MiaRec

Conversation Analytics - Platform Setup & Operations Guide

MiaRec, Inc.

Table of contents

1. Overview

This guide explains how to **set up and operate MiaRec Conversation Analytics** in a **multi-tenant** deployment.

It is written for **platform operators** (service providers / partners) who manage:

- tenant lifecycle (create/configure/deprovision tenants)
- system-wide ingestion and transcription configuration
- AI engines (LLM providers/models)
- global **Custom Fields** and **AI Tasks**
- the **AI Assistant job** (continuous processing pipeline)
- monitoring, troubleshooting, upgrades, and governance

If you are an **organization (tenant) administrator**, use the *Conversation Analytics – Administration Guide* instead.
If you are an end user (supervisor/analyst/agent), use the *Conversation Analytics – User Guide*.

1.1 What Conversation Analytics does (operator view)

At a high level, the platform:

1. Ingests conversations (voice calls today; omni-channel text sources such as chat/email/tickets as you enable them).
2. Produces a **Transcript** (for voice calls) or a normalized **Text Thread** (for text channels).
3. Runs tenant-enabled **AI Tasks** (LLM prompts + output mapping + filters) via the **AI Assistant job**.
4. Stores outputs into **Custom Fields**, which power dashboards, search filters, and conversation details.

> **Key operator concept:** AI Tasks are executed **per-tenant** based on tenant activation and filters. You manage the **global defaults** and the underlying processing pipeline.

1.2 How this guide is organized

- **Deployment Models and Responsibilities** – SaaS vs Partner-hosted, and who owns which settings.
- **Platform Setup** – prerequisites, tenant lifecycle, ingestion, transcription, AI engines, global tasks/fields, AI Assistant job.
- **Operations** – monitoring, cost controls, security & governance, upgrades.
- **Troubleshooting** – common problems and runbooks.
- **Reference** – naming standards, prompt/schema standards, default filter patterns.

1.3 Fast-start checklist (operator)

Use this as a sanity path when bringing up a new environment:

1. ✅ Validate infrastructure prerequisites (databases, storage, queues/workers, secrets).
2. ✅ Configure **channel ingestion** (voice and/or text sources).
3. ✅ Configure **transcription** for voice calls and verify transcripts are produced.
4. ✅ Configure one or more **AI Engines** (LLM providers/models).
5. ✅ Create/publish baseline **global Custom Fields** (CX + Sales + Ops).
6. ✅ Create/publish baseline **global AI Tasks** (CSAT, summarization, sentiment, etc.), default-disabled.
7. ✅ Configure and start the **AI Assistant job**.
8. ✅ Create a test tenant → enable a couple AI Tasks → run a smoke test.
9. ✅ Set up monitoring, alerting, and a change-management process.

1.4 Definitions used in this guide

- **Platform operator / system admin:** manages global settings and multiple tenants.
- **Tenant admin:** administers one tenant (enables tasks, overrides prompt/filters, dashboards).
- **Custom Field:** typed storage for an insight (number/text/date/dropdown).
- **AI Task:** an analysis definition (prompt + mapping + optional filters + AI engine selection).

- **AI Assistant job:** continuous worker/job that runs enabled tasks and writes results.

2. Deployment Models and Responsibilities

MiaRec can be deployed in different operational models. The purpose of this chapter is to make **responsibility boundaries explicit**, so operators and tenant admins know where to look when they need to change something.

2.1 Deployment models

SaaS (MiaRec-operated)

- MiaRec operates the **platform (system/global)** configuration and manages tenants.
- Customer admins operate only **tenant-level** configuration (enable tasks, override prompt/filters, dashboards, permissions).
- Customers are not exposed to multi-tenant concerns.

Partner-hosted / On-prem (partner-operated)

- A partner operates the platform and is responsible for:
- system/global settings
- tenant creation and lifecycle
- platform monitoring and upgrades
- Partner customers operate tenant-level configuration, similar to SaaS customers.

2.2 Responsibility matrix (recommended)

Area	SaaS (MiaRec-operated)	Partner-hosted / On-prem
Create/manage tenants	MiaRec	Partner (platform operator)
Channel ingestion configuration	MiaRec (or shared)	Partner
Transcription engine configuration	MiaRec	Partner
AI engines (LLM provider/model)	MiaRec	Partner
Global Custom Fields library	MiaRec	Partner
Global AI Tasks library	MiaRec	Partner
AI Assistant job configuration & scaling	MiaRec	Partner
Tenant activation of AI Tasks	Customer tenant admin	Customer tenant admin
Tenant overrides (Prompt/Filters)	Customer tenant admin	Customer tenant admin
Dashboards/search configuration	Customer tenant admin	Customer tenant admin
Day-to-day usage	Customer users	Customer users

> **Tip:** In partner-hosted deployments, you typically want a clear separation between **platform operator roles** and **tenant admin roles** to avoid accidental cross-tenant changes.

2.3 What belongs in which guide

- **Platform Setup & Operations Guide (this guide):** everything that affects multiple tenants or the platform as a whole.
- **Administration Guide:** tenant-only configuration (enabling tasks, overrides, dashboards, permissions).
- **User Guide:** day-to-day usage, interpretation, workflows.
- **Overview & Key Concepts:** customer-safe definitions and value overview.

2.4 Role terminology

Throughout MiaRec documentation:

- **Platform operator / System admin** – manages global settings, multiple tenants, AI engines, and global tasks
- **Tenant admin / Organization admin** – manages settings within a single tenant (enables tasks, configures overrides, manages dashboards)

> **Note for SaaS customers:** Some platform-level settings described in the Platform Setup & Operations Guide are managed by MiaRec. If you don't see a setting described in this guide, it may be managed on your behalf. Contact MiaRec support for assistance.

3. Platform Setup

3.1 Prerequisites and Architecture

This chapter helps platform operators understand **what needs to exist** (infrastructure and integrations) before enabling Conversation Analytics in a partner-hosted / multi-tenant deployment.

It also provides an **operator-level architecture view** so you know which subsystem to check when something breaks.

Reference architecture (operator view)

At a conceptual level:

Core platform components (what to identify in your environment)

Data ingestion (per channel)

- Voice call ingestion (audio and metadata)
- Chat/email/ticket ingestion (message thread + metadata)

- Normalization/mapping layer (IDs, timestamps, participants, channel types)

Transcription (voice channels)

- Transcription engine(s)
- Language strategy (auto-detect vs configured languages)
- Transcript storage and indexing

AI analysis

- **AI engines**: LLM providers/models and their credentials/config
- **AI Tasks**: global task library + tenant activation + overrides
- **AI Assistant job**: continuous pipeline executing tasks and persisting results

Data storage and retrieval

- Operational database (tenants, tasks, fields, mapping, status)
- Search index (conversations, transcripts/threads, custom fields)
- Analytics storage (for dashboards/aggregations), if separate from operational DB

Observability and governance

- Logs/metrics/traces for ingestion, transcription, jobs, task executions
- Audit logs for configuration changes (fields/tasks/overrides)
- Usage accounting (LLM spend, requests, tokens) per tenant (recommended)

Environment prerequisites (operator checklist)

Networking & security

- Outbound connectivity to:
- transcription provider(s) (if external)
- LLM provider(s) (if external)
- TLS certificates and secure endpoints for ingestion
- Secrets management for provider credentials

Compute and scaling

- Worker capacity for:
- ingestion processing
- transcription throughput
- AI Assistant job concurrency
- Backpressure mechanisms (queues) to handle spikes and retries

Storage & retention

- Storage for raw audio (voice) and text threads (omni-channel)
- Retention policy per tenant (recommended to support configurable retention)
- Backup and restore strategy

Operational readiness checklist

Before enabling customer tenants, verify:

- ✅ You can ingest and store conversations for at least one test tenant
- ✅ You can produce transcripts (voice) or normalized threads (text)
- ✅ You can run the AI Assistant job and persist Custom Field outputs
- ✅ You can view outputs in conversation details, dashboards, and search
- ✅ You have monitoring + alerting for each pipeline stage

Implementation notes

MiaRec uses a queue-based worker model for processing. Key tuning knobs include:

- Worker concurrency settings
- Retry policies and dead-letter queues
- Job schedule and data source configuration

Storage and indexing are handled by the MiaRec platform. Retention controls can be configured per tenant.

Contact MiaRec for detailed capacity planning guidance based on your expected workload (calls/day, average transcript length, tasks enabled).

3.2 Tenant Lifecycle Management

This chapter describes how a platform operator should **create, configure, and maintain tenants** in a multi-tenant deployment.

The goals are: - consistent onboarding (every tenant starts with a known baseline), - safe isolation, - predictable upgrades and change management.

Tenant lifecycle stages (recommended)

1. **Provision**
2. Create the tenant container (organization record, identifiers, domain settings).
3. **Configure baseline**
4. Enable channels/integration endpoints as applicable.
5. Configure transcription defaults (voice).
6. Assign default permissions/roles.
7. Attach baseline analytics configuration (fields + tasks).
8. **Validate**
9. Run ingestion + transcription + AI task smoke tests.
10. **Go live**
11. Enable production ingestion; monitor health and usage.
12. **Operate**
13. Support changes, add insights, handle overrides and governance.
14. **Deprovision**
15. Disable ingestion, stop processing, export if needed, apply retention and deletion.

Baseline tenant bundle (recommended)

A “baseline bundle” reduces time-to-value and support burden. Consider standardizing:

Baseline Custom Fields (examples)

- CX: CSAT, sentiment, top issues, escalation reason
- Conversation understanding: summary, topic, call reason, outcome
- Sales: lead score, objections, competitors, next actions
- Operational: language, channel type, disposition (if applicable)

Baseline AI Tasks (examples)

- Conversation summarization (with explanation)
- Sentiment classification (with explanation)
- CSAT scoring (value + explanation)
- Call reason/outcome detection (value + explanation)

> **Operator note:** Prebuilt tasks can be published globally and default-disabled. Tenant admins can enable them when ready.

Tenant isolation and data boundaries

Document how the platform enforces isolation for: - conversation content (audio/transcripts/threads), - analytics outputs (custom fields, dashboards), - configuration (tenant tasks, overrides), - usage and billing metrics (if applicable).

Onboarding checklist (operator)

- ✅ Tenant created and admin access granted
- ✅ Ingestion configured (voice and/or text connectors)
- ✅ Transcription validated (voice): transcripts exist for test calls
- ✅ Global baseline AI Tasks available in tenant as Disabled
- ✅ AI Assistant job processing confirmed for tenant
- ✅ Dashboards/search available (if part of baseline)
- ✅ Support and escalation contacts established

Deprovisioning (recommended process)

1. Disable new ingestion
2. Stop ongoing processing for the tenant (pause tasks/job eligibility)
3. Export required data (optional)
4. Apply retention/deletion policy (audio, transcripts, text threads, custom fields)
5. Remove credentials and access (SSO, API keys, connectors)
6. Confirm deletion completion (audit log)

Implementation notes

- Global tasks appear in tenant's **Disabled** tab until the tenant admin enables them
- Tasks with tenant-level overrides display an "Overridden settings" tag on the task card
- Provide a "tenant template" mechanism with baseline fields/tasks (default disabled)
- Track overrides explicitly and make them auditable (who changed what, when)
- Treat deprovisioning as a first-class runbook with audit evidence

3.3 Channel Ingestion Setup

Conversation Analytics relies on having conversation content (audio or text) plus consistent metadata (participants, timestamps, channel, queue/team, etc.).

This chapter describes how a platform operator should set up and validate ingestion across channels.

Ingestion goals (operator)

- Ingest conversations reliably and securely.
- Normalize metadata so reporting, filtering, and AI Tasks behave consistently.
- Detect ingestion gaps early (monitoring/alerts).

Supported channel types (conceptual)

Voice calls (speech)

Typical inputs: - audio recording(s) - call metadata (direction, start/end time, agent ID, phone numbers, queue/campaign)

Outputs to downstream pipelines: - raw audio + normalized metadata record

Text channels (omni-channel)

Typical inputs: - chat transcript / message thread - email thread (messages + headers) - ticket thread (comments, updates, resolution status) - metadata (agent/customer IDs, timestamps, channel, queue, tags)

Outputs to downstream pipelines: - normalized **text thread** + metadata record

> **Note:** Even in text channels, the “conversation” is treated as a single analyzable unit with a thread of messages/events.

