Handling Conflicting Information

When sources disagree, the synthesis skill surfaces the conflict rather than silently choosing one version.

Conflict Detection

Conflicts are detected when:

- Different sources state opposing facts about the same entity
- Timestamps or dates differ for the same event
- Mutually exclusive options are both presented as decisions

Conflict Resolution Strategy:

1. **Check timestamps:** More recent source likely supersedes older one
2. **Check authority:** Official doc trumps casual chat
3. **Check context:** Was the older source a proposal, and the newer one the final decision?

If clear resolution exists, present it:

```
The launch date was originally March 1 (per the January project plan) but
was pushed to March 15 (confirmed in Sarah's February 5 email and updated
in the project tracker).
```

If no clear resolution, surface the conflict:

```
I found conflicting information about the API approach:
- January 10 discussion in #engineering suggested GraphQL
- January 15 email from Sarah confirmed REST
- The design doc (last updated January 12) still references GraphQL

The email is more recent but the doc has not been updated yet. This may
need alignment.
```

Cross-Source Verification

High-stakes queries (decisions, policy, commitments) benefit from cross-source verification:

Verification Confidence Levels:

VERIFICATION	CONFIDENCE
Single informal source	Low — verify before acting
Single authoritative source	Moderate — likely correct but unconfirmed
Multiple sources agree (same type)	Moderate-High — consistent but may be echoing
Multiple sources agree (different types)	High — cross-verified
Authoritative + multiple confirming	Very High — definitively verified

Example:

```
Query: "What did we commit to the client for delivery date?"

High confidence answer requires:
- Email to client stating the date (authoritative commitment)
- PLUS internal confirmation (Slack discussion, project plan, or task tracker)

Single email alone: moderate confidence
Email + Slack + Jira all showing same date: very high confidence
```

Part VI: Customization and Extension

When to Customize

The enterprise-search plugin is designed to work out-of-the-box for most organizations. Customize when:

1. **You use sources not in the default configuration**: Your company uses Linear instead of Jira, or a proprietary internal tool
2. **Your organization has unique search patterns**: Financial services firms searching for compliance documentation, law firms searching case files
3. **You want to add specialized query types**: Code search, customer history search, compliance audit search
4. **You need to integrate proprietary knowledge**: Internal glossaries, org charts, custom taxonomies

Three customization approaches (from the plugin development guide):

APPROACH	EFFORT	BEST FOR
A: Direct modification	Low	Quick personal experiments, small additions
B: Fork the plugin	Medium	Team-wide customizations, significant changes
C: Complementary skills via CLAUDE.md	Low	Project-level additions without touching plugin

Adding Custom MCP Sources

The default enterprise-search configuration includes six MCP servers. You can add more.

Identifying Available MCP Servers

Check the MCP registry at the Anthropic developer portal or ask your IT team which internal MCP servers exist for your company's tools.

Common additional sources:

CATEGORY	TOOLS WITH MCP SERVERS
Code repositories	GitHub, GitLab
Design tools	Figma, Adobe XD
Product analytics	Amplitude, Mixpanel
Customer support	Zendesk, Intercom
Marketing automation	HubSpot, Marketo
HR systems	BambooHR, Workday

Adding an MCP Server

Step 1: Locate the plugin's .mcp.json file

```
enterprise-search/1.0.0/.mcp.json
```

Step 2: Add the new server configuration

Example adding Linear (project tracker):

```
{
  "mcpServers": {
    "slack": { "type": "http", "url": "https://mcp.slack.com/mcp" },
    "notion": { "type": "http", "url": "https://mcp.notion.com/mcp" },
    "guru": { "type": "http", "url": "https://mcp.api.getguru.com/mcp" },
    "atlassian": { "type": "http", "url": "https://mcp.atlassian.com/v1/mcp" },
    "asana": { "type": "http", "url": "https://mcp.asana.com/v2/mcp" },
    "ms365": { "type": "http", "url": "https://microsoft365.mcp.claude.com/mcp" },
    "linear": { "type": "http", "url": "https://mcp.linear.app/mcp" }
  }
}
```

Step 3: Update CONNECTORS.md

Document the new source so users know it is available:

```
| Project tracker | `~~project tracker` | Atlassian, Asana, Linear | Monday, ClickUp |
```

Step 4: Authenticate

After adding the MCP server, the next time you run `/search`, Claude will prompt you to authenticate with Linear.

