Gemini Enterprise & Low-Code Agents by Michaël Bettan

Gemini Enterprise (formerly **Google Agentspace**) is Google Cloud's employee-facing agent platform: an intranet search engine, an AI assistant, and an agentic platform in one web app. It answers from permissions-aware search across Google and third-party data, runs Google-made agents, lets employees build their own no-code agents, and hosts custom agents built by developers. Customer-facing conversational agents live in a sibling product, **Gemini Enterprise for Customer Experience (CX)**, whose builder is **CX Agent Studio**. Pro-code agents are built with ADK on the **Gemini Enterprise Agent Platform** (the evolution of Vertex AI).

Layer	Product	Primary audience	Build style
Employee-facing app	Gemini Enterprise (search, assistant, Agent Gallery, Workflow Builder)	Employees inside the company	No-code / low-code (Workflow Builder) plus registered custom agents
Customer-facing conversations	Gemini Enterprise for CX — CX Agent Studio (console at ces.cloud.google.com)	End customers on chat and voice channels	Minimal-code, built on ADK, with optional Python callbacks and tools
Builder platform	Gemini Enterprise Agent Platform — ADK, Agent Runtime, Agent Studio, Agent Search, Agent Gateway, Agent Registry	Developers and platform teams	Pro-code (ADK) and low-code prototyping (Agent Studio)

Old / legacy name	Current name	Note
Google Agentspace	Gemini Enterprise	Same product line, broadened to an agent platform
Agent Designer	Workflow Builder	Legacy UI name; current documentation uses Workflow Builder
NotebookLM Enterprise	Gemini Notebook Enterprise	Included with Standard and Plus editions
Vertex AI	Gemini Enterprise Agent Platform	Pro-code builder platform
Vertex AI Agent Engine	Agent Runtime	API resource name is still `ReasoningEngine`
Vertex AI Studio	Agent Studio	Low-code designer for prompts, system instructions, and agent workflows
Vertex AI Search	Agent Search	Grounding and search apps for custom builds
Dialogflow CX	CX Agent Studio (evolution)	Dialogflow CX flows can still be called from CX Agent Studio

> **TIP — THE ONE-LINE MENTAL MODEL**
> Employees → Gemini Enterprise (Workflow Builder first, register ADK/A2A agents when no-code runs out). Customers → CX Agent Studio. Developers who need full control of code, runtime and evals → ADK on Agent Runtime, then surface the agent in Gemini Enterprise or behind CX Agent Studio.

01. Editions & Product Map

Edition choice decides which builder features, data volume and security controls are available. Read the table as "what blocks me" rather than as a price list (prices are out of scope and change often).

Edition	Seats	Pooled storage	Defining traits
Business	1–500 users	25 GiB	SaaS at business.gemini.google with no Google Cloud console or IAM; no full connector ecosystem, no Data Insights, no own full-code agents, no Agent Marketplace, no enterprise-grade security and compliance controls
Standard	1+ users	30 GiB	Full Google Cloud administration; includes Gemini Notebook Enterprise
Plus	1+ users	75 GiB	Everything in Standard with a larger storage pool and higher quotas; includes Gemini Notebook Enterprise
Pay-as-you-go	1 seat minimum	Pay-as-you-go	Billed to an invoiced Cloud Billing account
Frontline (add-on)	Requires 150+ Standard or Plus users	2 GiB	For frontline workers; uses only admin-provisioned agents; cannot author no-code agents or create or upload skills

What changes between editions

- **Deep Research** is available in all editions; **Data Insights** (preview) is in Standard, Plus, Pay-as-you-go and Frontline, not Business.
- **No-code agent building** (Workflow Builder, formerly Agent Designer) is **GA** per the release notes; which editions include it can change, so check the edition comparison.

- **Custom full-code agents** (ADK, A2A) and **Agent Marketplace** require a Google Cloud–administered edition, not Business.
- **Agent creation quota** applies to registrations: Standard 1 agent per user per day, Plus 10 per user per day (pooled), Frontline 0.

Choosing between the three Google products

- **Gemini Enterprise** — you want a ready-made, permissions-aware assistant and agent hub for employees.
- **Gemini Enterprise Agent Platform** — you are building your own agentic application, model pipeline, or search experience with code (ADK, Agent Runtime, Agent Search).
- **Gemini Enterprise for CX** — you are serving customers on web chat, apps or telephony with CX Agent Studio and Agent Assist.

> **EXAM TRAP — BUSINESS EDITION IS A DEAD END FOR PLATFORM REQUIREMENTS**
> If the scenario needs IAM, VPC Service Controls, CMEK, custom ADK or A2A agents, or the full connector catalog, Business edition is the wrong answer even for a small team. Choose Standard or Plus.

02. Connectors, Data Stores & Grounding

Gemini Enterprise grounds answers in **data stores** fed by **connectors**. A connector creates a separate data store per entity type — for example, a Jira connector yields stores for issues, comments, attachments and worklogs — and search respects source-system permissions.

Mode	How it works	Strengths	Trade-offs
Federated search	Query is sent live to the third-party API; nothing is copied or stored	No storage cost, no copy of data, near-real-time	Lower answer quality; subject to the third-party's terms and rate limits
Data ingestion (indexing)	Content and ACLs are copied into a Gemini Enterprise data store and indexed	Best search and answer quality, semantic ranking	Uses storage quota, needs sync time and scheduling

Sync type	Covers	Exam-relevant gotcha
Full sync	All entities, including deletions and identity/ACL changes	Needed to propagate removed documents and permission changes
Incremental sync	New and changed entities only	Does not propagate deletions or identity changes
Entity sync vs identity sync	Content records vs user and group membership	Both must run for ACL-correct answers

Data store options

- **Structured vs unstructured** stores; regions **global, US or EU** for data residency; **CMEK** for customer-managed encryption keys.
- **Parsing**: the **layout parser** is the default for PDF, HTML, DOCX, PPTX, XLSX and XLSM and does content-aware chunking; the **OCR parser** handles scanned PDFs; the **digital parser** handles machine-readable text; you can also bring your own parsed documents.
- **Chunking** splits documents into retrievable chunks for retrieval-augmented generation (RAG); chunk quality drives answer quality.

Parser choice

The parser is set when the data store is created, as a default plus optional per-file-type overrides (`pdf`, `html`, `docx`, `pptx`, `xlsx`, `xlsm`).

Parser	What it detects	File types	Choose it when
Layout parser (default, recommended)	Paragraphs, tables, lists, titles, headings and images; adds generated descriptions for images and tables	PDF, HTML, DOCX, PPTX, XLSX, XLSM	Documents carry structure (sections, tables, figures) and you want layout-aware chunks
OCR parser	Paragraph text recognized from page images; only the first 500 pages of a PDF are processed	PDF only; other types fall back to the digital parser	PDFs are not searchable (scans, text inside images); set `useNativeText` to merge embedded text for mixed PDFs
Digital parser	Machine-readable text blocks only; no tables, lists or headings	All supported types, including TXT	Content is plain machine-readable text with no structure worth keeping
Bring your own parsed document (preview)	Whatever your own pipeline emits, as JSON text, table and list blocks	Pre-parsed JSON	You already parse elsewhere, for example with a Document AI layout parser processor

- **Fallback rule:** when the chosen parser does not support a file type, the digital parser handles it (TXT files under a layout-parser store, non-PDF files under an OCR-parser store).
- **API default:** a data store created through the API without `document ProcessingConfig.defaultParsingConfig` uses the digital parser, so request the layout parser explicitly (`layoutParsingConfig`).
- **Federated connectors:** connectors in federated mode (such as SharePoint and OneDrive) also use the layout parser, but real-time queries leave less room to analyze complex PDFs; use data ingestion mode when deep parsing matters.
- **HTML cleanup:** `excludeHtmlElements`, `excludeHtmlClasses` and `excludeHtmlIds` drop navigation menus, headers, footers and sidebars before indexing.