Metadata normalization (recommended)

Define and enforce a standard schema so everything downstream can rely on it.

At minimum, capture:

- **Tenant ID**
- **Conversation ID** (stable)
- **Channel type** (call/chat/email/ticket)
- **Start/end timestamps**
- **Participants**
- agent identifier
- customer identifier (if available)
- **Direction** (inbound/outbound), where applicable
- **Queue/team/campaign** (optional but strongly recommended)
- **Language** (optional; transcription/AI may infer)
- **Tags / dispositions** (optional)

Validation checks (before enabling analytics)

For each channel: - ✅ content is present (audio or text thread) - ✅ metadata fields are populated and consistent - ✅ IDs are stable (no duplicates) - ✅ timestamps are correct (time zone handling) - ✅ sample conversations appear in the UI for the intended tenant

---

Operational best practices

- **Backfill support:** Ensure you can ingest historical conversations for onboarding and reprocessing.
- **Deduplication:** Protect against duplicate ingestion events.
- **PII handling:** Decide whether redaction happens pre-ingestion or inside MiaRec pipelines (document the chosen approach).
- **Error handling:** Use retries and a dead-letter strategy for poison messages.
- **Observability:** Monitor ingestion throughput, latency, and error rates per tenant and channel.

Implementation notes

MiaRec supports various voice ingestion methods including:

- SIPREC / PBX recordings
- CCaaS integrations (Genesys, NICE, Five9, Amazon Connect, etc.)
- Upload via API

For text channels (chat, email, tickets), support varies by deployment. Contact MiaRec for details on specific connector availability.

- Document a minimum required metadata schema and a "recommended schema"
- Strongly recommend stable conversation IDs and tenant IDs
- Provide sample payloads (voice + text) and validation steps

3.4 Transcription System Setup

Transcription is a **required foundation** for AI insights on **voice calls**. If a call has no transcript (or a very low-quality transcript), AI Tasks cannot reliably produce insights.

This chapter covers transcription from a **platform operator** perspective: configuring transcription engines and the transcription pipeline for a multi-tenant environment.

What transcription provides (operator view)

- Converts audio to text (**transcripts**) per call
- Stores transcripts so they can be:
- viewed by users
- searched
- analyzed by AI Tasks (CSAT, topics, summaries, etc.)

Typical transcription pipeline stages

1. Audio ingestion (recordings + metadata)
2. Transcription job creation (queueing)
3. Speech-to-text execution (provider/model)
4. Transcript post-processing (punctuation, diarization, redaction as applicable)
5. Transcript persistence and indexing
6. Availability in UI and downstream analytics

Engine selection and language strategy

Engine selection (examples)

Operators may configure one or more transcription engines/providers. Consider documenting: - which providers are supported - how to choose engines per tenant, per language, or per region (if applicable)

Language strategy

Common approaches: - **Auto-detect** language (simplest, may be less accurate for short calls) - **Tenant-configured languages** (more accurate, requires setup) - **Per-conversation language hint** (from ingestion metadata)

Validation / smoke test (operator)

For a test tenant: 1. Ingest a short call (30–60 seconds) with known audio clarity. 2. Confirm transcript is produced within expected latency. 3. Verify: - transcript is visible in call details - speaker separation (if supported) is reasonable - timestamps align with the audio 4. Confirm the transcript is available to AI Tasks (see AI Assistant job smoke test).

Monitoring and alerting (transcription)

Track, at minimum: - transcription backlog/lag - job failure rate - provider timeouts/quotas - average transcription latency - percent of calls with missing/empty transcripts - language distribution and “unknown language” rate

---

Operational best practices

- Provide a reprocessing mechanism (retranscribe) for:
- provider incidents
- improved models
- bug fixes in diarization/punctuation
- Define tenant-level retention for audio and transcripts (aligned with compliance requirements)
- If using external providers, document:
- credential rotation
- regional/data residency constraints
- rate limits and quotas

Implementation notes

- Transcription is typically configured via `Administration > Speech Analytics > Transcription`
- Multiple transcription engines may be available with per-tenant or per-language selection
- Features like speaker diarization and punctuation depend on the transcription engine
- Document a baseline "supported audio quality" guideline and minimum call duration thresholds
- Implement and document a "transcription health dashboard" for operators
- Provide a "reprocess transcription" runbook and clearly describe expected impact (cost/time)

3.5 AI Engines (LLM Providers/Models)

AI Engines define which Large Language Model (LLM) provider/model MiaRec uses to run AI Tasks (summaries, sentiment, CSAT, Auto QA, and custom insights).

This chapter describes how platform operators should configure and govern AI Engines in a multi-tenant environment.

What an AI Engine is

An **AI Engine** is a named configuration that typically includes: - provider (e.g., OpenAI/Azure OpenAI/Anthropic/etc.) - model name/version - authentication (API key, endpoint, deployment name) - safety and policy settings (if applicable) - limits (rate limits, quotas) - default parameters (timeouts, max tokens, temperature) if the platform supports them

AI Tasks reference an AI Engine when executing.

Recommended engine strategy

Start with a small set of standard engines

For example: - **Standard** – balanced cost/quality - **High accuracy** – for complex scoring/ categorization - **Low latency** – for near-real-time needs (if applicable)

Keep the catalog small to reduce operational complexity.

Decide how engine selection works

Common patterns: - **Single global default** engine for all tasks - **Per-task engine selection** (your UI appears to support this) - **Per-tenant engine policy** (if required for cost or compliance)

Credential and security management

- Store credentials in a secrets manager (recommended).
- Rotate credentials regularly and document rotation steps.
- Limit engine access by environment (dev/stage/prod).

- Maintain audit logs for engine configuration changes.

Validation / smoke tests

When adding an engine: 1. Run a minimal test prompt (health check) to verify credentials and connectivity. 2. Run a representative AI Task (e.g., summarization) on a test transcript. 3. Verify: - latency - successful JSON responses (if using JSON output) - stable output quality

---

Operational considerations

Cost management

- Track usage by tenant and by task (requests, tokens, cost).
- Use task filters to avoid running tasks on ineligible conversations.
- Consider “preview mode” for new tasks before broad enablement.

Reliability and failover (if supported)

- Define a fallback engine if the primary provider is down.
- Document expected behavior:
- automatic failover vs manual switch
- per-task vs global failover

Data handling

- Document data residency considerations (especially in partner-hosted deployments).
- Document whether transcripts are sent to external providers and what is included (metadata vs transcript only).

Where to configure

Menu path: `Administration > Speech Analytics > AI Assistant > Engines`

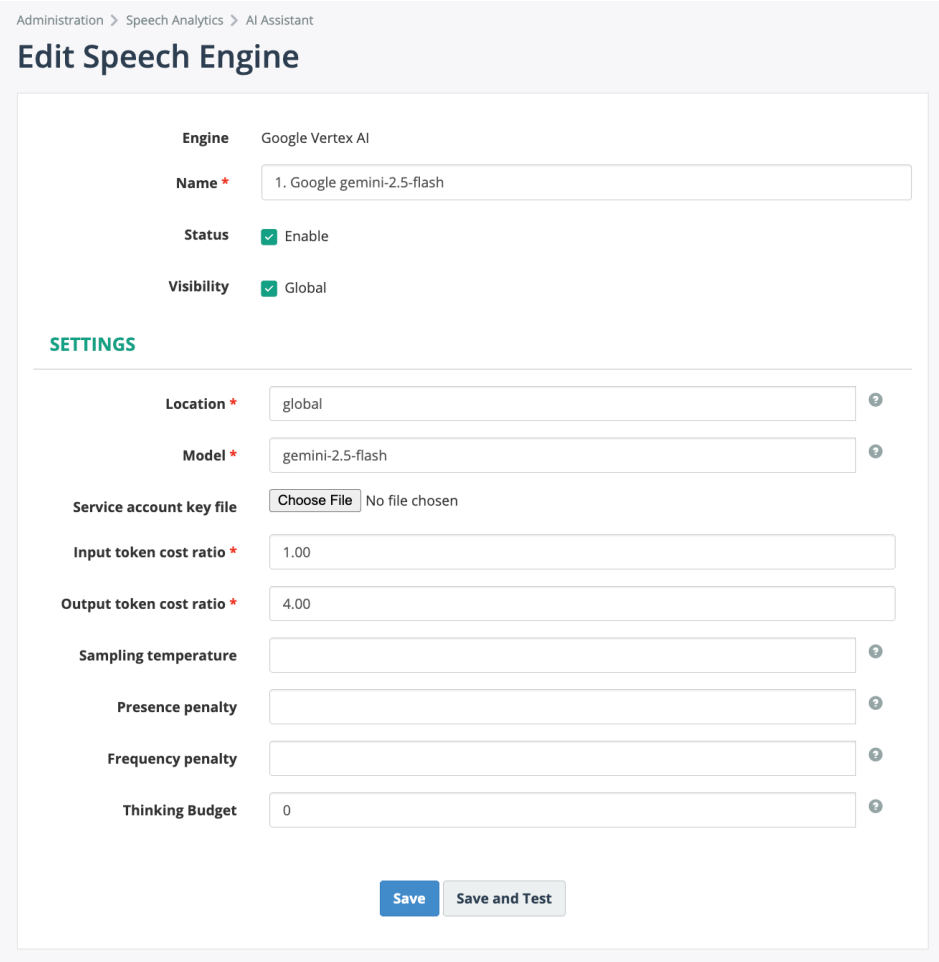

Figure: AI Engine configuration showing name, status, visibility settings, and model configuration options.

Engine settings

When configuring an AI Engine, you specify:

- **Name** – A descriptive name for the engine
- **Status** – Enabled or Disabled
- **Visibility** – Global (available to all tenants) or Tenant-specific
- **Model settings** – Provider-specific configuration (API endpoint, model name, authentication)

Per-task engine selection

Each AI Task can select which engine to use. This allows you to:

- Use different models for different task types (e.g., a faster model for simple tasks, a more capable model for complex analysis)
- Test new models on specific tasks before broader rollout
- Manage costs by using appropriate models for each use case

3.6 Global Custom Fields

Custom Fields are the **data schema** for Conversation Analytics. They determine: - what values can be stored (type and constraints), - how insights appear in conversation details (display groups), - how dashboards and search filters behave.

In a multi-tenant platform, partners typically maintain a **global library** of Custom Fields that can be reused across tenants.

Where to configure

Menu path: `Administration > Customization > Custom Fields`

> This path is confirmed.

Global field strategy (recommended)

Why standardize fields globally?

- Enables consistent dashboards and reporting across tenants.
- Supports reusable global AI Tasks (mapping requires stable fields).
- Reduces tenant-by-tenant configuration burden.

What should be global vs tenant-specific?

Make global when: - the metric is common across many tenants (CSAT, sentiment, topic, reason/outcome) - you ship it as a prebuilt insight - you want consistent naming and dashboard behavior

Make tenant-specific when: - it is unique to a vertical customer (e.g., hospitality reservation fields) - it contains tenant-private taxonomy (custom dropdown values)

Field design guidelines

1) Use stable identifiers

- **Computer name** should be stable and treated like an API identifier.

- Avoid renaming computer names after tasks/dashboards depend on them.

2) Choose the right field type

- **Number (integer/decimal):** scores (CSAT), durations, amounts
- **Date:** reservation start date, follow-up due date
- **Text:** summaries, explanations, free-form extraction
- **Dropdown (single/multi-select):** classifications (topic, reason, outcome)

3) Keep dropdown taxonomies controlled

- Prefer a small, curated set of allowed values.
- Avoid synonyms (e.g., “Billing issue” vs “Billing Issues”).
- Document the meaning of each label.

4) Thresholds for numeric fields

Thresholds are used for: - dashboard buckets (e.g., satisfied vs dissatisfied) - consistent coloring/ labels in UI - one-click drilldowns by bucket

Example: CSAT 1–5 with colored thresholds.