Creating Custom MCP Servers for Proprietary Tools

If your company has internal tools without public MCP servers, you can create one. This requires development work but follows a standard protocol.

MCP Server Development Resources:

- Anthropic MCP specification: https://modelcontextprotocol.io
- MCP server templates: Available in the Anthropic developer portal
- Example implementations: Open-source MCP servers on GitHub

Basic MCP Server Structure:

```
# Example MCP server for a proprietary knowledge base

from mcp.server import Server
from mcp.types import Tool

server = Server("my-internal-kb")

@server.tool()
async def search_knowledge_base(query: str, limit: int = 10):
    """Search the internal knowledge base"""
    # Connect to your internal API
    results = await internal_kb_api.search(query, limit)
    return {
        "results": [
            {
                "title": r.title,
                "content": r.content,
                "url": r.url,
                "last_modified": r.timestamp
            }
            for r in results
        ]
    }

if __name__ == "__main__":
    server.run()
```

Deployment:

For stdio-type servers (running locally), add to `.mcp.json`:

```
{
  "mcpServers": {
    "internal-kb": {
      "command": "python",
      "args": ["${CLAUDE_PLUGIN_ROOT}/servers/internal_kb_server.py"],
      "env": {
        "KB_API_TOKEN": "${INTERNAL_KB_TOKEN}"
      }
    }
  }
}
```

Industry-Specific Adaptations

Different industries have unique search needs. Here are adaptation patterns.

Legal Industry

Key Needs:

- Case law search
- Matter-specific document retrieval
- Conflict checking
- Precedent research

Customization Example:

Create a complementary skill `legal-search-enhancement` in `.claude/skills/`:

```
---
name: legal-search-enhancement
description: >
  Legal-specific search enhancements for enterprise search. Use when
  searching for case law, matter documents, precedents, conflicts,
  or legal research materials.
---

# Legal Search Enhancement

## Matter-Specific Search

When a user references a matter number or client name, automatically
filter results to that matter context:

- Documents tagged with matter code
- Emails in matter-specific folders
- Time entries and billing records

## Citation Recognition

Recognize legal citation formats and search appropriately:

- Case citations: "123 F.3d 456" searches case databases
- Statute citations: "42 U.S.C. § 1983" searches statutory databases
- Regulation citations: "17 C.F.R. § 240.10b-5" searches regulatory databases

## Precedent Search

When user asks "Have we dealt with X before?", search:
- Closed matters with similar issues
- Internal memos and research
- Outside counsel opinions
- Prior filings and briefs
```

Add routing in `CLAUDE.md`:

```
## Legal Search Rules

When searching for case-related information, always apply the
legal-search-enhancement skill.

When a user provides a matter number (format: YYYY-XXXXX), restrict
all searches to documents, emails, and time entries tagged with
that matter.

Citations in queries (e.g., "find the 42 U.S.C. 1983 research") should
trigger searches in legal databases, not general enterprise sources.
```

Healthcare Industry

Key Needs:

- Patient care protocol search
- Clinical guideline retrieval
- Regulatory compliance documentation
- Research and literature access

Customization Example:

Fork the plugin and add a new skill `clinical-search`:

```
---
name: clinical-search
description: >
  Clinical and healthcare-specific search capabilities. Use when
  searching for patient care protocols, clinical guidelines,
  treatment pathways, medication information, or regulatory
  compliance documentation in healthcare settings.
---

# Clinical Search

## Patient Care Protocol Search

When searching for care protocols:
- Prioritize official clinical guideline sources
- Check for version currency (outdated protocols are flagged)
- Cross-reference with regulatory requirements

## Medication Information Queries

When user asks about medications:
- Search internal formulary first
- Cross-reference with drug interaction databases
- Check for recent safety alerts or recalls

## Regulatory Compliance Search

When searching for compliance documentation:
- HIPAA policies in official policy repository
- Joint Commission standards in accreditation folder
- CMS guidelines in regulatory compliance database
```