Chunking options

By default a search data store is tuned for document retrieval (whole documents as results). For RAG, turn on chunking at data store creation, together with the layout parser.

Setting	Values	Effect
Layout-aware chunking (`layoutBasedChunkingConfig`)	On or off at creation	Text in each chunk comes from a single layout entity (a heading section, a list), which keeps chunks coherent
`chunkSize`	100 to 500 tokens in Search stores (default 500); `500-1024` tokens (overlap `100-256`) in Document AI / RAG Engine	Smaller chunks give tighter matches on specific facts; larger chunks keep surrounding table and clause context
`includeAncestorHeadings` / `include_ancestor_headings`	`true` or `false`; default `false` (set `true` for nested tables/clauses)	Prepends the document title and H1/H2/H3 section headings so split table rows and mid-document clauses keep structural context
Search result mode	`chunks` in `ContentSearchSpec.SearchResultMode`	Returns chunks instead of documents in search responses
Bring your own chunks (preview)	Pre-segmented JSON per document	Replaces built-in chunking when you control segmentation

> **NOTE**
> The smaller-versus-larger chunk trade-off is general retrieval guidance, not a documented Gemini Enterprise benchmark. Inspect real chunks (`Chunks.list`, `getProcessedDocument`) before settling on a size.

Image-heavy PDFs, scanned forms & infographics

When documents exceed simple digital text or require corpus-wide numeric math, match the ingestion pipeline to the exact parser quotas and downstream sink:

Pipeline & stage	Hard quotas & limits	Strengths & configuration	Primary limitation / trap
Document AI Layout Parser (`GA`) → Agent Search / RAG Engine	Online sync: **15 pages** (**30 pages** with `imageless_mode`), **40 MB**. Async GCS batch: **500 pages**, **1 GB**. `ImportRagFiles` cap: **20 MB / 500 pages** per PDF	Preserves multi-column reading order and nested tables; pair `500-1024` token layout-aware chunks (overlap `100-256`) with `include_ancestor_headings = true` so split table rows retain H1/H2/H3 headers	Top-k chunk retrieval cannot compute SQL-style sums or averages across thousands of documents
Document AI Enterprise OCR Parser (`GA`)	Online sync: **15 pages** (**30 pages** with `imageless_mode`). Async GCS batch: **500 pages**, **1 GB**	Recovers text from scanned TIFFs, faxes, and non-searchable PDFs lacking embedded glyphs (`useNativeText` merges digital text on hybrid PDFs)	Flattens multi-column layouts and complex nested tables into plain reading-order text
Document AI Custom Extractor (`GA`) → BigQuery *(dual-sink pattern)*	Online sync: **15 pages**. Async GCS batch: **200 pages** (**10 pages** with checkboxes)	Extracts typed JSON fields from standardized forms (invoices, W-2s, claims) into **BigQuery** for exact SQL aggregations	Structured fields omit unstructured prose; use the **dual-sink pattern** alongside Layout Parser → Agent Search / RAG Engine
Gemini Document Understanding / LLM Parser (`GA`)	Direct `gs://` PDF input up to **50 MB**; consumes **~258–1,120 tokens/page** depending on `MediaResolution`	Interprets infographics, charts, architectural blueprints, and visually dense slides (also powers Gemini-backed Document AI layout processor versions)	Higher token cost and latency than deterministic layout parsing for routine text-only corpora

- **Dual-sink pattern for corpus-wide aggregations:** vector and hybrid chunk stores retrieve top-k passages for narrative Q&A, not aggregate math across an entire repository. Whenever users ask corpus-wide analytical questions (such as *"average payment terms across 50,000 vendor contracts"*), route standardized fields via **Document AI Custom Extractor → BigQuery** (or **BigQuery** `Object Ref` **tables +** `AI.GENERATE`) for deterministic SQL aggregation alongside **Layout Parser → Agent Search / RAG Engine** for clause-level semantic retrieval.
- **Multimodal assistant uploads:** the layout parser annotates embedded images (BMP, GIF, JPEG, PNG, TIFF) and tables with generated descriptions for indexed search, while uploading a single PDF or image directly into the Gemini Enterprise chat analyzes it in-conversation without indexing.

Standalone audio and video

Unstructured data stores accept TXT, PDF, HTML, DOCX, PPTX, XLSX and XLSM (Agent Search also lists JSON and Markdown), under 200 MB and up to 100,000 files per import, and **reject raw** `.mp3` **/** `.mp4` **media files**. Meanwhile, **Agent Search Media Data Stores** only index catalog metadata (`title`, `uri`, `categories`, `available_time`,

`duration`) plus clickstream user events (`media-play`, `media-complete`), never spoken dialogue or visual frames. Choose among four service mappings for spoken and visual content:

Service mapping	What gets indexed / processed	Timestamps & metadata	Access control (ACLs)	Quotas & technical knobs	Key trade-off / trap
1. Cloud Speech-to-Text V2 (`Chirp 3`, `GA`) → Agent Search Unstructured Store	Speaker-diarized transcripts ingested as HTML/TXT segments with JSONL metadata rows pointing to the source recording	Word- and segment-level `start_offset` / `end_offset` stored in `structData` for deep-linking	Native **Workforce Identity / Google Workspace per-user ACLs** via `acl_info` readers (up to 3,000 readers/doc)	Streaming cap **~5 min**; async GCS `BatchRecognize` handles **multi-hour** audio/video archives	Loses purely visual cues unless paired with frame captioning; raw `.mp3` / `.mp4` upload to unstructured stores fails
2. Cloud Video Intelligence API (`GA`) → JSONL / BigQuery + Agent Search	Deterministic shot-change detection, video OCR (on-screen text), object tracking, and explicit content moderation	Exact `startTimeOffset` / `endTimeOffset` JSON annotations per detected entity or shot	Enforced on downstream BigQuery tables or Agent Search `acl_info` rows	Multi-GB async GCS batch processing; billed per enabled feature-minute	Emits discrete visual/OCR labels rather than causal reasoning; requires an **LLM synthesis pass** for narrative Q&A
3. Gemini Direct Multimodal (`gs://` URI + `audio_timestamp=True` / `mediaProcessing="AGENTIC"`)	Zero index; reads raw media from Cloud Storage at query time	Set `audio_timestamp=True` (`audioTimestamp` in REST) in `GenerationConfig`; prompt for `MM:SS` video citations	Bucket/app-level IAM only; no per-document Workforce ACLs	**~8.4 hrs** audio or **~45 min** video+audio (**~1 hr** silent video) per 1M window; **max 10 videos/prompt**; `MEDIA_RESOLUTION_LOW` cuts frame tokens; keep opaque **`step_list`** across turns in Agentic Video (`GA`)	Re-reads tokens on every question; high cost and latency at archive scale
4. Agent Retrieval + `gemini-embedding-2` (`Preview` / rolling `GA`)	One `DataObject` per short segmented clip in a Collection with **3,072-dim** multimodal vectors	Clip-level `start_offset` / `end_offset` stored as `DataObject` fields	**No turnkey Workforce IdP ACL sync**; enforce via explicit principal/group query filters	Max **8,192 tokens** per embedded item (~120 s MP4/MPEG video, ~80 s with audio, ~180 s audio; convert MOV first)	Long recordings past 8,192 tokens truncate unless pre-split into clips; retrieves clips only (pass top clips to Gemini to answer)