5) Display groups

Use display groups (e.g., “CX Metrics”, “Sales Metrics”) to organize fields in conversation details.

Creating a global Custom Field (operator checklist)

1. Create field with appropriate **type**
2. Configure:
3. **visibility / availability to all tenants** (global)
4. **availability to AI Insights tasks** (so tasks can map to it)
5. Configure display group and ordering
6. Configure thresholds (if numeric)
7. Optionally enable “create dashboard for this metric” (if supported)
8. Validate:

9. the field appears in AI Task mapping selector
10. the field appears in conversation details (if configured)
11. the field is searchable/filterable as expected

Change management (important)

Changing a global field can break: - AI Task mappings - dashboards - saved searches

Recommended change policy: - **Safe changes:** description updates, new dropdown values (if backward compatible), new thresholds - **Breaking changes:** removing dropdown values, changing type, deleting fields, changing computer name

For breaking changes: - create a new field (v2) and migrate tasks gradually - communicate deprecation timelines - provide a migration runbook for tenants

---

Custom Fields list

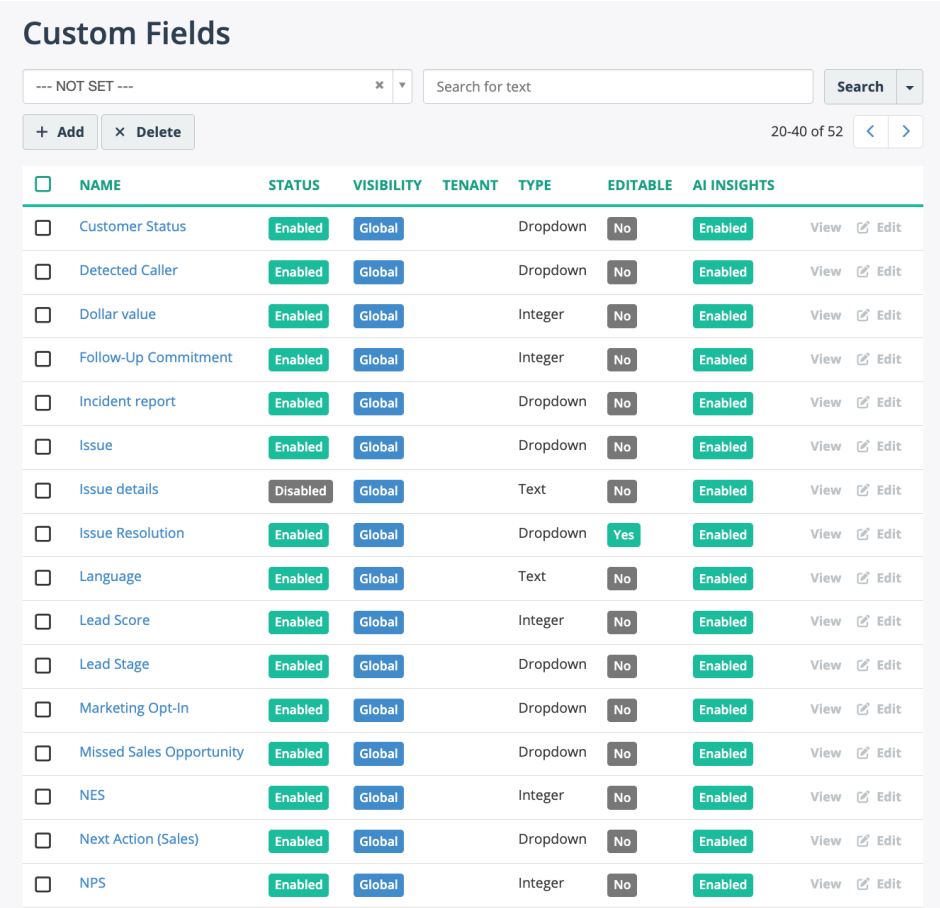

Figure: Custom Fields list showing all configured fields with their types and settings.

Creating a numeric field (example: CSAT)

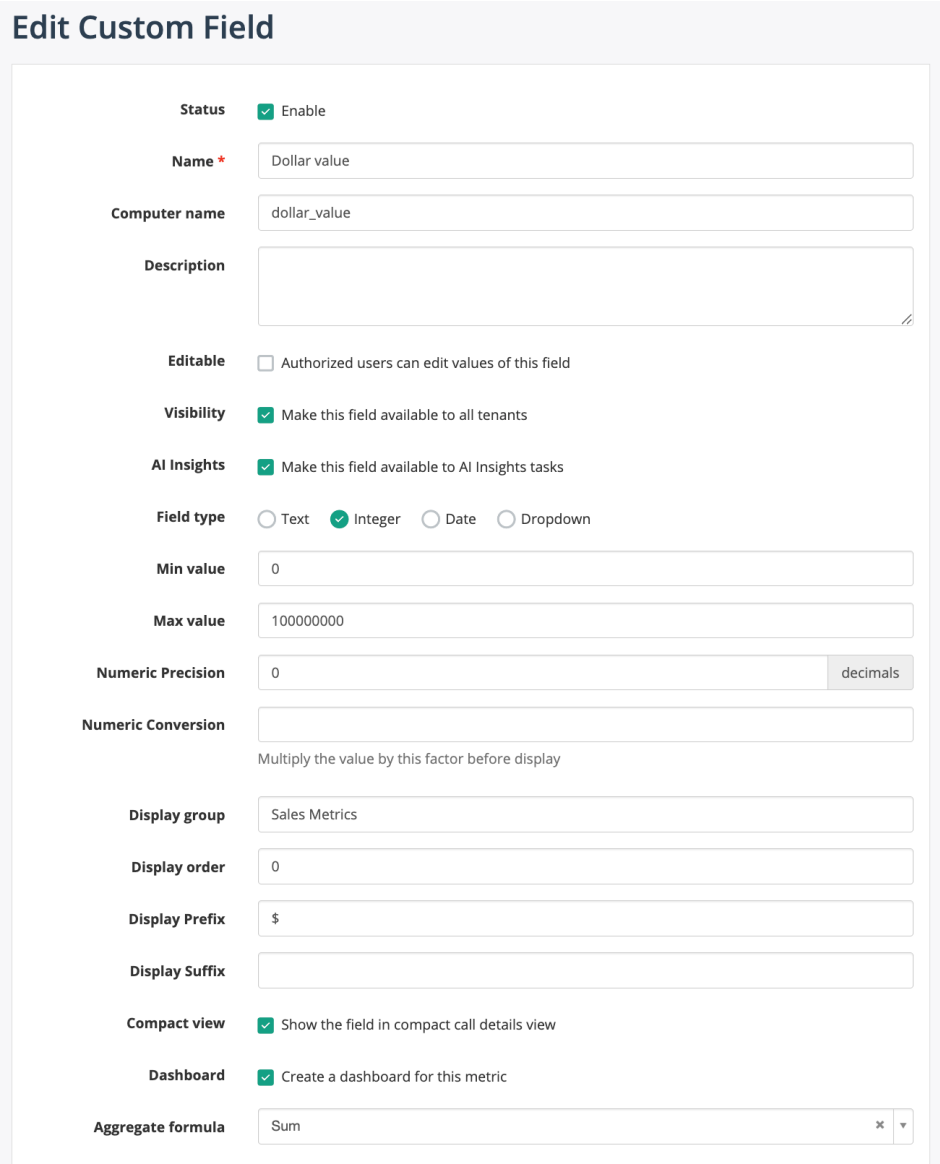

Figure: Custom Field configuration for a numeric field (CSAT). Note the AI Insights checkbox to make the field available to AI Tasks.

Key settings for numeric fields:

- **AI Insights** – Enable this checkbox to make the field available for AI Task mapping
- **Dashboard** – Enable to create a dashboard for this metric
- **Visibility** – Global (all tenants) or Tenant-specific

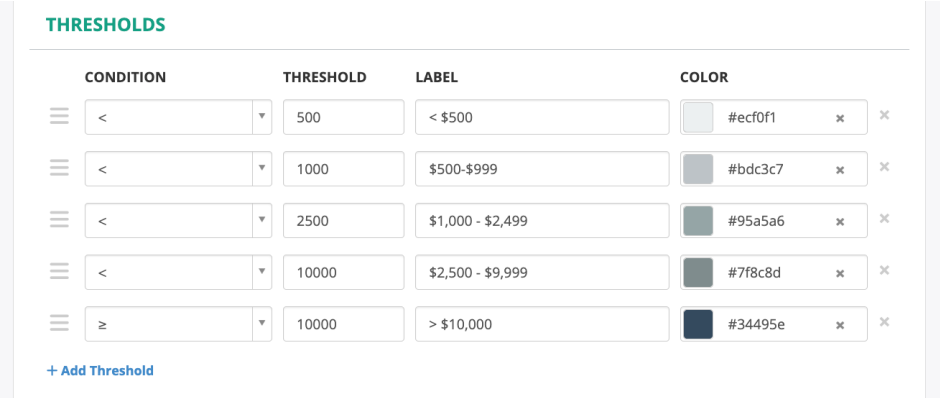

Figure: Threshold configuration for a numeric field. Define buckets with labels and colors for dashboard visualization.

Threshold configuration

For numeric fields, configure thresholds to:

- Define labeled buckets (e.g., "Very Dissatisfied" through "Very Satisfied")
- Set colors for visual distinction
- Enable clickable drilldowns in dashboards

Aggregate formula options

- **Sum** – Total of all values
- **Average** – Mean value
- **Top %** – Top percentage of values
- **Bottom %** – Bottom percentage of values
- **Top % - Bottom %** – Difference between top and bottom percentages

Creating a dropdown field

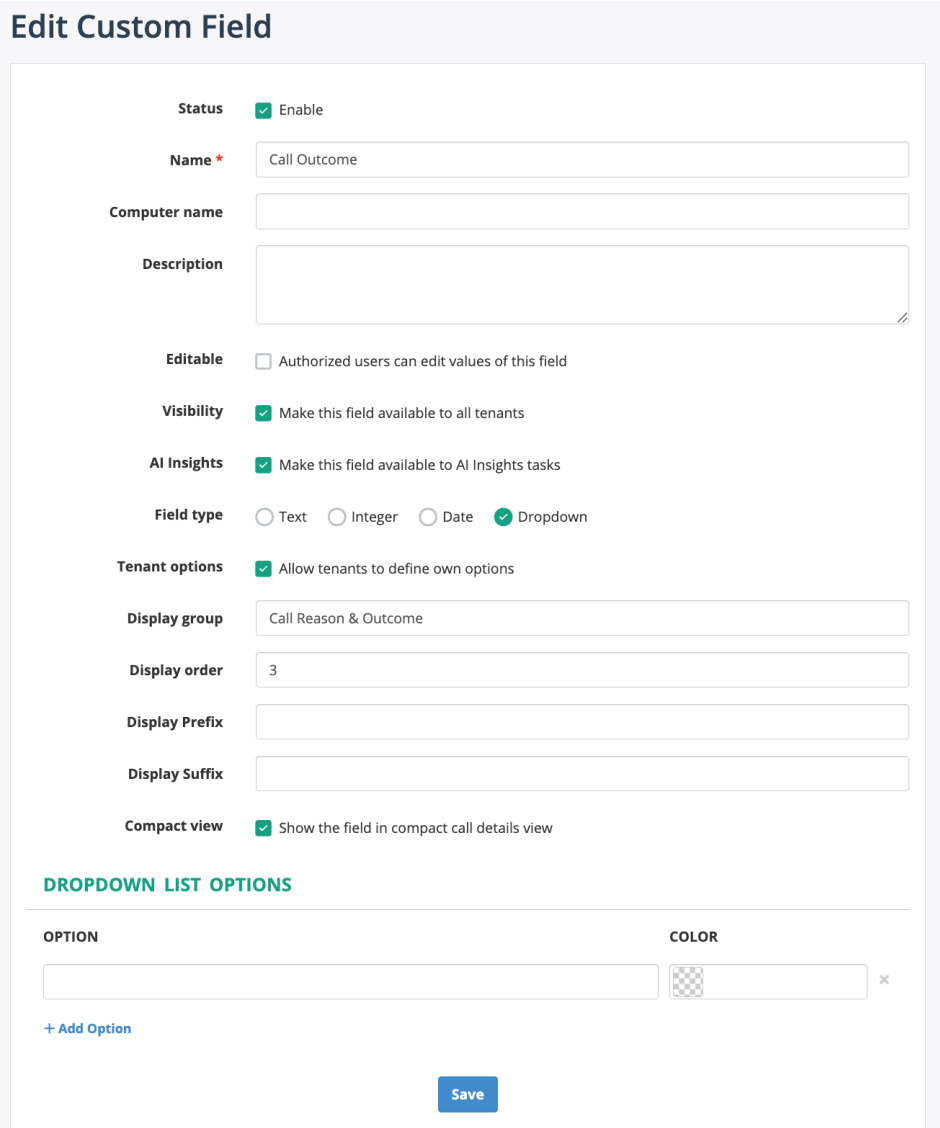

Figure: Custom Field configuration for a dropdown field with predefined options.