Financial Services

Key Needs:

- Regulatory documentation (SEC filings, compliance policies)
- Client portfolio information
- Market research and analysis
- Trade documentation and audit trails

Customization Example:

Add custom MCP server for proprietary trading system and portfolio management tool, then create a skill for financial terminology:

```
---
name: financial-search-terminology
description: >
  Financial services terminology and search context. Use when
  searching for portfolio information, trade documentation,
  regulatory filings, compliance policies, or market research
  in financial services context.
---

# Financial Search Terminology

## Entity Recognition

Recognize and properly handle:
- Ticker symbols (AAPL, MSFT) → search portfolio holdings and trade history
- CUSIP identifiers → search specific security documentation
- Account numbers → restrict to authorized user access only

## Regulatory Document Classification

When searching regulatory content:
- SEC filings: prioritize EDGAR database
- Internal compliance: official compliance repository
- Audit documentation: audit trail database

## Time-Sensitive Information

Financial data degrades quickly. Apply strict freshness requirements:
- Market data: only current day
- Portfolio positions: only current day
- Research reports: flag if older than 30 days
- Regulatory guidance: check for updates quarterly
```

Company-Specific Customization

Every company has unique terminology, processes, team structures, and tools. Encode these to improve search relevance.

Custom Terminology and Entity Recognition

Problem: Your company uses internal product names, project code names, and abbreviations that do not match public terminology.

Solution: Create a terminology mapping skill.

File: `.claude/skills/acme-terminology/SKILL.md`

```
---
name: acme-terminology
description: >
  Acme Corp internal terminology, project code names, and entity
  recognition. Use when search queries reference internal projects,
  products, teams, or company-specific abbreviations.
---

# Acme Corp Terminology

## Product Name Mapping

| Internal Name | Public Name | Search Terms |
|---------------|-------------|--------------|
| Platform | Acme Platform | platform, acme, main product |
| Widget Pro | Professional Edition | widget, pro, professional |
| Enterprise Suite | Acme Enterprise | enterprise, suite, ent |

When user searches for any of these terms, include all aliases.

## Project Code Names

| Code Name | Official Name | Status |
|-----------|---------------|--------|
| Project Phoenix | Database Migration Initiative | Active |
| Project Aurora | API Redesign | Active |
| Project Sunset | Legacy System Retirement | Completed Q4 2024 |

Map code names to official names for search expansion.

## Team and Department Identifiers

| Abbreviation | Full Name | Slack Channels |
|--------------|-----------|----------------|
| PMM | Product Marketing | #pmm, #product-marketing |
| RevOps | Revenue Operations | #revops, #revenue |
| EngInfra | Engineering Infrastructure | #infra, #infrastructure |

Expand abbreviations in searches to improve recall.

## Location Codes

| Code | Office Location |
|------|-----------------|
| SF | San Francisco HQ |
| NY | New York Office |
| LON | London Office |
| SYD | Sydney Office |
When filtering `in:SF`, interpret as San Francisco-related content.
```

Process and Workflow Integration

Problem: Your company has specific approval workflows, review processes, and routing that affect how decisions are documented.

Solution: Create a company-process skill that helps interpret search results in context.

File: `.claude/skills/acme-processes/SKILL.md`

```
---
name: acme-processes
description: >
  Acme Corp internal processes, approval workflows, and decision
  routing. Use when interpreting search results related to decisions,
  approvals, or status to provide context about Acme's workflows.
---

# Acme Corp Processes

## Decision Approval Flow

All major decisions follow this pattern:
1. Proposal in ~~chat or email
2. Discussion in relevant channel or meeting
3. Decision doc created in ~~knowledge base
4. Executive approval via email
5. Announcement in #general

When searching for "what was decided", look for the decision doc
and executive approval email as authoritative sources.

## Launch Checklist

Product launches require:
- Launch plan doc (Product team)
- Engineering readiness (Engineering sign-off)
- Marketing campaign plan (Marketing team)
- Support documentation (Support team)
- Legal review (for new terms or pricing)

When searching "is X ready to launch", check all checklist items.

## Budget Approval Routing

| Amount | Approver | Documentation Required |
├─────────┼───────────┼───────────────────────┤
│ < $5K │ Director │ Email request │
│ $5K-$25K │ VP │ Budget proposal doc + email │
│ $25K-$100K │ C-level │ Formal proposal + finance review │
│ > $100K │ CEO + Board │ Board deck + detailed analysis │

When searching for budget decisions, look for appropriate approval level.
```