Transcript segment metadata (JSONL, wrapped here; one line per record in the file):

```
{"id": "townhall-q3-seg-014",
 "structData": {"title": "Q3 town hall", "recording": "gs://media/townhall-q3.mp4",
   "start_offset": "00:14:20", "end_offset": "00:16:05"},
 "acl_info": {"readers": [{"principals": [{"group_id": "all-staff@example.com"}]}]},
 "content": {"mimeType": "text/html", "uri": "gs://media-text/townhall-q3/seg-014.html"}}
```

Choose this when:

- **Speech-to-Text V2 (`Chirp 3`) → Unstructured Store:** employees search thousands of multi-hour meetings, support calls, or trainings inside Gemini Enterprise with speaker diarization, hybrid lexical+semantic ranking, and strict Workforce Identity / Workspace per-user ACL enforcement.
- **Cloud Video Intelligence API → JSONL / BigQuery + Agent Search:** compliance or media pipelines need deterministic shot boundaries, on-screen text OCR, object tracking, or content moderation timestamps across multi-GB video batches before an LLM synthesizes answers.
- **Gemini Direct Multimodal:** an engineer or incident responder needs deep cross-modal reasoning over 1 to 10 specific recordings right now (such as an outage bridge call or screen-share bug report), preserving `step_list` across turns when using `processing="agentic"` / `mediaProcessing="AGENTIC"` (`GA`).
- **Agent Retrieval + `gemini-embedding-2`:** a custom agent matches natural-language queries to visual/audio clips (*"forklift near the loading dock"*) across a segmented clip catalog with app-enforced principal/group metadata filters.
- **Agent Search Media Data Store:** a streaming or learning portal needs catalog search and personalized *"watch next"* recommendations driven by catalog fields and clickstream events (`media-play`, `media-complete`), not spoken-content Q&A.
- **Anti-pattern (media store as a video brain):** loading recordings into a Media Data Store and expecting answers about spoken dialogue or visual scenes; it indexes only catalog metadata and user events.
- **Anti-pattern (whole recordings as embeddings):** sending a 60-minute video directly to `gemini-embedding-2`; inputs exceeding 8,192 tokens or per-clip duration caps must be segmented with `start_offset` / `end_offset` metadata first.

NOTE

Duration and token figures reflect published Gemini 2.5 and 3.x Flash/Pro capability tables and vary by model checkpoint and `Media Resolution`. Verify target model limits before sizing production capacity.

Identity and access

- Identity provider: **Google Identity** (recommended) or **Workforce Identity Federation** (the workforce pool must be global); one IdP type per location.
- Search enforces source ACLs — users never see documents they cannot open in the source system (no privilege escalation). Third-party SaaS identities are mapped to the IdP identities.
- A document's ACL can include up to 3,000 readers.

EXAM TRAP — FEDERATION IS NOT "FREE QUALITY"

Federated search avoids copying data but gives lower-quality answers than ingestion. When the requirement is "best answer quality on Confluence content", pick ingestion; when it is "data must never be copied out of the SaaS tool", pick federation.

KEY FACT — AGENT SEARCH FOR CUSTOM BUILDS

When you build your own app or ADK agent instead of using the Gemini Enterprise web app, ground it with **Agent Search** (formerly Vertex AI Search) on the Agent Platform. Gemini Enterprise is the packaged employee experience; Agent Search is the developer building block.

03. Google-made Agents, Agent Gallery & Sharing

Agent type	Who builds it	Examples / notes
Google-made agents	Google	Core assistant, Deep Research, Gemini Notebook, Data Insights (preview), Co-Scientist
Workflow Builder agents	Employees (no-code)	Chat agents and workflows; admin must turn on "Enable Workflow Builder"
Custom ADK agents	Developers	Hosted on Agent Runtime, then registered in Gemini Enterprise
Custom A2A agents	Developers or partners	Registered via agent card (preview) or imported from Agent Registry
Dialogflow agents	Conversational designers	Existing Dialogflow agents surfaced in Gemini Enterprise
Marketplace agents (preview)	Partners	A2A agents procured from Google Cloud Marketplace and integrated by an admin

Agent Gallery

- Filters: **All, Pinned, Made by Google, From your organization, Your agents, Marketplace**; users can pin favorites.
- **+ New agent** opens Workflow Builder; users invoke an agent in chat with **@agent_name**.
- Agents can also be invoked programmatically through the **StreamAssist API** or the agent's Agent Registry A2A endpoint.
- Admins can view each agent's identity (a SPIFFE ID, falling back to the Agent Registry resource ID).

Agent state	Visible in gallery?	Users can interact?
Private	Only to the owner	Owner only
Enabled	Yes	Yes
Suspended	Yes	No
Disabled	No (hidden)	No

Sharing controls (admin)

- **Enable agent sharing** — allow users to share agents they built.
- **Enable agent sharing without admin approval** — skip the approval step.
- **Enable End Users to share with Groups** — allow sharing to Google Groups instead of individuals.

EXAM TRAP — IDEA GENERATION IS GONE

The Idea Generation agent (preview) was removed in July 2026. Scenarios about brainstorming or hypothesis generation should point to the assistant, Deep Research, or Co-Scientist.

04. Agent Designer → Workflow Builder

Workflow Builder (renamed from **Agent Designer**) is the **GA** no-code workforce builder inside **Gemini Enterprise**. Do not confuse it with Google Cloud's sibling consoles:

- **Workflow Builder (Gemini Enterprise):** no-code employee builder for **Chat Agents** and multi-step **Workflow Agents** over Google Workspace and enterprise SaaS connectors with built-in human-in-the-loop (HITL) review.
- **Agent Studio (Agent Platform Console):** developer-facing low-code prompt, system instruction, and agent prototyping surface that **exports starter ADK Python code** and supports **Deploy as A2A** (`Preview`) when deploying to an **Agent Runtime** instance.
- **CX Agent Studio (Gemini Enterprise for CX):** customer-facing omnichannel chat and bidirectional voice/telephony builder backed by ADK callbacks and deterministic Dialogflow CX state-machine flows (**max flow stack depth = 25**).
- **ADK (pro-code):** full Python/TypeScript/Java/Go SDK for custom graph topologies, lifecycle callbacks, `Live API` streaming, and custom storage or sandbox backends.

Artifact	Behavior	How it starts	Best for
Chat agent	Conversational and non-deterministic; single- or multi-step with subagents	User chats with it (`@agent_name`); can also be scheduled	Q&A over team knowledge, drafting, research helpers
Workflow	Ordered steps with variables passed between them; supports branching, loops, and native HITL gates	Manual (`@agent_name`), Schedule (including cron), or Event (connector data change)	Repeatable business processes, triage, approvals, report generation

Authoring surfaces: Conversational Designer, Editor Canvas & Details pane

- **Two authoring modes:** describe the agent or workflow in natural language inside the **Conversational Designer**, or open the visual

Editor Canvas (**Proceed to builder**) to wire steps directly. Prompt/conversational mode cannot attach Google Cloud data sources (BigQuery, Cloud Storage, Cloud SQL); switch to the visual builder canvas for those sources.