For dropdown fields, define allowed values that the AI Task must output. Keep the list controlled to ensure consistent dashboards and reporting.

Tenant-specific dropdown options

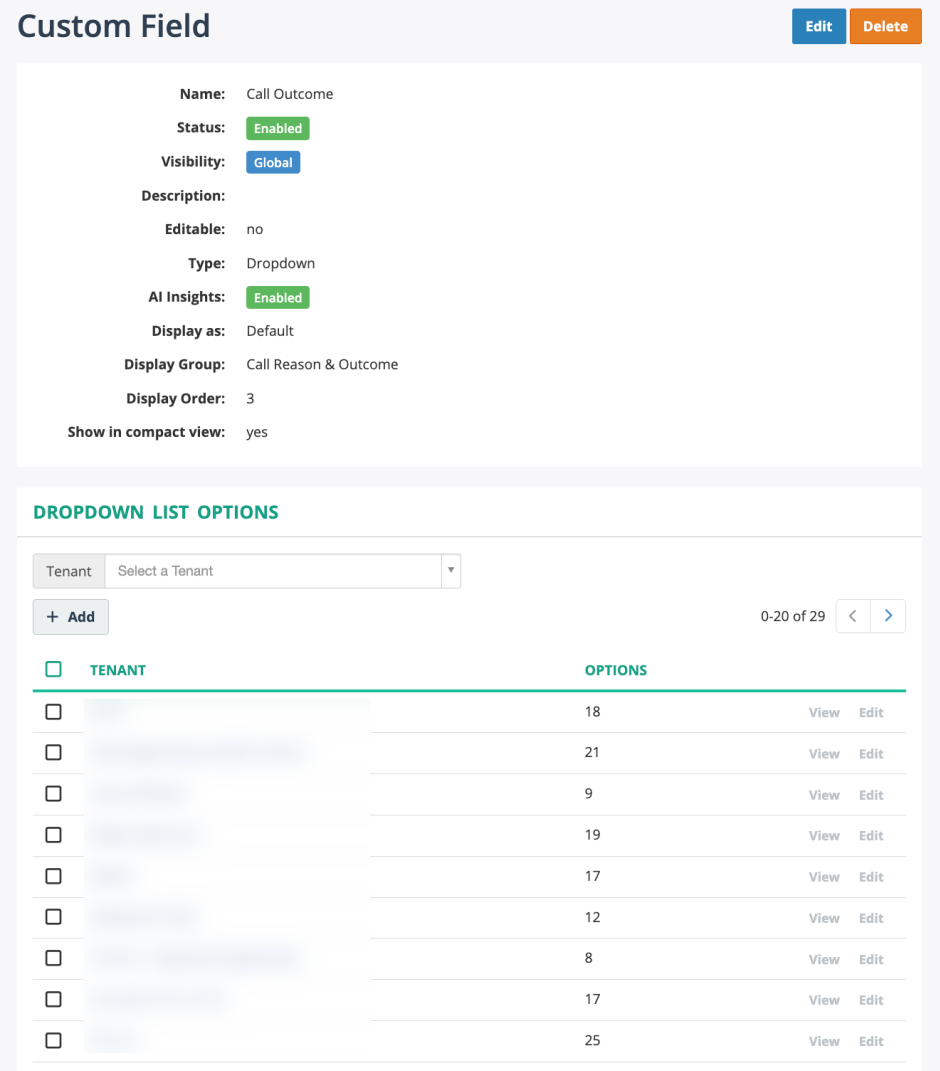

Figure: Tenant-specific dropdown options allow tenants to customize allowed values while maintaining the same field structure.

3.7 Global AI Tasks

Global AI Tasks are reusable, system-managed analysis definitions that can be made available to all tenants. They are the primary mechanism for delivering **prebuilt insights** (CSAT, sentiment, summaries, topics, etc.) at scale.

In MiaRec, an **AI Task** is a bundle that includes: - **Prompt** (instructions + inputs to the LLM) - **Attribute mapping** (output attributes → Custom Fields) - **Filtering criteria** (optional; limits which conversations the task runs on) - **AI engine** selection (LLM provider/model) - Response settings (text/JSON, optional JSON schema validation)

Where to configure

Menu path (global tasks):

`Administration > Speech Analytics > AI Assistant > AI Tasks | Global Tasks`

> This path is confirmed.

Global task lifecycle (recommended)

1. **Design**
2. Define the insight(s) and output schema.
3. Ensure required Custom Fields exist (global fields recommended for prebuilt insights).
4. **Build**
5. Create the global AI Task with prompt, mapping, filters, engine.
6. **Test**
7. Use “Save and Test” (and/or Playground) on real transcripts.
8. **Publish**
9. Mark as available to tenants (global).
10. Default to **Disabled** for tenants unless you have an explicit baseline bundle.
11. **Roll out**
12. Enable for pilot tenants.
13. Monitor results and cost.

14. **Maintain**
15. Version changes safely (see change management below).

Designing a global AI Task (operator standards)

1) Prefer JSON output for structured insights

Use JSON for: - scores (CSAT, NPS) - classifications (reason/outcome/topic) - extraction (amount, date, competitor name)

2) Use a strict output schema

If the UI supports “Response JSON schema”, define required keys and types. This reduces failure modes and makes monitoring easier.

3) Include value + explanation

For most insights, return: - a structured value (score/label/date/text) - a short explanation that cites transcript evidence

This improves trust and helps supervisors and QA teams.

> **Important:** ensure your platform has a clear way to **store or display** the explanation (either a separate field or a dedicated explanation area). Document the correct approach for MiaRec.

4) Keep tasks modular

- One task should do one logical “job”.
- It is acceptable for one task to write multiple fields **only when they are tightly related** (e.g., reason + outcome + explanation).

5) Use filters to control cost and relevance

Examples: - call duration > 15 seconds - inbound only - channel = call (vs chat/email) until text support is enabled

---

Creating a global AI Task (operator checklist)

1. **Create task**

2. Set type (single field / multiple fields / Auto QA where applicable)
3. Name, description, icon
4. Enable status
5. Mark as **Global** (available to all tenants)
6. **Select AI engine**
7. Choose the engine that matches the expected output reliability (JSON compliance matters).
8. **Define outputs**
9. Set Response Type = JSON (recommended)
10. Provide Response JSON schema (recommended)
11. Configure **Attribute mapping**:
    - ATTRIBUTE: output key (e.g., `csat`)
    - CUSTOM FIELD: destination Custom Field (e.g., CSAT)
12. **Write the prompt**
13. Include transcript/thread variable (e.g., `${transcript}` or equivalent)
14. Define scoring definitions or allowed labels
15. Specify “JSON only” output and exact format
16. Include value + explanation pattern
17. **Add filtering criteria** (optional but recommended)
18. duration, direction, channel, queue, etc.
19. **Save and Test**
20. Test with representative examples:
    - positive, neutral, negative
    - edge cases (short calls, transfers, escalations)
21. **Publish and roll out**
22. Keep task **Disabled** by default for tenants unless it is part of a baseline onboarding bundle.

Tenant activation and overrides (how global tasks behave)

In tenant UI, global tasks appear under: `Administration > Speech Analytics > AI Assistant > AI Tasks`
with **Enabled** and **Disabled** tabs.

- Tenant admins enable a task by clicking **Enable** in Disabled.
- Tenant admins can click **Edit** to override:
- Prompt, Filter, or both.

> **Operator governance:** decide whether partners allow overrides by default and how to support override troubleshooting.

Change management for global tasks (critical)

Global task changes can affect: - comparability of metrics over time - downstream dashboards and thresholds - tenant overrides (tenants may diverge from defaults)

Recommended practices: - **Version prompts** (clone task or version within task if supported) - Avoid breaking output schemas for existing mappings - When making major scoring logic changes: - communicate to tenant admins - provide a calibration plan - consider running old and new tasks in parallel for a period

---

Task types

When creating a new AI Task, select from these task types:

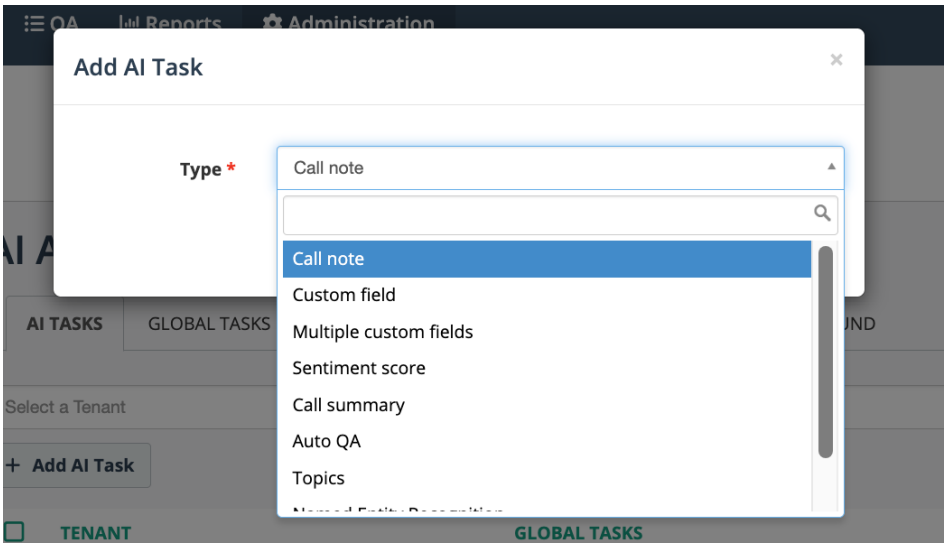

Figure: Select a task type when creating a new AI Task.