Creating Specialized Search Commands

You can create domain-specific search commands that combine enterprise search with specialized logic.

Example: Customer Search Command

File: `commands/customer-search.md` (in a forked plugin)

```
---
description: Search all information related to a specific customer
argument-hint: "<customer name or account ID>"
---

# Customer Search Command

Search across all sources for information related to a specific customer.

## Process

1. **Identify the customer**: Extract customer name or account ID from `$ARGUMENTS`

2. **Search across sources**:
   - ~~CRM: Account history, contacts, opportunities, support cases
   - ~~chat: Mentions in customer-facing channels (#sales, #customer-success)
   - ~~email: Email threads with customer domain
   - ~~project tracker: Tickets or tasks tagged with customer name
   - ~~cloud storage: Documents in customer-specific folders

3. **Synthesize customer profile**:
```

Customer: [Name]

Account created: [Date] Primary contact: [Name, title] Current tier: [Plan level]

Recent Activity

- [Recent interactions, support cases, sales opportunities]

Open Items

- [Active tickets, pending tasks, ongoing projects]

Key Documents

- [Contracts, proposals, case studies]

Internal Discussions

- [Recent mentions in Slack, email threads]

```

4. **Highlight action items**: Surface any open support cases, pending renewals, or ac

## Privacy Note

Only return customer information the user has permission to access based
on their role and CRM access controls.
```

Example: Compliance Audit Search

File: `commands/compliance-search.md`

```
---
description: Search for compliance-related documentation and evidence
argument-hint: "<regulation or audit topic>"
---

# Compliance Audit Search

Search across all sources for compliance documentation and audit evidence.

## Process

1. **Identify the regulation or topic**: Extract from `$ARGUMENTS` (e.g., "GDPR", "SOC

2. **Search compliance-specific sources**:
   - ~~knowledge base: Official compliance policies and procedures
   - ~~cloud storage: Audit documentation, compliance checklists
   - ~~project tracker: Compliance-related tasks and remediation items
   - ~~email: Correspondence with auditors or compliance team

3. **Cross-reference requirements**:
   - Load the relevant compliance framework from references
   - Map found documentation to specific requirements
   - Identify gaps (requirements without supporting documentation)

4. **Present audit-ready summary**:
```

Compliance Audit: [Topic]

Documentation Found

- Policies: [List of policy documents with dates]
- Procedures: [List of procedure documents]
- Evidence: [Training records, logs, certifications]

Coverage Analysis

- Requirements met: [List with supporting docs]
- Requirements with partial documentation: [List with gaps]
- Requirements without documentation: [List]

Action Items

- [Items needing attention before audit]

```

5. **Flag freshness issues**: Highlight any documentation older than the required revi
```

Part VII: Troubleshooting and Best Practices

Common Error Scenarios

"No sources available" Error

Symptom:

```
To search across your tools, you'll need to connect at least one source.
Check your MCP settings to add ~~chat, ~~email, ~~cloud storage, or other tools.
```

Cause: No MCP servers are connected or enabled.

Resolution:

1. Go to Settings → MCP Servers
2. Verify that at least one enterprise-search MCP server is listed
3. Toggle "Enabled" for the servers you want to use
4. Authenticate when prompted
5. Retry the search

"Rate limited" Notice

Symptom:

```
Note: ~~chat is temporarily rate limited. Results above are from
~~email, ~~cloud storage, and ~~knowledge base only.
```

Cause: Too many API calls to the Slack MCP server in a short time.

Resolution:

- Wait 1-2 minutes before searching again
- Retry with more specific query to reduce API calls needed
- If rate limits are frequent, review query patterns (are you searching too broadly?)