- **Details pane vs. step-level prompts:** configure global agent persona, default model (Flash for low latency, Pro for complex reasoning; Gemini 3.0+ on workflow agent nodes), provisioned data stores/tools, knowledge files, and starter prompts (up to 3 shown at random) in the **Details** pane. Configure **step-level prompt templates**, **inline few-shot examples**, chain-of-thought instructions, and structured **JSON Schema** outputs directly on individual **Gemini Agent** canvas nodes.
- **Preview, versions & Runs tab:** test drafts interactively in **Preview** (leaving without **Create** saves a **Draft**), compare or restore saved **Versions**, and inspect step-by-step inputs, outputs, execution logs, and paused human-review tasks in the **Runs** tab.

Native HITL steps (`Request info` & `Approval`) and publishing

- **Native HITL steps (`GA`):** Workflow Builder includes two first-class human-in-the-loop step types alongside **Gemini Agent** and **Condition** nodes. Both pause execution in `Needs Review` status inside the **Runs** tab until an authorized reviewer responds: 1. `Request info`: pauses the workflow to collect structured form inputs or missing parameters from a human before downstream steps resume. 2. `Approval`: enforces an explicit **Approve / Reject** gate before executing mutative connector actions (such as updating Salesforce records or posting announcements).
- **Publishing & A2A interoperability:** Workflow Builder publishes workforce agents internally to the **Gemini Enterprise Agent Gallery** (`Turn on` / `Publish`, invoked via `@agent_name` or the StreamAssist API) and **consumes** existing **ADK** and **A2A** agents as workflow steps (**Add step > Existing agents**, pausing cleanly when a downstream agent asks clarifying questions via A2A `input-required` / `TaskUpdater.requires_input()` or ADK `LongRunning FunctionTool` / `RequestInput`) or chat sub-agents. Conversely, **Deploy as A2A** (`Preview`) is an **Agent Studio** option (*Deploy to an Agent Runtime instance*) that registers a prototype in **Agent Registry** as an A2A endpoint—not a Workflow Builder toggle. Direct one-click binary import from Workflow Builder into CX Agent Studio state machines is **not publicly documented**.
- **Supported workflow connectors:** Gmail, Calendar, Chat, Drive, Confluence, Jira, Teams, Notion, OneDrive, Outlook, Salesforce, SharePoint, ServiceNow, and Slack.

Quotas, connector dependencies & custom MCP traps

- **Quotas & undocumented boundaries:**
- **Max `100` enabled actions** per custom MCP server data store (if an MCP server exposes `> 100` tools, curate or filter the exposed toolset down to `⩽ 100` actions).
- **Max `50` data stores** per blended Gemini Enterprise / Agent Search app.
- **Uploaded knowledge files:** placed into context every turn up to **800,000 tokens** combined (attach an indexed data store instead for libraries larger than ~100 pages).
- **Undocumented caps:** hard limits on **maximum workflow canvas steps** and the **internal session compaction token threshold** in Workflow Builder are *not publicly documented*.
- **Connector & MCP traps:**
- **Connector search/write toggle trap: Search For Data** actions are enabled per action, whereas **Add Or Update Data** (write) is an all-or-nothing connector-level toggle with human review. Disabling *"Search For Data"* breaks *"Add Or Update Data"* with `ENTITY_NOT_FOUND`, because write operations rely on search actions to resolve target entity IDs.
- **Custom MCP transport trap:** custom MCP server connectors require `StreamableHTTP` transport; legacy `SSE` MCP endpoints are rejected.
- **Custom MCP `readOnlyHint` prompt trap:** custom MCP tools must explicitly set `readOnlyHint: true` in their tool annotations to suppress user confirmation prompts on read operations; setting only `destructiveHint: false` is insufficient and still triggers confirmation prompts on every read call.
- **Knowledge scoping rule:** a full Drive or OneDrive connector tool overrides restricted file knowledge on the same node; split into two separate Gemini Agent nodes to keep access scoped. Builders can only attach connectors and actions provisioned by an admin.

State-based design and prompt craft

- A workflow is a **visible state machine**: canvas steps are states, **Condition** nodes are transition routes, `Request info` / `Approval` steps are human gates in **Runs** → `Needs Review`, and typed variables carry state forward.
- Author step-level prompt templates with role, goal, constraints, inline few-shot examples, and JSON Schema output contracts whenever downstream steps parse fields programmatically. For customer-facing state machines with pages, slot-filling forms, and telephony event handlers, use CX Agent Studio and Dialogflow CX flows (sections 07–08).

> **EXAM TRAP — CHAT AGENT VS WORKFLOW**
> "Every Monday, pull new Jira bugs, summarize, and ask a lead to approve before posting to Chat" is a **workflow** (Schedule trigger, Gemini Agent step, `Approval` HITL step pausing in `Needs Review`). An open-ended "ask me anything about our HR policies" helper is a **chat agent**.

05. Bringing Custom ADK, A2A & Dialogflow Agents

When no-code is not enough, developers build agents in code and **register** them so employees use them inside Gemini Enterprise.

Path	Hosting requirement	Status	Key setup
ADK agent	Must run on **Agent Runtime**	Available (verify launch stage per feature)	Register in console, API, or with a single Agents CLI command; needs Gemini Enterprise Admin role and Discovery Engine API
A2A agent	Any host exposing an A2A agent card (e.g., Cloud Run)	Preview	Paste the agent card JSON; A2A v0.3 streaming supported (v1.0+ via compatibility packages)
Agent Registry import	A2A agent already in Agent Registry	Available (verify launch stage)	Import a governed agent instead of re-registering by hand
Dialogflow agent	Dialogflow	Available (verify launch stage)	Surface an existing conversational agent
Marketplace agent	Partner-hosted A2A agent	Preview	Procure in Cloud Marketplace, admin integrates and sets visibility

ADK agent registration details

- Connection is VPC-SC compliant; the Agent Runtime instance can be in a different project.
- The agent receives the signed-in user's email.
- Optional **OAuth authorization resource** so the agent can act on third-party systems as the user: create an OAuth client (Web application) with redirect URIs `https://vertexaisearch.cloud.google.com/oauth-redirect` and `https://vertexaisearch.cloud.google.com/static/oauth/oauth.html`, then supply `clientId`, `clientSecret`, `authorizationUri` and `tokenUri` in `serverSideOauth2`. The registration references the engine via `adkAgentDefinition.provisionedReasoningEngine`.

A2A agent registration details

- On Cloud Run (`*.run.app`), grant `roles/run.invoker` to the Discovery Engine service agent; Gemini Enterprise then sends an OIDC token in `X-Serverless-Authorization` (not for custom domains, load balancers or gateways).
- The end user's OAuth token travels in the `Authorization` header.
- Agent card `securitySchemes`, API keys and custom headers are not used; if you front the agent with Apigee, validate the token there (VerifyJWT or a tokeninfo ServiceCallout).
- Traffic to A2A agents registered this way **does not go through Agent Gateway**.
- Each registration consumes the daily agent creation quota.

> **EXAM TRAP — CONSOLE MODEL ARMOR DOES NOT COVER CUSTOM AGENTS**
> The Model Armor setting in the Gemini Enterprise console screens the assistant only. ADK, A2A and Marketplace agents must call Model Armor (or equivalent guardrails) **inside the agent code**.