Available task types:

- **Call note** – Generate a note or summary for the call
- **Custom field** – Populate a single custom field
- **Multiple custom fields** – Populate multiple related custom fields
- **Sentiment score** – Calculate sentiment from the conversation
- **Call summary** – Generate a structured call summary
- **Auto QA** – Automated quality assurance scoring
- **Topics** – Multi-label topic classification
- **Named Entity Recognition** – Extract specific entities (dates, amounts, names)
- **Speaker Label** – Identify and label speakers

Task editor structure

The task editor includes these sections:

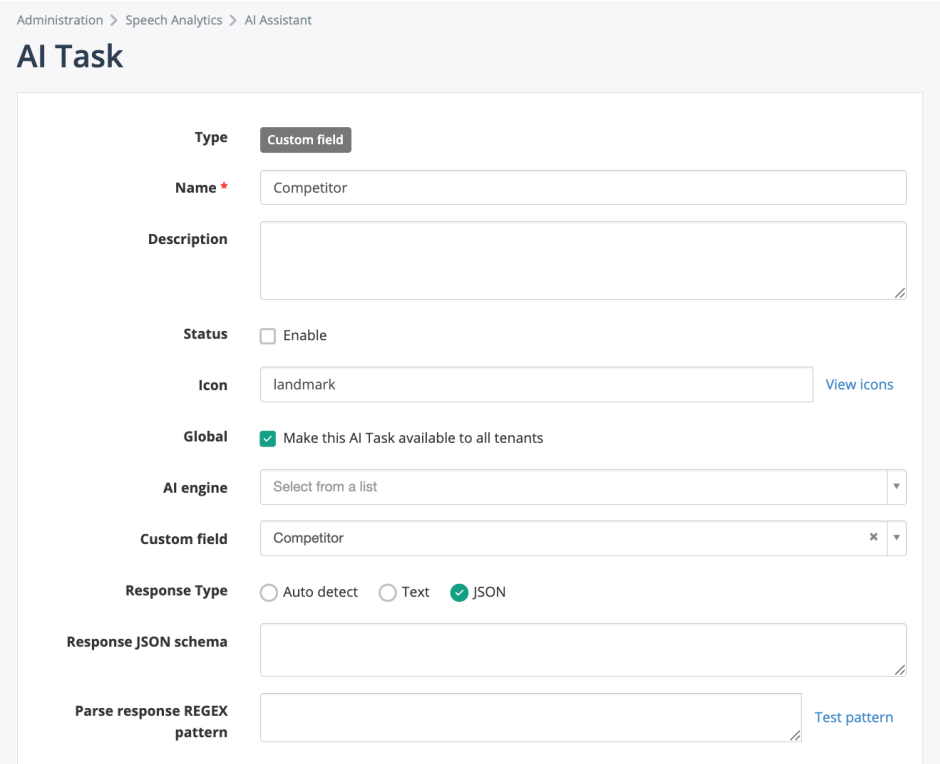

Figure: AI Task general settings and visibility configuration.

Prompt configuration

The prompt is split into two parts:

- **Task instructions** – High-level role and rules for the AI
- **Task inputs** – Detailed instructions including the transcript and output format

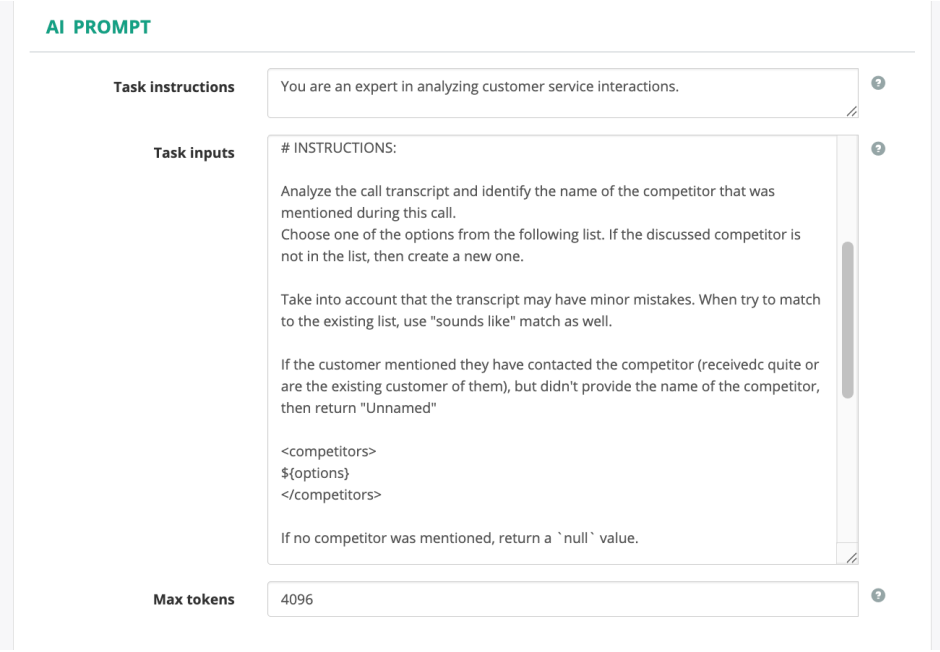

Figure: Task instructions and inputs configuration.

Available variables

Variables you can use in prompts:

- `${transcript}` – The conversation transcript
- `${direction}` – Call direction (inbound/outbound)
- `${duration}` – Call duration
- `${caller-name}` – Caller's name
- `${called-name}` – Called party's name
- `${options}` – Dropdown field options (for classification tasks)

Response configuration

- **Response type** – Auto detect, Text, or JSON
- **JSON schema** – Optional schema for validating JSON responses

---

Viewing tenant activation status

Track which tenants have enabled a global task:

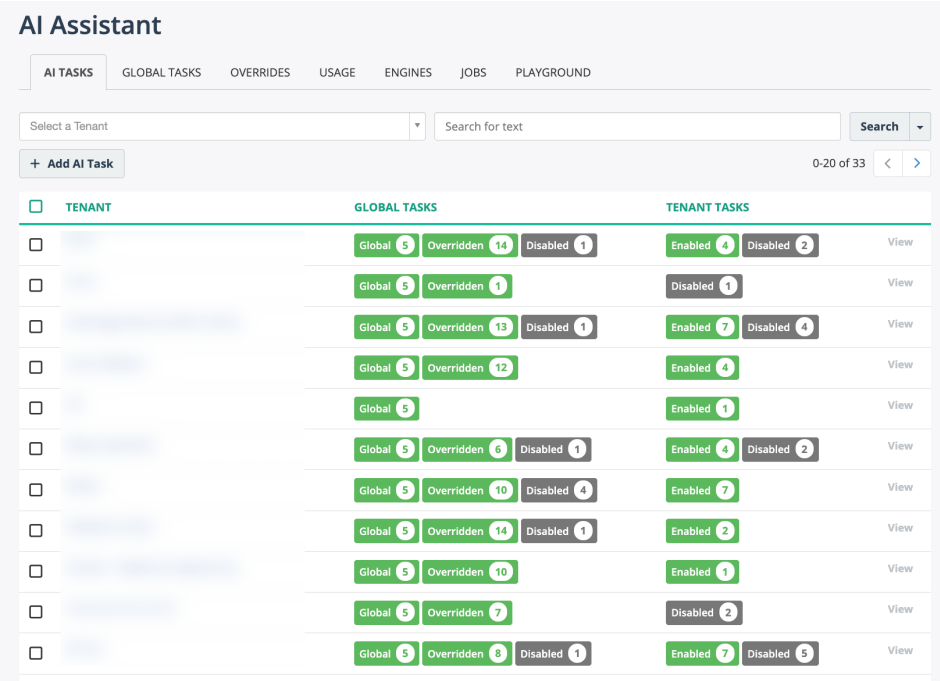

Figure: View tenant activation status for a global AI Task.

Viewing tenant overrides

Monitor which tenants have overridden the default settings:

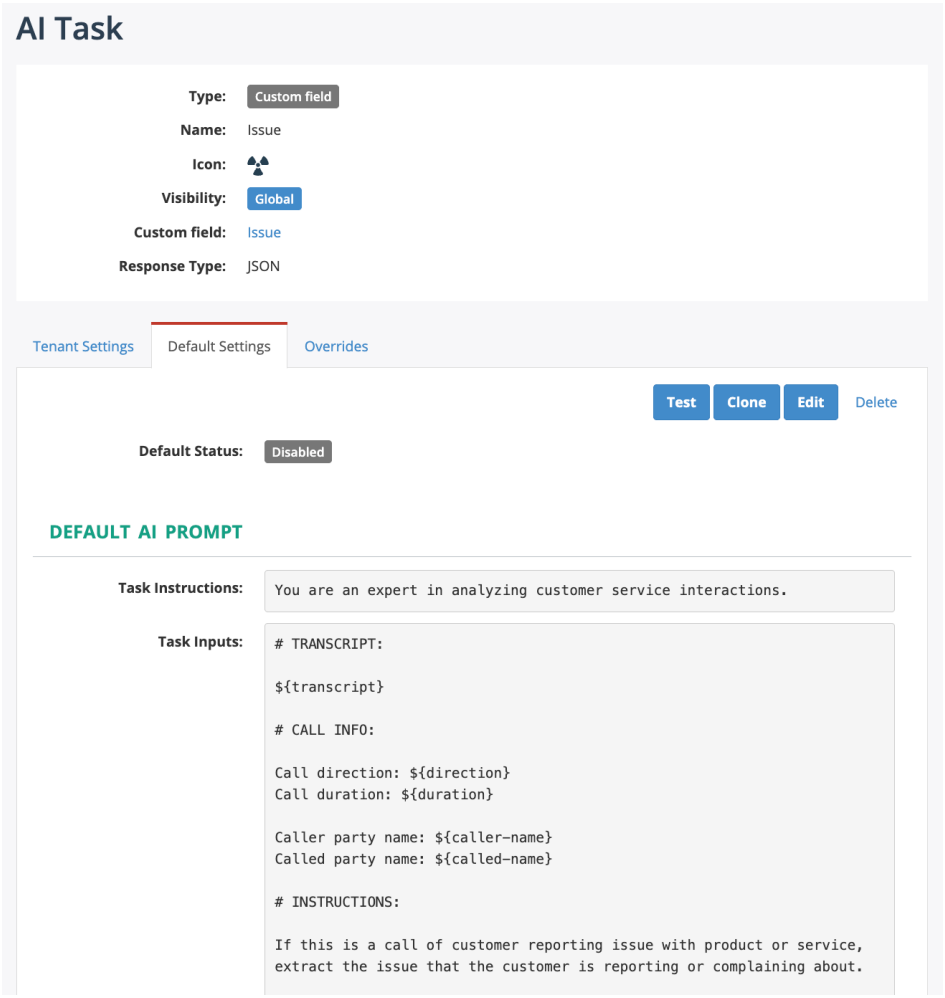

Figure: View the default task settings.

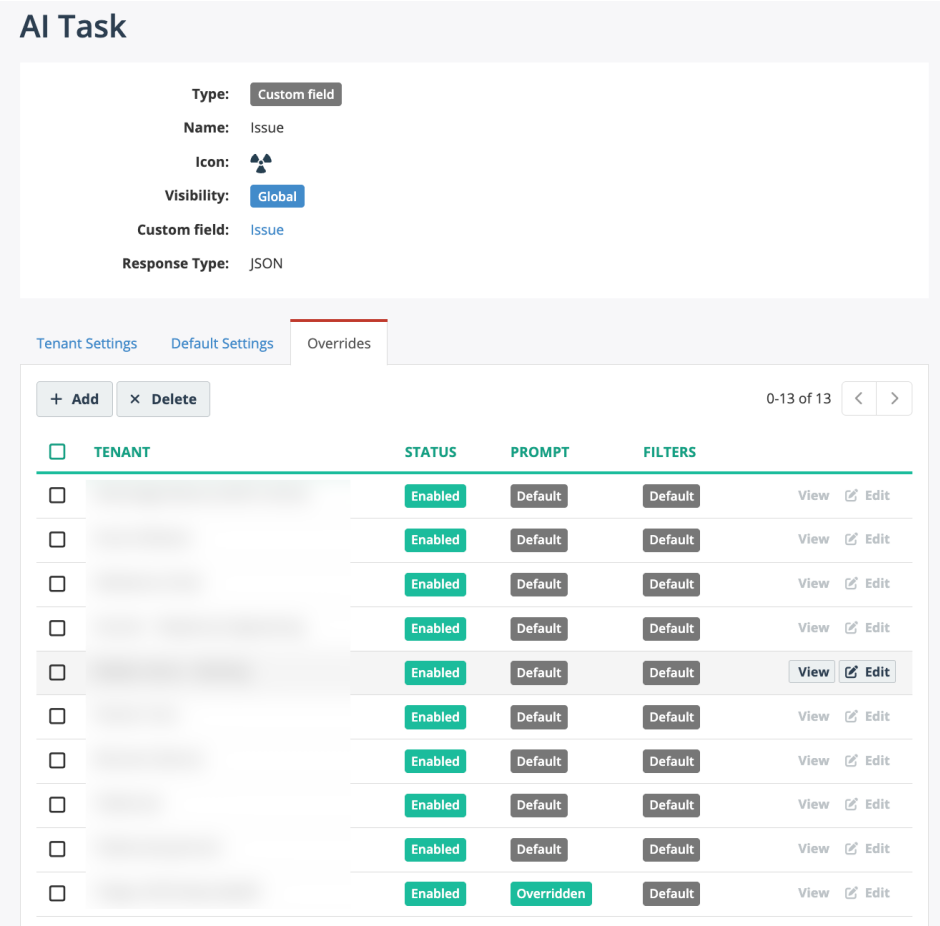

Figure: View tenants with overridden settings. Tasks with overrides show an "Overridden settings" tag.