Prevention:

- Use filters to narrow searches
- Avoid running very broad searches repeatedly
- Stagger digest generation if many users run at the same time

"Authentication expired" Error

Symptom:

```
Could not reach ~~email. Your authentication may have expired.
```

Cause: OAuth token expired or was revoked.

Resolution:

1. Go to Settings → MCP Servers
2. Find the affected server (e.g., Microsoft 365)
3. Click "Re-authenticate"
4. Complete the OAuth flow again
5. Retry the search

"No results found" When Results Should Exist

Symptom: Search returns no results, but you know the information exists.

Possible Causes and Resolutions:

CAUSE	HOW TO CHECK	RESOLUTION
Source not connected	Look at "Sources scanned" in result	Connect the missing source
Permission issue	Can you access the information directly in the source tool?	Request access or search with an account that has permissions
Query too specific	Try broader terms	Remove filters, use fewer keywords
Information too old	Check if default time filters apply	Add `after:` filter with older date
Wrong query type detected	Review results to see which sources were prioritized	Rephrase query to be more explicit

Debugging Technique:

If you know the information exists in Slack but did not appear in results:

1. Search Slack directly for the same query
2. Note what terms or filters you used in Slack to find it
3. Try enterprise search with those exact terms: `/search [exact Slack query]`
4. If Slack results appear now, the original query was too broad or ambiguous

Conflicting Results from Multiple Sources

Symptom:

```
I found conflicting information about the launch date:
- Project plan says March 1
- Slack discussion mentions March 15
- Jira milestone is still March 1
```

This is not an error — it is the plugin correctly identifying that your sources disagree.

Resolution:

1. Check timestamps: Which source is most recent?
2. Check authority: Is one source more official than others?
3. Verify manually: Check the actual sources to see which is current
4. Update outdated sources to align information

Performance Optimization

Improving Search Speed

Query Optimization:

TECHNIQUE	EFFECT	EXAMPLE
Use specific keywords	Reduces result set size, faster ranking	"PostgreSQL migration Q2" instead of "migration"
Add time filters	Limits search scope	`after:2025-01-01`
Specify source	Searches fewer sources	"in Slack: X" focuses on chat only
Use exact phrases	More precise matching	`"API migration decision"` in quotes

Source-Specific Optimization:

- **Slack**: Filter by channel (`in:#engineering`) to avoid searching all channels

- **Email**: Filter by sender (`from:sarah`) to reduce mailbox scan
- **Drive**: Search specific folders when you know the location
- **Jira**: Specify project or workspace to limit scope

Reducing Digest Generation Time

Daily digests scan 24 hours of activity across all sources, which can take 10-20 seconds for users with high activity.

Optimization Strategies:

1. **Run digests during low-activity periods**: Generate your daily digest at end of day or early morning when fewer concurrent users are searching
2. **Reduce source count**: If you do not regularly use CRM data, disconnect it to reduce digest scope
3. **Use weekly digests less frequently**: Weekly digests scan 7x more data; use sparingly

Caching Considerations

Enterprise search does not currently implement result caching (each query is fresh). For organizations with very high query volume, consider:

- **Common query caching**: If many users search the same thing (e.g., "company holidays"), cache the result for 24 hours
- **Digest pre-generation**: For very large teams, pre-generate digests overnight and serve cached versions in the morning

These optimizations require custom development.

Security and Compliance Considerations

Data Access and Privacy

Key Principle: Enterprise search respects all existing access controls from connected sources.

What This Means:

- A user can only see search results they already have permission to view
- Private Slack channels, restricted Google Drive folders, and confidential emails only appear in results for users with access
- MCP servers enforce the authenticated user's permissions

Compliance Implications:

- **GDPR**: Search results may contain personal data. Users must have lawful basis to access that data. Enterprise search does not bypass existing access controls.
- **HIPAA**: Healthcare organizations must ensure MCP connections use encrypted channels and authenticated sessions. Review each MCP server's security posture.
- **SOC 2**: Audit trails for search activity may be required. Check if your MCP servers log access (most do).