06. Governance & Admin Controls

Control	Where	What it governs
End-user feature toggles	Gemini Enterprise admin console	"Enable Workflow Builder" and other features per app
Sharing settings	Admin console	Agent sharing, approval-free sharing, sharing with Groups
Agent states	Admin console	Private, Enabled, Suspended, Disabled per agent
Connector provisioning	Admin console	Which data sources and actions builders can use
Identity provider	Google Identity or Workforce Identity Federation	Who users are; ACL enforcement
Model Armor templates	Admin console + Model Armor	Prompt and response screening for the assistant
VPC Service Controls, CMEK, data residency	Google Cloud project	Exfiltration perimeter, key ownership, global/US/EU location
Agent identity, Agent Registry, Agent Gateway	Agent Platform	Per-agent identity (SPIFFE), catalog of governed agents, policy enforcement on agent traffic
IAM roles	Google Cloud IAM	Gemini Enterprise Admin for registration; least privilege for service agents

Model Armor for the assistant

- Screens assistant prompts and responses; available in all editions at no extra cost, may add latency.
- Recommended: two templates — a **prompt template** (prompt injection, jailbreak, sensitive data) and a **response template** (data leakage, harmful content, malicious URLs).

- Uses **INSPECT_AND_BLOCK**; it blocks rather than de-identifies. A violation returns `answer.state = SKIPPED` with `CUSTOMER_POLICY_VIOLATION`.

Layered governance pattern

- **Data layer**: ACL-aware connectors, full syncs for permission changes, data residency, CMEK.
- **Agent layer**: admin-provisioned connectors only, human review for write actions, sharing approval, suspend or disable misbehaving agents.
- **Traffic layer**: Model Armor (console for assistant, in-code for custom agents), Agent Gateway for agent traffic on the platform, OAuth on behalf of the user.
- **Network layer**: VPC Service Controls around Gemini Enterprise, Agent Runtime and data sources.

> EXAM TRAP — "SUSPEND" VS "DISABLE"
> Suspended agents stay visible in the gallery but cannot be used (good for a temporary freeze with transparency); Disabled agents are hidden entirely.

07. CX Agent Studio for Customer-Facing Agents

CX Agent Studio is the minimal-code builder in **Gemini Enterprise for CX** (console: ces.cloud.google.com). It is built on ADK and is the evolution of Dialogflow CX, adding asynchronous tool calls, low-latency bidirectional streaming voice, and versioning with rollback.

Agent application structure

- An **agent application** is a tree: a **root (steering) agent** that routes to **sub-agents**.
- Set the model globally or override it per sub-agent; design in English with automatic language detection for end users.
- AI assistance: generate an agent from a description, **Restructure instructions** into XML tags (`role`, `persona`, `primary_goal`, `constraints`, `taskflow`, `subtask`, `step`, `trigger`, `action`, `examples`), **Refine instructions**, and hill-climb against test cases.

Instructions

- Reference tools as `{@TOOL: tool_name}`, sub-agents as `{@AGENT: agent_name}`, and variables as `{username}`.
- **Global instructions** are inherited by every agent — put brand tone and DOs/DON'Ts there.
- Modular conditional components (if/elif/else by channel, modality or variable values) keep one agent valid for chat and voice.
- **Inline few-shot examples** (`[user]`, `[model]`, `tool_code`, `tool_outputs`) teach format and tool use; use them sparingly — clear instructions first. The XML `taskflow` → `step` structure is the in-console equivalent of an explicit chain-of-thought procedure.

Tool type	Use it for
Data store / File search / Google Search	Grounded answers from enterprise content, uploaded files, or the web
OpenAPI / Integration Connector / MCP	Calling business APIs and enterprise systems
Salesforce, ServiceNow, Jira, Confluence, SharePoint	Prebuilt SaaS actions
Python code / Client function / Widget	Server-side logic, client-side actions, rich UI elements
Agent as a tool / A2A (preview)	Delegating to another agent while keeping the session
System tools (e.g., `end_session`) / Google Maps (preview)	Session control and handoff; location answers

Tool execution & callbacks

- Tool names are snake_case. **Synchronous** tools should finish in under ~5 s; **asynchronous** tools (≈5–60 s) let the agent keep talking while the call completes.
- **Callbacks** are Python hooks equivalent to ADK: `before_agent` / `after_agent`, `before_model` / `after_model` (returning an `Llm Response` skips the model), `before_tool` / `after_tool` (returning a dict skips the tool). Use them for deterministic guardrails, canned responses, and redaction.
- **A2A tool** delegation: the primary agent keeps the session; exchanges are text only; billing counts one session in the same project, otherwise two; roles include `roles/ces.client`, `roles/run.invoker` and `roles/aiplatform.user`, with Secret Manager for third-party credentials.

Evaluation test case	What it checks
Scenario	An AI-simulated user pursues a goal; tests open-ended robustness
Simulation	Ordered goals and milestones; tests multi-step conversations
Golden	An ideal path including tool calls; regression testing with semantic match (0–4)

Evaluation metrics

- Tool correctness, user goal satisfaction, goal completion, hallucinations, semantic match (golden only), scenario expectations; personas vary the simulated user.
- **Task completion** = user goal satisfied AND no hallucinations AND scenario expectations met.

08. Flows, State Handlers & Human Handoff

Dialogflow CX models conversations as explicit state machines. CX Agent Studio can call these flows when you need deterministic control.

Dialogflow CX concept	Meaning
Flow	A conversation topic (e.g., "Authenticate caller") made of pages
Page	A conversational state; can collect form parameters
State handler	Has requirements, an optional fulfillment (response, parameter preset, webhook) and an optional transition target
Route	Intent route or condition route that moves to another page or flow
Event handler	Reacts to events such as `sys.no-match-default`, `sys.no-input-default`, or a sub-flow ending with a request for a human (`flow-failed-human-escalation`, raised by the `END_FLOW_WITH_HUMAN_ESCALATION` target)
Handler scope	Flow-level, page-level, or form-parameter-level
Form parameters	Slots a page must fill before it completes

Page anatomy and form filling

- **One active page per active flow.** A session always sits on exactly one page; a page bundles an **entry fulfillment** (messages or a webhook that run when the page becomes active), an optional **form**, and its **routes and event handlers**.
- **Form filling** runs turn by turn: parameters already present in session state pre-fill the form, the agent prompts only for the missing ones, and each parameter can carry its own reprompt handlers (`sys.no-match-1`, `sys.no-input-1` and so on).
- **Leaving the page** is usually a condition route on `$page.params.status = "FINAL"`, which becomes true once every required parameter is filled.
- **Typed extraction:** bind each parameter to a system entity (`@sys.date`, `@sys.number`) or a custom entity so values are validated, not guessed.

Handler evaluation order

1. **Scope filter:** only handlers attached to the active flow, the active page, or the parameter being filled are considered.
2. **Intent routes:** page routes, then page route groups, then flow routes, then flow route groups, each in the order you listed them.
3. **Condition routes:** page routes, then page route groups; flow routes and flow route groups only when the current page is the flow's start page.
4. **Event handlers:** parameter-level (reprompt) handlers, then page, then flow, when an event fires (no match, no input, webhook failure, flow outcome).

- **Stop rule:** a called handler with a transition target ends evaluation; a handler without one lets evaluation continue, so several fulfillments can queue several messages in one turn.
- **Consumption:** a matched intent or event is consumed by the first handler that uses it, while a satisfied condition is not, so several condition routes can fire in the same turn.