Tenant override scope

Tenants can override only:

- **Prompt** (Task instructions and Task inputs)
- **Filters** (eligibility criteria)

Tenants cannot override:

- Attribute mapping
- Response schema
- AI engine selection

This ensures consistent data structure across tenants while allowing customization of analysis logic.

3.8 AI Assistant Job (Processing Pipeline)

The **AI Assistant job** is the background processing pipeline that executes AI Tasks against conversations and writes results to Custom Fields.

In most deployments, there is **one global continuous job** that: - continuously picks up new conversations, - determines which AI Tasks are enabled for the conversation's tenant, - applies each task's filters, - executes eligible tasks using the selected AI engine, - persists outputs to Custom Fields (and associated explanation where configured).

Conceptual execution flow

For each conversation:

1. **Eligibility check**
2. Does the conversation have content the task can analyze?
    - voice: transcript present
    - text: thread present
3. **Tenant activation**
4. Which global tasks are enabled for this tenant?
5. Which tenant-specific tasks exist?
6. **Filters**
7. Does the conversation match each task's filters (duration, direction, channel, etc.)?
8. **Execution**
9. Prompt is sent to configured AI engine.
10. Output validated (JSON/schema if configured).
11. **Persistence**
12. Output attributes mapped into Custom Fields.
13. Task execution status stored for monitoring/audit.

Operator configuration goals

- Ensure continuous processing keeps up with ingestion volume.
- Provide predictable execution latency (near-real-time vs batch).
- Prevent runaway costs from misconfigured tasks/filters.
- Provide observability: backlog, errors, per-tenant usage.

Recommended job modes

Even if the UI shows a single “job,” internally you typically want to support two modes:

1) Continuous processing (real-time-ish)

- processes newly arrived conversations
- tuned for steady throughput
- retries transient failures

2) Backfill / reprocessing (batch)

- processes historical data after:
- enabling a new task
- changing scoring logic
- fixing ingestion/transcription gaps
- throttled to avoid impacting live workloads

> If the product does not support a separate backfill job today, document a safe operational procedure to run backfills without impacting the continuous job.

Throughput, retries, and failure handling (recommended)

Operators should document/verify:

- **Concurrency controls**: number of parallel executions per worker and per engine
- **Timeouts**: per request and per conversation
- **Retry policy**: exponential backoff; max retries; when to give up
- **Dead-letter handling**: capture “poison” conversations that always fail (invalid transcript, huge transcript, etc.)
- **Idempotency**: rerunning a task should overwrite/append deterministically (define behavior)

Cost controls and safety rails

- Default filters for expensive tasks (e.g., minimum duration / minimum text length).
- Rate limit by tenant if supported (fairness).
- Disable tasks by default for new tenants until explicitly enabled.
- Monitor for spikes in:
- executions per conversation
- tokens per execution
- failure retries (can multiply cost)

Monitoring indicators (must-have)

- backlog/lag (time from ingestion to completion)
- success/failure rate per task and per engine
- percent of conversations with missing outputs
- average cost/usage per tenant (requests/tokens)
- top failing tasks and top failing tenants

Where to configure

The AI Assistant job consists of two components:

Processing Queue

Menu path: `Administration > Jobs > Processing Queues`

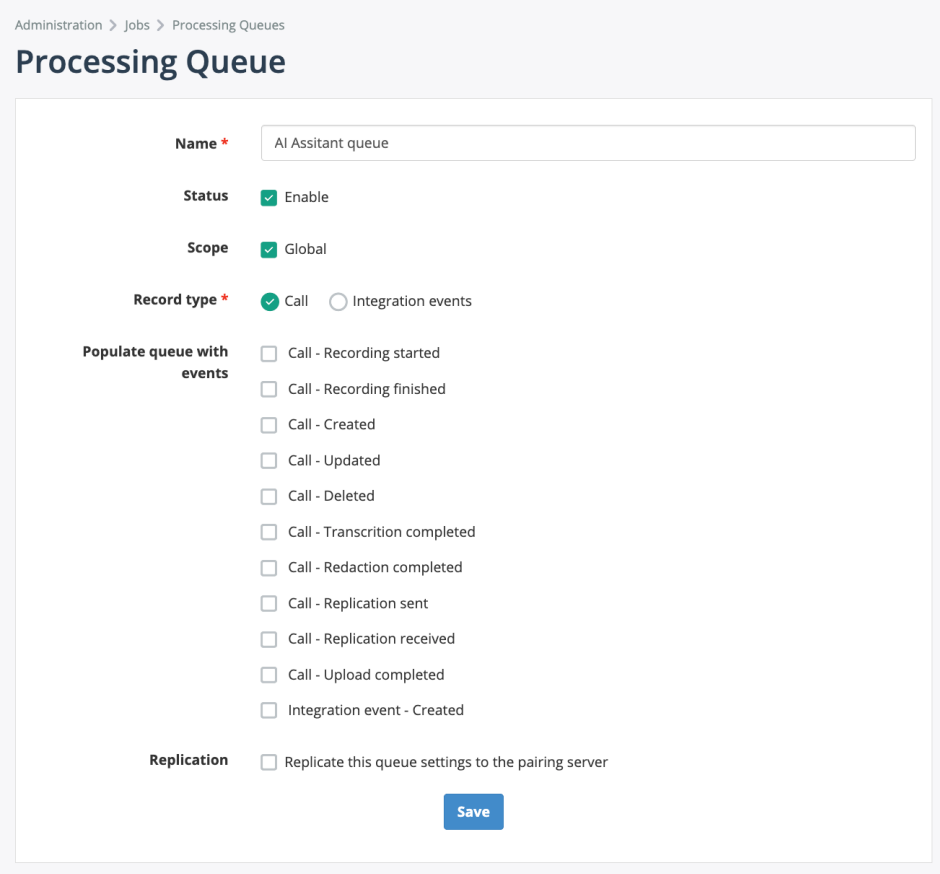

Figure: Processing Queue configuration defining which conversations are eligible for AI processing.

AI Assistant Job

Menu path: `Administration > Speech Analytics > AI Assistant > Jobs`

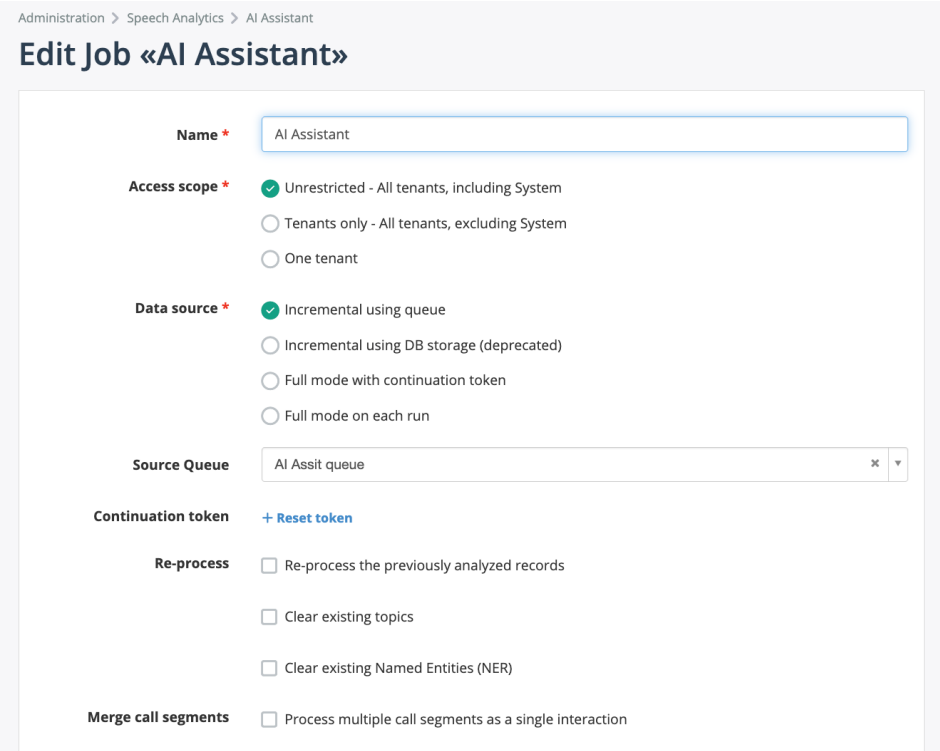

Figure: AI Assistant Job general settings including name, status, and access scope.

Job settings

When configuring an AI Assistant job:

- **Access scope** – Unrestricted (all tenants), Tenant only (specific tenant groups), or One tenant
- **Data source** – Queue-based (continuous processing) or Full mode with continuation token (one-off backfill)
- **AI engine** – Which engine to use for this job
- **Process tasks** – All tasks or Selected tasks only
- **Filtering criteria** – Additional filters (date range, call duration, direction, etc.)
- **Schedule** – When the job should run

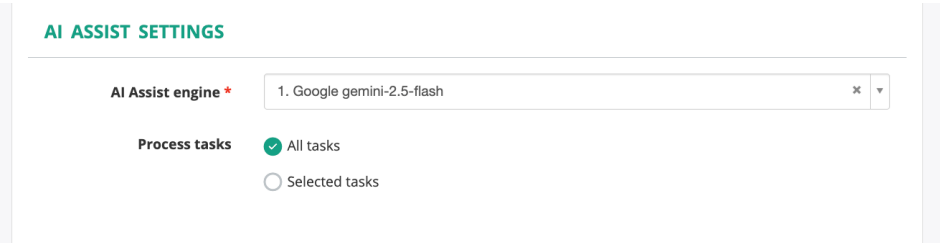

Figure: Job data source configuration and task selection.

Backfill processing

When enabling a new task or reprocessing historical data:

1. **New conversations only (default)** – When you enable a task, only new conversations are processed going forward
2. **On-demand backfill** – For historical reprocessing, contact MiaRec support to request a backfill job

To run a one-off backfill: - Create a new job with **Data source = Full mode with continuation token** - Configure the date range in filtering criteria - Run the job manually

> **Note:** Backfill jobs should be throttled to avoid impacting production workloads.