Data Residency

Question: Where does search data go during processing?

Answer:

- Query text is sent to Claude's AI model (Anthropic servers)
- Results are retrieved from MCP servers (your connected tools)
- Synthesized answers are generated by Claude and returned to your local Cowork application
- No search results are persistently stored by Anthropic

For Data Residency Requirements:

If your organization has strict data residency requirements (e.g., EU data must stay in EU), verify:

- Are your MCP servers hosted in compliant regions?
- Does your Anthropic contract specify data processing locations?
- Do you need to use a self-hosted Claude deployment?

Sensitive Information Handling

Redaction and Filtering:

Enterprise search does not automatically redact sensitive information (SSNs, credit card numbers, API keys). If your sources contain sensitive data:

1. **Preventative**: Ensure sensitive data is properly secured in source systems (should not be in Slack channels or shared Drive folders)
2. **Detective**: Audit search query logs for sensitive terms if your organization requires it
3. **Corrective**: If sensitive data appears in results, address at the source (remove from Slack, revoke Drive access)

Recommended Practice:

Create a company-specific skill that flags sensitive patterns:

```
---
name: sensitive-data-detection
description: Detect and flag potentially sensitive information in search results
---

# Sensitive Data Detection

When presenting search results, scan for patterns indicating sensitive data:

- Social Security Numbers (XXX-XX-XXXX)
- Credit card numbers (16-digit sequences)
- API keys and tokens (long alphanumeric strings)
- Passwords (password:, pw:, credentials:)

If detected, present a warning:
"⚠️ This result may contain sensitive information. Verify access permissions
before sharing."
```

Best Practices Summary

For End Users

Search Effectively:

1. Start with natural language questions, add filters only if needed
2. Use specific terms when you know them ("Project Aurora" instead of "that project")
3. Include time context when relevant ("last week", "after January")
4. Review "Sources scanned" to understand coverage

Interpret Results Wisely:

1. Check source attribution — official docs are more authoritative than chat messages
2. Note timestamps — old information may be outdated
3. When results conflict, verify with source owners before acting
4. Use search results as starting points, not final answers for critical decisions

Adopt Good Habits:

1. Run `/digest --daily` every morning to stay informed
2. When you find something important via search, ensure it is properly documented for future discoverability
3. If a search fails to find something you know exists, report the gap to your administrator

For Administrators

Deployment:

1. Start with a pilot group before organization-wide rollout
2. Connect core sources first (chat, email, documents), add specialized sources based on pilot feedback
3. Provide training on query patterns, not just feature lists
4. Monitor usage metrics to identify adoption gaps

Maintenance:

1. Review zero-result queries monthly to identify missing sources or poor query formulation
2. Check for rate limit patterns indicating users who need query optimization training
3. Keep MCP server configurations current as new sources become available
4. Audit connected sources quarterly — disconnect unused sources to reduce complexity

Support:

1. Establish a feedback channel for users to report missing results or poor relevance
2. Create organization-specific search examples in training materials
3. Document common queries and expected result patterns
4. Build a FAQ based on actual user questions

For Customizers

Extending the Plugin:

1. Use Approach C (complementary skills via CLAUDE.md) when possible — lightest touch, easiest to maintain
2. Fork (Approach B) only when changes are substantial and team-wide
3. Document all customizations in a CUSTOMIZATION.md file for future maintainers
4. Test thoroughly with real user queries before deploying custom commands or skills

Adding Sources:

1. Verify MCP server availability and authentication method before committing
2. Document new sources in CONNECTORS.md
3. Test cross-source deduplication when adding sources with overlapping content
4. Monitor rate limits after adding high-volume sources

Skill Development:

1. Write clear, specific trigger phrases in skill descriptions

2. Keep skills focused — one skill per domain, not one mega-skill
3. Use reference files for detailed content to keep SKILL.md under 3,000 words
4. Test skill activation with realistic user queries

Appendix: Quick Reference

Command Syntax

```
/enterprise-search:search <query>
/enterprise-search:search <query> from:<person> in:<location> after:<date>

/enterprise-search:digest
/enterprise-search:digest --daily
/enterprise-search:digest --weekly
/enterprise-search:digest --since <date>
```