Built-in event	Fires when	Design use
`sys.no-match-default`, `sys.no-match-1` to `-6`	Input matches no in-scope intent or does not satisfy the form parameter	Numbered handlers replace the default and run in order: rephrase, give an example, then escalate
`sys.no-input-default`, `sys.no-input-1` to `-6`	Empty text, no recognized speech, or a no-speech timeout	Voice reprompts and a graceful exit after repeated silence
`sys.invalid-parameter`	A webhook marks a form parameter `INVALID`	Server-side validation (for example an unknown account number)
`sys.long-utterance`	Input over 256 characters for non-generative intents or parameters	Ask for a shorter answer instead of treating it as no-match
`webhook.error` and specific variants (`webhook.error.timeout`, `webhook.error.bad-request`, `webhook.error.rejected`)	Timeout, error status, invalid response, or unavailable service	Fallback message or alternate path; fires only if the calling route has no transition target of its own
`flow-cancelled`, `flow-failed`, `flow-failed-human-escalation`	A sub-flow ended with cancellation, failure, or a request for a human	Handle the outcome on the calling page

Symbolic transition targets

Target	Effect
`START_PAGE`	Go to the start page of the active flow
`CURRENT_PAGE`	Re-enter the current page (useful to repeat a prompt)
`PREVIOUS_PAGE`	Return to the page that led here and restore its page state
`END_FLOW`	Pop the active flow and resume the calling page; handlers already evaluated there are skipped
`END_FLOW_WITH_CANCELLATION` / `END_FLOW_WITH_FAILURE` / `END_FLOW_WITH_HUMAN_ESCALATION`	Pop the flow and raise the matching `flow-*` event on the calling page
`END_SESSION`	Clear the session; the next input restarts at the start page of the Default Start Flow

- **Flow stack limit:** each flow-to-flow transition pushes onto a stack capped at **25**. Going deeper can cause flows to be popped from the stack, so `END_FLOW` returns somewhere unexpected; keep sub-flow chains shallow. With an empty stack, `END_FLOW` ends the session.

Sample flow: order capture

```
[Start page] —intent: order.new—► [Order details] (form: product, qty)
                                    │ sys.no-match-1 → rephrase prompt
                                    │ sys.no-match-2 → offer examples
                                    │ webhook.error  → [Fallback page]
                                    └ $page.params.status = "FINAL"
                                              ▼
                                   [Confirm page] —yes—► END_SESSION
                                          └ no —► PREVIOUS_PAGE
```

Combining playbooks, generators and flows

- **Playbooks** are the generative building block of Dialogflow CX agents: a goal, step instructions, examples (few-shot conversations) and parameters. **Task playbooks** are reusable subtasks that exchange input and output parameters; **routine playbooks** model sequential stages and can hand off to other routine playbooks or to flows.
- **References in playbook instructions:** `${PLAYBOOK: name}` routes to another playbook, `${TOOL: name}` calls a tool, and `${FLOW: name}` hands control to a deterministic flow.
- **Generators** call an LLM from inside a fulfillment with a prompt template (placeholders such as `$last-user-utterance` or session parameters) for summarization, parameter extraction or rewriting, without writing a webhook.
- **Session parameters** are the shared memory between the generative and deterministic halves: a routine playbook writes them and a flow reads them, or the reverse (task playbooks pass data through input and output parameters instead).
- **Playbook examples** are sample conversations (user turns, tool calls with inputs and outputs, agent replies) that act as few-shot prompts. Give every playbook at least one and aim for four or more, covering happy paths plus edge cases such as a tool returning no data; example quality usually drives behavior more than instruction wording. For multilingual agents, write examples in each language.
- **Example Store (preview, Agent Platform)** is the code-agent counterpart: a managed set of few-shot examples retrieved by similarity to the user query. An ADK agent linked to an instance pulls relevant examples into its requests automatically, and new examples take effect without redeploying, which makes it a way to correct observed misbehavior.

Pattern	Shape	Good fit
Generative front, deterministic back	A playbook handles open discovery, then `${FLOW: name}` runs payment or identity steps	Sales or support where the closing steps are regulated
Deterministic spine, generative pages	A flow drives the process; a generator or data store tool answers side questions on one page	Claims intake that also fields policy questions
Flow as subroutine	A CX Agent Studio root agent routes to a flow-based agent and gets output variables back	Reusing an existing Dialogflow CX authentication flow

Flow-based agents in CX Agent Studio

- A flow-based agent hands the conversation to a Dialogflow CX flow until it ends (END_SESSION), using DetectIntent; input and output variables are mapped between the two.
- The Customer Engagement Suite service agent needs the Dialogflow API Client role.
- CX Agent Studio agents are not drop-in replacements for Dialogflow playbooks; use a CX Agent Studio agent as the steering and routing layer.
- Use flows for deterministic data collection, validation and authentication. Do not use flows for canned responses (use a callback) or intent classification (let the LLM route). Treat flows as black boxes — session parameters behave like implicit globals.

Human handoff with end_session

- Add the `end_session` system tool to the root agent and every sub-agent that may escalate.
- Parameters (camelCase since August 1, 2026; old ALL_CAPS names remain supported): `escalationMessage`, `telephonyTransferCall` (`phoneNumber`, `sipUri`, `useOriginatingTrunk`), `liveAgentHandoff` (`endSession`, `sipRefer`, `uuiHeaders`, `xHeaders`).
- With **Agent Assist**, keep `liveAgentHandoff.endSession` at its default `False` so the conversation transfers to the human agent with assist; set it to `True` only when not using Agent Assist. Agent Assist handoff works with the bidirectional API.
- After a session ends, CX Agent Studio cannot reconnect to it.

`end_session` payload (transfer to a billing agent with Agent Assist):

```
{
  "escalationMessage": "Transferring your call to a billing
agent",
  "telephonyTransferCall": {"phoneNumber": "+15552420836",
"useOriginatingTrunk": true},
  "liveAgentHandoff": {"endSession": false}
}
```

> EXAM TRAP — DETERMINISTIC DOES NOT MEAN "ALL FLOWS"
> Use LLM agents for understanding and routing, flows or callbacks for steps that must never vary (identity verification, payment capture, legally required wording).

State-machine design mistakes

Mistake	What goes wrong	Fix
No no-match or no-input handlers	Users loop on the same prompt or the call goes silent	Add numbered `sys.no-match-*` and `sys.no-input-*` handlers at flow and parameter level, ending in escalation
Circular routes between flows and playbooks	Turns repeat until the user hangs up; deep chains overflow the 25-flow stack	Count retries in a session parameter and add a page-level condition route that exits after a limit; keep sub-flow nesting shallow
LLM-extracted values for strict fields	Missing or invented account numbers, dates and amounts	Use page forms with typed entities and webhook validation (`sys.invalid-parameter`)
Reference data pasted into instructions	Higher latency and token cost on every turn; stale facts	Move FAQs and policies to a data store tool and keep instructions about behavior
Webhook route that also sets a transition target	`webhook.error` never fires, so failures pass silently	Leave the target off routes whose webhook can fail, and handle the error event explicitly

Mistake	What goes wrong	Fix
Broad intents placed on flow-level routes	Flow intent routes stay in scope on every page, so a loosely trained intent can pull the user out of a half-filled form	Attach specialized intents to the pages or route groups where they apply; keep flow-level routes for true global exits such as "talk to a person"
No instruction for empty tool or data store results	The playbook fills the gap with invented answers	Instruct it to say it does not know or ask the user to rephrase when a tool returns nothing, and add an example showing that path
State-changing action with no confirmation	Orders, refunds or record updates run on a misheard or misread value	Add a confirmation page (read back the values, then a yes or no route) before the webhook; in Workflow Builder, put a Human-in-the-loop approval step before the write