Monitoring job execution

The job view provides several monitoring tabs:

- **Latest run** – Current execution status and progress
- **All runs** – Historical execution chart showing success/failure over time
- **Processing records** – Individual conversation processing status
- **Logs** – Detailed execution logs for troubleshooting

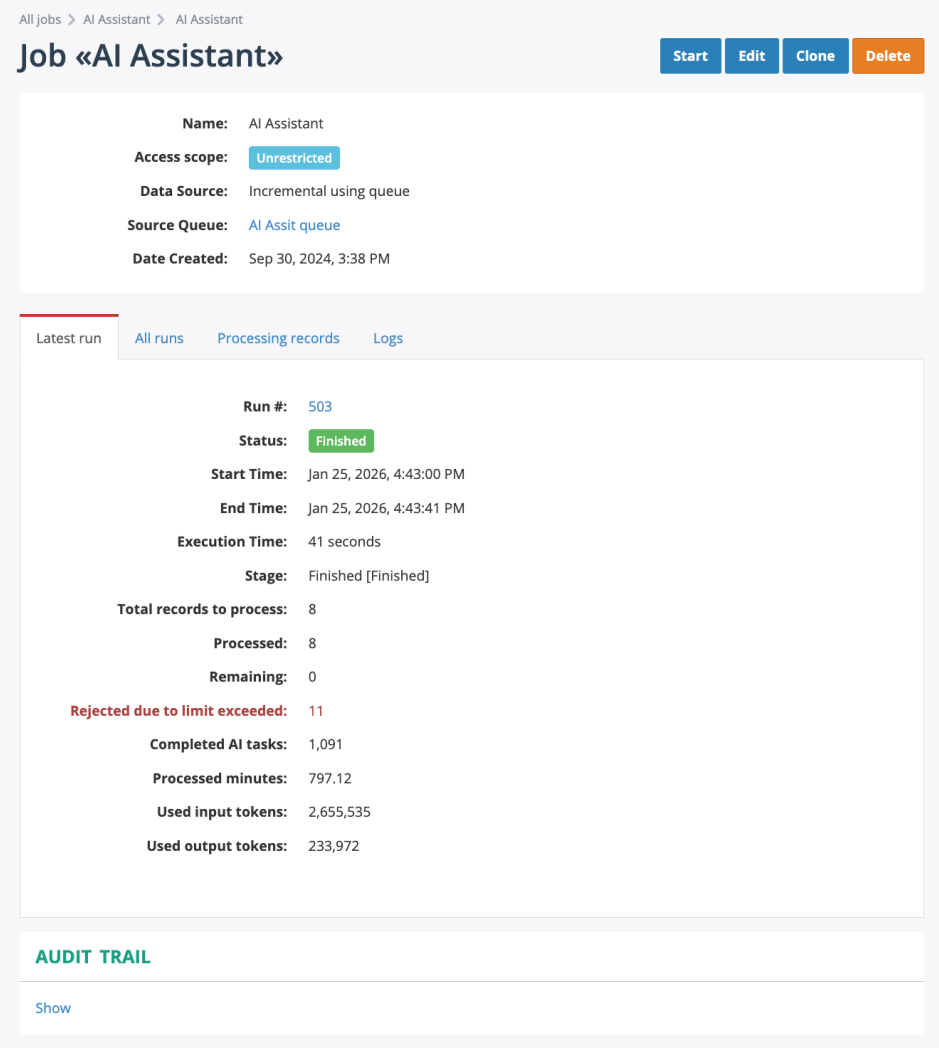

Figure: Latest run tab showing current job execution status.

4. Operations

4.1 Monitoring and Alerting

This chapter defines what “healthy” looks like for Conversation Analytics and what operators should monitor and alert on.

Goal: detect issues **before tenants notice** (missing transcripts, stale dashboards, failed AI Tasks, provider outages).

Monitoring layers (recommended)

1) Ingestion health

- ingestion throughput (conversations/hour)
- ingestion error rate
- duplicate rate (if applicable)
- latency from source → platform
- per-tenant ingestion volume anomalies

2) Transcription health (voice)

- transcript completion latency (p50/p95)
- transcript failure rate
- percent of calls with missing transcript
- language detection distribution and “unknown” rate
- provider-specific error codes/quota events

3) AI Assistant job health

- backlog/lag (time from transcript ready → tasks completed)
- execution success rate
- retry rate (high retries often = provider instability)
- dead-letter/poison count

4) AI Task health

- task-level failure rate (per task, per tenant)
- schema validation failures (JSON invalid)
- average tokens/request (cost proxy)
- number of conversations skipped due to filters (sanity check)

5) AI Engine health

- API availability (errors, timeouts)
- rate limiting/quota
- latency
- cost per tenant/task

6) User-facing indicators (synthetic checks)

- can you open a transcript?
- do dashboards load?
- does search return results for known test filters?

Alerting recommendations

Severity levels

- **SEV-1:** platform-wide outage or major data loss (ingestion down, job stopped)
- **SEV-2:** partial outage (one provider down, one region impacted)
- **SEV-3:** degradation (latency high, retries spiking)
- **SEV-4:** tenant-specific issues

Suggested alert conditions

- ingestion backlog grows continuously for > X minutes
- transcript missing rate exceeds Y%
- AI Assistant job lag exceeds Z minutes
- AI engine error rate > threshold
- schema validation failures spike after task change

- per-tenant usage spikes unexpectedly (cost guardrail)

> Operators should tune X/Y/Z based on tenant expectations and workload.

Operational dashboards (recommended)

Maintain at least: - **Pipeline health dashboard** (ingestion → transcription → AI tasks) - **Top failing tasks** (by failures and by tenants impacted) - **Usage dashboard** (requests/tokens/cost by tenant and by task) - **Overrides dashboard** (tenants with prompt/filter overrides)

Incident response workflow (recommended)

When an alert fires: 1. Identify scope: platform-wide vs tenant-specific 2. Identify stage: ingestion vs transcription vs AI job vs engine 3. Apply the runbook (see Troubleshooting → Runbooks) 4. Communicate status to impacted tenants (template recommended) 5. Post-incident: - root cause - remediation - prevention (guardrails, monitoring improvements)

Built-in monitoring in MiaRec

The AI Assistant job view provides built-in monitoring through several tabs:

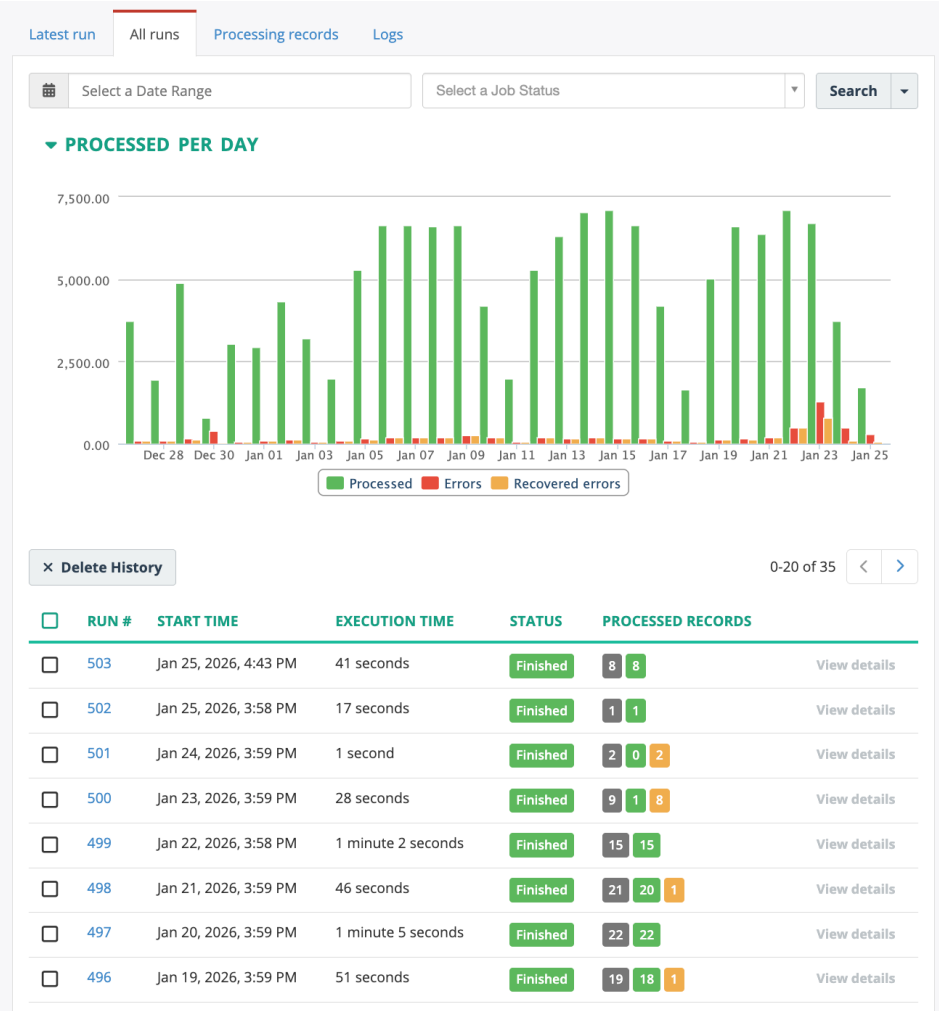

Figure: All Runs tab showing execution history with success/failure chart over time.

Job monitoring tabs

- **Latest run** – Current execution status and progress
- **All runs** – Historical execution chart showing patterns over time
- **Processing records** – Individual conversation processing status and results
- **Logs** – Detailed execution logs for troubleshooting

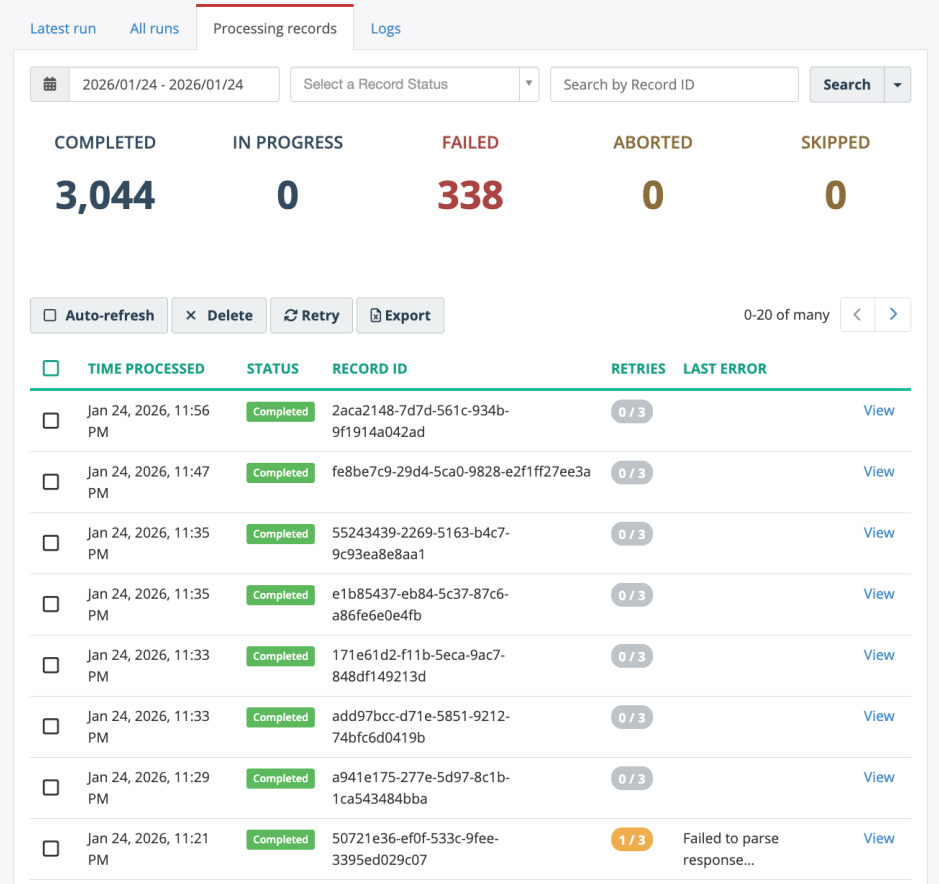

Figure: Processing Records tab showing per-conversation execution status.

AI Assistant menu structure

The AI Assistant section (`Administration > Speech Analytics > AI Assistant`) includes:

- **AI Tasks** – Task configuration and tenant activation
- **Global Tasks** – System-wide task definitions
- **Engines** – LLM provider/model configuration
- **Jobs** – Processing job configuration and monitoring

4.2 Usage, Limits, and Cost Controls

LLM-powered analytics introduces variable cost and performance. This chapter describes the controls operators should implement and document to keep the platform predictable.

Cost drivers (what matters most)

1. **Number of executions**
2. conversations processed × enabled AI Tasks per tenant
3. **Prompt + transcript size**
4. long transcripts increase tokens dramatically
5. **Retries**
6. failures with retries can multiply cost
7. **Model choice**
8. higher-end models cost more per token and may have higher latency

---

Primary cost controls (recommended)

1) Default tasks to Disabled

Publish global tasks but keep them disabled for new tenants until explicitly enabled.