Filter Reference

FILTER	PURPOSE	EXAMPLES
`from:`	Author/sender	`from:sarah` , `from:john.doe`
`in:`	Location/channel/folder	`in:engineering` , `in:Q1-Planning`
`after:`	Date range (start)	`after:2025-01-01` , `after:last Monday`
`before:`	Date range (end)	`before:2025-02-01` , `before:yesterday`
`type:`	Content type	`type:thread` , `type:doc` , `type:email`

Default MCP Servers

SERVER	URL	CATEGORY
Slack	`https://mcp.slack.com/mcp`	~~chat
Notion	`https://mcp.notion.com/mcp`	~~knowledge base
Guru	`https://mcp.api.getguru.com/mcp`	~~knowledge base

SERVER	URL	CATEGORY
Atlassian	`https://mcp.atlassian.com/v1/mcp`	~~project tracker, ~~knowledge base
Asana	`https://mcp.asana.com/v2/mcp`	~~project tracker
Microsoft 365	`https://microsoft365.mcp.claude.com/mcp`	~~email, ~~cloud storage, ~~office suite

Query Type Detection

QUERY PATTERN	DETECTED TYPE
"What did we decide about X?"	Decision
"What's the status of Y?"	Status
"Where's the Z doc?"	Document
"Who knows about W?"	Person
"What's our policy on V?"	Factual
"When did X happen?"	Temporal
"What do we know about Y?"	Exploratory

Confidence Level Indicators

PHRASE IN ANSWER	CONFIDENCE LEVEL
Direct statement: "The team decided X"	High
"Based on [source], X was the decision"	Moderate
"I found a reference to X, but..."	Low
"I found conflicting information..."	Uncertain/Conflicting

Troubleshooting Checklist

If no results appear:

- □ Check "Sources scanned" — are expected sources listed?
- □ Verify MCP servers are enabled in Settings

- □ Confirm you have access to the information in the source tool directly
- □ Try broader search terms
- □ Remove date filters
- □ Search each source directly to verify information exists

If results are irrelevant:

- □ Add more specific keywords
- □ Use filters (`from:` , `in:` , `after:`)
- □ Try exact phrase matching with quotes
- □ Verify query type detection matches your intent

If a source is missing:

- □ Check Settings → MCP Servers for enabled sources
- □ Re-authenticate if token expired
- □ Check for rate limit notices
- □ Verify the source contains searchable content

Conclusion

Enterprise search transforms how knowledge workers find information. By eliminating the need to search each tool individually, it saves hours every week and ensures decisions, documents, and expertise are discoverable regardless of where they were captured.

Key Takeaways:

1. **One query, all sources**: Stop searching Slack, then email, then Drive separately. Ask once, get synthesized answers from everywhere.

2. **Respect for access controls**: Enterprise search only shows you what you already have permission to see. It amplifies discoverability without compromising security.

3. **Confidence and attribution**: Every answer includes source attribution and confidence signals, so you know how much to trust the information and where to verify.

4. **Extensible and customizable**: Add new sources, create specialized commands, adapt to industry-specific needs, and encode company-specific knowledge.

5. **Daily digest workflow**: Start every day with `/digest --daily` to catch up across all your tools in one read.

Getting Started:

1. Install the enterprise-search plugin
2. Connect at least three sources (chat, email, documents)
3. Run your first search: `/enterprise-search:search [something you looked for recently]`
4. Set up a daily digest habit: `/enterprise-search:digest --daily` every morning
5. Explore advanced queries as your needs evolve

Your company's collective knowledge should not be locked in silos. Enterprise search makes it accessible, discoverable, and useful.

Document Information:

- Plugin: enterprise-search
- Version: 1.0.0
- Handbook Version: 1.0
- Last Updated: 2025-02-14
- Target Audience: Knowledge managers, IT administrators, operations teams, power users

RationalEyes.ai

contact@rationaleyes.ai

Intelligent Automation for Knowledge Work

Enterprise Search Plugin Handbook — Version 1.0.0