09. Low-Code vs Pro-Code Decision Matrix

Criterion	Workflow Builder (Gemini Enterprise)	CX Agent Studio (Gemini Enterprise for CX)	Agent Studio (Agent Platform Console)	ADK (Agent Runtime / Cloud Run / GKE)
Audience	Employees & business ops	External customers	Developers prototyping	Developers and platform teams
Channels	Gemini Enterprise web app (`@agent_name` , Agent Gallery, StreamAssist API); consumes ADK/A2A steps	Web chat, mobile apps, bidirectional telephony / voice	Console playground (`design-agents`); **Deploy as A2A** (`Preview` on Agent Runtime) or **export starter ADK Python code**	Any (API, Gemini Enterprise, CX via A2A, custom UI, `Live API`)
Authoring & coding	Zero-code (**Conversational Designer** + visual **Editor Canvas**; **Details** pane + step prompts/few-shots)	Minimal-code (instructions, XML `taskflow` , optional Python callbacks & webhooks)	Low-code prompt, system instruction & tool designer	Full pro-code (Python, TypeScript, Java, Go)
Control flow & HITL	Chat agents or step workflows with `Condition` branches and native **`Request info`** / **`Approval`** HITL (`Needs Review` in **Runs**)	Steering root agent + sub-agents + deterministic Dialogflow CX flows (**max flow stack = 25**) + `end_session` handoff	Single/multi-step prompt & tool prototyping	Sequential, parallel, loop, and dynamic unbounded `@node` graph workflows
Grounding & tool limits	Provisioned connectors, max **50 data stores** per blended app, max **100 actions** per `StreamableHTTP` MCP store, 800k-token uploaded files	Data store, file search, Google Search, OpenAPI, Integration Connectors, MCP, A2A (`Preview`)	Google Search, Maps, RAG Engine, Agent Search, MCP	Agent Search, RAG Engine, Agent Retrieval, `McpToolset` , `ToolboxToolset` , `ApplicationIntegrationToolset` , `ApiHubToolset`
Evaluation	Interactive **Preview**, **Runs** execution logs, saved **Versions**	Scenario, simulation and golden test cases	Side-by-side prompt & model comparison	`adk conformance` , `adk eval` , `agents-cli eval` , Gen AI Evaluation Service (`CustomMetric` + autoraters)
Governance	Admin toggles, sharing approval, console Model Armor (assistant only), Workforce/Google IdP ACLs	Versioning, rollback, Python callbacks, IAM	Project IAM	SPIFFE Agent Identity, PAB, Agent Gateway (`REQUEST_AUTHZ` + `CONTENT_AUTHZ`), in-code Model Armor

Four code-only capabilities requiring ADK / pro-code

Escalate from Workflow Builder (or graduate from Agent Studio's exported starter code) to **ADK** whenever the architecture requires any of these four pro-code capabilities:

1. **Deterministic lifecycle callbacks on workforce/platform agents:** in-process `before_model_callback` , `after_model_callback` , `before_tool_callback` , `after_tool_callback` , and `on_model_error_callback` hooks for custom guardrails, PII tokenization, schema normalization, or fine-grained budget enforcement.
2. **`Live API` bidirectional audio and video streaming with barge-in:** custom real-time multimodal WebSocket/WebRTC experiences outside packaged CX Agent Studio telephony channels.
3. **Dynamic, unbounded `@node` graph topologies:** runtime-computed cyclic or arbitrary multi-agent orchestration graphs beyond static visual canvas branching (note also that direct binary migration from Workflow Builder into CX Agent Studio state machines is *not documented*).
4. **Custom state, cache, sandbox, and evaluation backends:** `DatabaseSessionService` (AlloyDB / Cloud SQL / Spanner), Memorystore `< 1 ms` idempotency and semantic caches, GKE `gVisor` **Agent Sandbox** pods for untrusted code, and deterministic Python **`CustomMetric`** / `adk conformance` CI pipelines.

Decision rules

- Employee productivity or approval workflow over provisioned SaaS/Workspace connectors → **Workflow Builder** (can also invoke existing ADK and A2A agents via **Add step > Existing agents**, handling A2A `input-required` and ADK `RequestInput` clarifications).
- Customer-facing omnichannel chat or telephony with deterministic slot-filling flows (`⩽ 25` stack depth), Agent Assist handoff, and golden/simulation suites → **CX Agent Studio**.
- Rapid developer prompt and tool prototyping with one-click code handoff or A2A endpoint exposure → **Agent Studio** (**Deploy as A2A** `Preview` to Agent Runtime, or export starter Python code into **ADK**).
- Any of the 4 code-only capabilities above, cross-channel reuse, or strict CI/CD conformance gates → **ADK on Agent Runtime** (or Cloud Run / GKE), then **register** in Gemini Enterprise.
- Partner or non-ADK external agent → expose over **A2A** and register via Agent Registry or agent card.

10. Exam Traps & Architect Checklist

EXAM TRAP — NAME DRIFT & CONSOLE DISAMBIGUATION

Agent Designer = **Workflow Builder** (no-code employee builder in Gemini Enterprise); **Agent Studio** = developer prototyping console in Agent Platform that exports starter ADK Python code (and offers **Deploy as A2A** `Preview` on Agent Runtime); **CX Agent Studio** = customer-facing omnichannel/telephony builder (max flow stack = 25; no direct binary import from Workflow Builder). Agentspace = Gemini Enterprise; Vertex AI Search = Agent Search; Agent Engine = Agent Runtime.

EXAM TRAP — WHERE MODEL ARMOR APPLIES

Console Model Armor protects the Gemini Enterprise assistant only. Custom ADK/A2A agents need guardrails in code (or Agent Gateway); CX Agent Studio uses callbacks and its own safety settings.

EXAM TRAP — A2A REGISTRATION BYPASSES AGENT GATEWAY

If a scenario requires central gateway policy on every agent call, A2A agents registered directly in Gemini Enterprise do not satisfy it by default.

EXAM TRAP — CONNECTOR WRITE TOGGLE & CUSTOM MCP TRAPS

Disabling *"Search For Data"* on a connector breaks *"Add Or Update Data"* write actions (`ENTITY_NOT_FOUND`). Custom MCP server stores require **`StreamableHTTP`** (legacy `SSE` is rejected), cap at **100 enabled actions** per MCP store (blended apps cap at **50 data stores**), and require **`readOnlyHint: true`** (not just `destructiveHint: false`) to suppress user confirmation prompts on reads.

EXAM TRAP — MULTIMODAL & PDF INGESTION BOUNDARIES

Unstructured stores reject raw `.mp3` / `.mp4` , and Media Data Stores index catalog/clickstream metadata rather than spoken dialogue (use Speech-to-Text V2 **`Chirp 3`** **`BatchRecognize`** → **Unstructured Store** for Workforce-ACL-aware spoken search, or **Agent Retrieval +** **`gemini-embedding-2`** with explicit query-time ACL filters). Document AI Layout Parser caps at **15 pages sync** (**30** `imageless_mode` , 40 MB) vs **500 pages / 1 GB batch** (`ImportRagFiles` **20 MB / 500p**; Custom Extractor **15p sync / 200p batch**, 10p with checkboxes); corpus-wide SQL math across thousands of forms requires the **dual-sink pattern** (Custom Extractor → BigQuery alongside Layout Parser → Agent Search / RAG Engine).

Architect checklist

- Edition supports the requirement (no Business edition for IAM, VPC-SC, CMEK, custom agents).
- Data & parser mode chosen deliberately: ingestion for quality, federation for no-copy; full syncs scheduled for deletions and ACL changes; dual-sink (Custom Extractor → BigQuery + Layout Parser → Agent Search) wired when corpus-wide SQL aggregations are needed; Speech-to-Text V2 `Chirp 3` `BatchRecognize` used for multi-hour ACL-aware audio archives.
- Identity provider configured (Google Identity or global Workforce Identity Federation pool) and ACLs verified (including explicit principal/group filters when using Agent Retrieval + `gemini-embedding-2`).
- Builder path chosen with the 4-way decision matrix; escalate to ADK whenever lifecycle callbacks, `Live API` barge-in, dynamic `@node` graphs, or custom state/sandbox/ `CustomMetric` backends are required.
- Custom MCP connectors use `StreamableHTTP` , curate `⩽ 100` enabled actions, and set `readOnlyHint: true` on non-mutating tools; blended apps stay within `⩽ 50` data stores.
- OAuth on behalf of the user configured for third-party actions; least-privilege roles for service agents.
- Guardrails: Model Armor templates for the assistant, in-code/gateway guardrails for custom agents, callbacks and flows for deterministic CX steps.
- Human-in-the-loop: `Request info` and `Approval` (`Needs Review` in **Runs**) wired before Workflow Builder writes; `end_session` handoff wired on every CX agent that can escalate.
- Evaluation: CX golden and scenario test cases, Workflow Builder **Preview** + **Runs** logs and versions, `adk eval` / `agents-cli eval` / Gen AI Evaluation Service for ADK.

11. Glossary

- **Gemini Enterprise:** Google's employee-facing search, assistant and agent platform (formerly Agentspace).
- **Workflow Builder:** GA no-code builder for employee chat agents and multi-step workflows in Gemini Enterprise (formerly Agent Designer), offering a Conversational Designer and visual Editor Canvas.
- **Agent Studio:** low-code prompt, system instruction, and agent prototyping console inside Agent Platform that exports starter ADK Python code and supports **Deploy as A2A** (`Preview`) on Agent Runtime (distinct from Workflow Builder and CX Agent Studio).
- **Needs Review (** **`Request info`** **&** **`Approval`** **):** paused execution status in Workflow Builder's **Runs** tab triggered by native HITL steps collecting structured human input (`Request info`) or gating mutative actions (`Approval`).
- **Deploy as A2A:** preview option in **Agent Studio** (*Deploy to an Agent Runtime instance*) that exposes a prototype agent in Agent Registry as an A2A endpoint (distinct from **Workflow Builder**, which publishes internally to the Gemini Enterprise Agent Gallery and consumes existing A2A/ADK agents as steps).
- **`readOnlyHint`** : MCP tool annotation (`readOnlyHint: true`) required by Gemini Enterprise to execute read-only custom MCP actions without user confirmation prompts (`destructiveHint: false` alone is insufficient).

- **Agent Gallery:** catalog of Google-made, organization, personal and Marketplace agents inside Gemini Enterprise.
- **Federated search:** live query to a third-party source without copying data.
- **Data ingestion:** copying and indexing source content and ACLs into a data store for higher quality.
- **Layout parser:** default document parser with content-aware chunking and image and table annotation (Document AI Layout Parser supports 15p / 30p `imageless_mode` sync at 40 MB, 500p / 1 GB batch, and 20 MB / 500p in `ImportRagFiles`).
- **Dual-sink document pattern:** pairing Document AI Custom Extractor → BigQuery (or BigQuery `ObjectRef` + `AI.GENERATE`) for corpus-wide SQL aggregations alongside Layout Parser → Agent Search / RAG Engine for narrative chunk retrieval.
- **Chirp 3 (`BatchRecognize`):** Cloud Speech-to-Text V2 model and async GCS batch API used to transcribe multi-hour recordings with speaker diarization and `start_offset` / `end_offset` metadata before indexing into an ACL-aware unstructured data store.
- **Agentic Video (`processing="agentic"` / `media Processing="AGENTIC"`):** GA Gemini direct multimodal mode for multi-turn visual inspection that requires preserving the opaque `step_list` state across stateless turns.
- **Agent Runtime:** managed runtime for code agents (formerly Agent Engine; API resource `ReasoningEngine`).
- **A2A agent card:** JSON description of an A2A agent's endpoint and skills used for registration.
- **Model Armor:** prompt and response screening service; console setting covers the assistant only.
- **CX Agent Studio:** minimal-code builder for customer-facing agents, built on ADK, evolution of Dialogflow CX.
- **Global instructions:** CX Agent Studio instructions inherited by every agent in an application.
- **Golden test case:** CX evaluation case describing the ideal path, including tool calls, for regression testing.
- **end_session:** CX Agent Studio system tool for ending a session, transferring calls, and live-agent handoff.
- **Entry fulfillment:** messages or a webhook that run when a Dialogflow CX page becomes active.
- **Symbolic transition target:** a reserved destination such as `END_FLOW`, `PREVIOUS_PAGE` or `END_SESSION` that moves the session without naming a specific page.
- **Flow stack:** the stack of flows entered through flow-to-flow transitions, capped at 25, that `END_FLOW` pops to return to the caller.
- **Generator:** an LLM call configured inside a Dialogflow CX fulfillment with a prompt template, used without a webhook.
- **Task playbook vs. Routine playbook:** a task playbook is a reusable subtask with input and output parameters; a routine playbook is a sequential stage that can hand off to other routines or flows.
- **OCR parser:** parser for scanned or image-based PDFs and TIFFs (15p/30p sync, 500p/1 GB batch in Document AI; flattens complex multi-column tables and can merge embedded text with `useNative Text`).
- **Digital parser:** basic parser that extracts machine-readable text without detecting tables, lists or headings; fallback for file types the chosen parser does not support.
- **Layout-aware chunking:** chunking that keeps each chunk's text within one layout entity (100–500 tokens in Search stores; `500-1024` tokens with `100-256` overlap in Document AI / RAG Engine).
- **Ancestor headings:** optional setting (`includeAncestorHeadings` / `include_ancestor_headings = true`) that prepends the document title and H1/H2/H3 headings to each chunk so split table rows retain context.
- **Playbook example:** a sample conversation, including tool calls, used as a few-shot prompt for a Dialogflow CX playbook.
- **Example Store:** Agent Platform service (preview) that stores few-shot examples and retrieves the relevant ones for an agent at request time.
- **Agent Retrieval:** Agent Platform storage and vector search engine (formerly Vector Search 2.0) built on Collections of JSON Data Objects, with optional autogenerated embeddings (requires query-time principal/group filters for per-user ACLs).
- **`gemini-embedding-2`:** multimodal embedding model mapping text, images, documents, audio and video into one 3,072-dimension space, with an 8,192-token input limit per item.
- **`audio_timestamp`:** generation config flag (`audioTimestamp` in REST) required for accurate timestamps on audio-only files.
- **Media data store:** Agent Search structured data store with a media schema plus required user events, used for media search and recommendations; it does not analyze the media files themselves.