2) Use task filters

Examples: - duration > 15 seconds for voice calls - minimum text length for chats/emails/tickets - exclude internal/test queues

3) Cap transcript/thread size (if supported)

Options: - truncate to last N minutes - summarize in stages (two-pass approach) - drop low-value content (hold music segments)

4) Standardize prompt templates

Avoid per-tenant prompt sprawl: - provide recommended prompts - allow overrides but track them (audit/monitor) - consider approval workflows for high-cost tasks (if practical)

5) Rate limiting / quotas (if supported)

- per-tenant execution limits per hour/day
- per-tenant token budget
- per-engine concurrency limits

Usage reporting (what to expose)

At minimum, track: - executions per tenant per task - tokens per tenant per task (input/output) - average tokens/execution - estimated cost (if provider pricing known)

Recommended UI/report: - top 10 tenants by cost - top 10 tasks by cost - anomaly detection (spikes after a prompt change)

Performance and latency considerations

- Long transcripts increase latency.
- JSON schema validation failures may increase retries (if the platform retries on invalid JSON).
- If near-real-time dashboards are needed, avoid expensive multi-output tasks in the hot path.

Guardrails for prompt changes (recommended)

Because prompts can change cost and quality: - run prompt changes through a test suite (representative transcripts) - measure token usage before/after - stage rollouts (pilot tenants) - document rollback steps

---

Implementation notes

- Track "tokens per execution" as the most actionable cost KPI

- Use task filters aggressively to control eligible conversations
- Monitor job processing records and logs for error patterns
- If no built-in quotas exist, implement operational procedures: manual throttling and staged rollouts
- Contact MiaRec for specific rate limiting and usage reporting capabilities in your deployment

4.3 Security, Compliance, and Data Governance

This chapter outlines the security and governance practices recommended for running MiaRec Conversation Analytics as a multi-tenant platform.

Because deployments vary (partner-hosted vs SaaS), treat this section as a framework and fill in the exact controls supported by MiaRec.

Security goals (operator)

- Ensure strict tenant isolation
- Protect sensitive data (audio, transcripts, message threads)
- Control and audit administrative changes (tasks, fields, engines, overrides)
- Meet regulatory requirements (retention, privacy, residency) where applicable

Data classification (recommended)

Identify and document: - content types stored: - audio recordings (voice) - transcripts / threads - AI outputs (custom fields, explanations) - sensitive fields: - personally identifiable information (PII) - payment data (PCI) - health data (HIPAA) (if applicable) - where data flows externally (transcription providers, LLM providers)

Access control model

Recommended role categories: - **Platform operator / system admin** - **Tenant admin** - **Supervisor / analyst** - **Agent / standard user** - **Read-only auditor** (optional)

Best practices: - least privilege for each role - separate operator accounts from tenant accounts - MFA/SSO enforcement for admin roles - API keys scoped by tenant and purpose

Audit logging (must-have)

Log and retain changes to: - AI Engines configuration - Global AI Tasks and Custom Fields - Tenant activation of tasks - Tenant overrides (prompt/filter) - Retention settings - User/role changes

Include: - who changed it - what changed (before/after) - when it changed - which tenant(s) are impacted

Data retention and deletion

Document: - retention defaults (audio, transcripts, threads, AI outputs) - configurable per-tenant retention (if supported) - deletion workflows and evidence (audit records) - legal hold and export mechanisms (if applicable)

Data residency and third-party processing

If you use external providers: - document what data is sent (full transcript? metadata? both?) - document provider regions and data handling guarantees - document how to select region-specific engines (if supported)

PII handling and redaction (common patterns)

Depending on product capabilities: - **Pre-ingestion redaction** (redact before MiaRec receives data) - **Post-transcription redaction** (redact in transcripts) - **Prompt-level redaction** (remove sensitive data before sending to LLM) - **Storage-level controls** (encrypt fields, restrict visibility)

Governance for AI configuration

Because AI Tasks can materially change outputs: - establish a policy for global task changes (review + test + staged rollout) - track tenant overrides as “configuration drift” - document how model changes affect comparability over time

Implementation notes

- Provide a "Data Flow" diagram for voice and text channels showing all external processors
- Require audit logs for task/prompt overrides (high value for enterprise customers)
- Provide default retention settings that partners can tune per tenant
- Document what data is sent to LLM providers (typically: transcript content with optional metadata)
- Contact MiaRec for details on specific compliance certifications and data handling practices

4.4 Upgrades and Change Management

This chapter describes how platform operators should manage change safely across tenants, including: - platform software upgrades, - AI Engine changes, - global Custom Field / AI Task updates, - prompt and schema changes.

Because AI outputs can change with prompts/models, change management is not only technical—it affects customer reporting and trust.

Change types and risk levels

Platform software changes

- UI and workflow changes
- performance and scaling changes
- schema migrations

AI Engine changes

- model upgrades (new versions)
- provider changes
- parameter changes (timeouts, token limits)

AI configuration changes

- global AI Task prompt edits
- output schema changes (JSON keys/types)
- mapping changes
- new/removed Custom Fields
- default filter changes

Recommended change process

1. **Plan**

2. identify impacted tenants and tasks
3. classify change as safe vs breaking
4. **Test**
5. run regression tests on a transcript test suite
6. verify JSON compliance and schema validation
7. measure token usage and latency changes
8. **Stage**
9. deploy to staging environment
10. run pilot tenants first
11. **Roll out**
12. gradual rollout
13. monitor errors, cost, and output shifts
14. **Communicate**
15. release notes for tenant admins (what changed and why)
16. highlight expected metric shifts
17. **Rollback**
18. have a rollback plan (revert engine/task/prompt; disable feature)

Versioning strategy (recommended)

For global AI Tasks

Prefer a **versioned task** approach: - create `CSAT Scoring v2` rather than editing the existing prompt in place when logic changes significantly. - allow tenants to migrate when ready. - deprecate v1 after a transition period.

For Custom Fields

Treat **type and computer name** as immutable. For breaking changes: - create a new field - migrate mapping in a new task version

---

Tenant overrides and drift

Tenant overrides (prompt/filter) create divergence.

Recommended operator practices: - maintain an “Overrides inventory” (who overrides what) - when shipping global updates, document: - which tenants are affected directly (no overrides) - which tenants are not affected (overridden tasks) - provide a workflow to “reset to defaults” if supported

---

Communicating AI changes to customers (recommended template)

Include: - what changed (prompt/model/schema) - why it changed (accuracy, consistency, performance) - expected effect (score distribution may shift) - what customers should do (recalibrate thresholds, revalidate) - rollback/opt-out option (if available)

Implementation notes

- Tenants automatically receive global task updates **unless** they have applied overrides
- Tenant admins can override prompt and filter settings; they can also revert to defaults
- New tasks and changes apply only to **new conversations** by default; backfill requires manual coordination
- Adopt "new version = new task" for any change that can shift metrics significantly
- Maintain a partner-facing changelog and tenant-admin facing release notes

5. Troubleshooting

5.1 Common Issues and Fixes

This chapter provides quick diagnosis and resolution steps for the most frequent platform-operator incidents.

For deeper, step-by-step procedures, see **Troubleshooting → Runbooks**.

---

1) Conversations ingest but do not appear in the UI

Likely causes - ingestion connector failure or auth expired - metadata missing tenant ID or conversation ID - indexing lag / search index failure

Quick checks - ingestion error logs - sample payload validation (tenant_id, conversation_id present) - search/index health

Quick fixes - rotate credentials - fix mapping; replay failed messages - restart indexing component (if applicable)

---

2) Voice calls have no transcripts

Likely causes - transcription engine misconfigured - provider quota/rate limiting - audio format unsupported or corrupted - transcription job backlog

Quick checks - transcription backlog and failure rate - provider dashboard (quota/limits) - test audio sample transcription

Quick fixes - scale transcription workers - adjust provider quotas or switch engine - retry failed transcription jobs - validate supported audio formats

---

3) AI Tasks are enabled but insights do not populate

Likely causes - AI Assistant job not running or backlog is high - task filters exclude most conversations (e.g., duration too high) - missing transcripts/threads for those conversations - output mapping points to missing/disabled Custom Fields

Quick checks - AI Assistant job health / backlog - check task filter hit rate (skipped due to filter) - verify Custom Fields exist and are AI-enabled

Quick fixes - start/scale AI Assistant job workers - relax filters for validation - fix mapping to correct fields - reprocess a test conversation

---

4) High rate of “invalid JSON” or schema validation failures

Likely causes - prompt does not enforce JSON-only output - model selected is weak at structured output - transcript too long; output truncated - schema too strict or mismatched with prompt instructions

Quick checks - inspect raw model outputs (sample failures) - compare prompt vs schema vs mapping keys - check token limits and truncation behavior

Quick fixes - tighten prompt (“Return JSON only. No extra text.”) - simplify schema, then tighten gradually - switch engine/model for JSON reliability - truncate inputs or use multi-stage approach

---

5) Cost spike after enabling a task or changing a prompt

Likely causes - task enabled for all tenants unintentionally - filters too permissive - prompt grew significantly - retry storm due to upstream provider failures

Quick checks - usage dashboard by tenant/task - retry rate and engine error rate - token usage per execution

Quick fixes - disable the task globally or for impacted tenants - tighten filters / add minimum duration or minimum text length - rollback prompt/version - throttle backfill/reprocessing

---

6) Dashboard shows unexpected distribution shifts

Likely causes - prompt or model changed (scoring definition drift) - thresholds misconfigured - backfill caused mixing of old/new scoring

Quick checks - task change log (what changed and when) - overrides inventory (some tenants diverged) - compare a sample of conversations before/after change

Quick fixes - recalibrate thresholds - use versioned tasks for major changes - communicate expected shifts to tenant admins

Where to inspect task outputs and logs

Job logs

Access detailed execution logs from the AI Assistant job view:

`Administration > Speech Analytics > AI Assistant > Jobs` → Select job → **Logs** tab

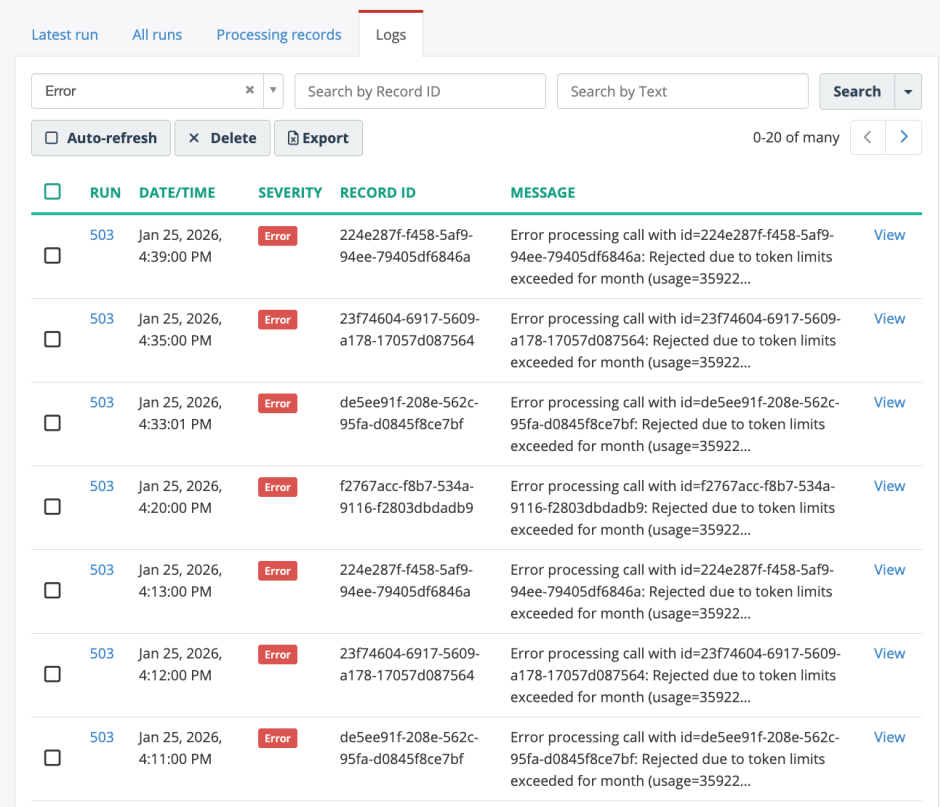

Figure: Job Logs tab showing execution log entries.

Click on a log entry to see detailed information:

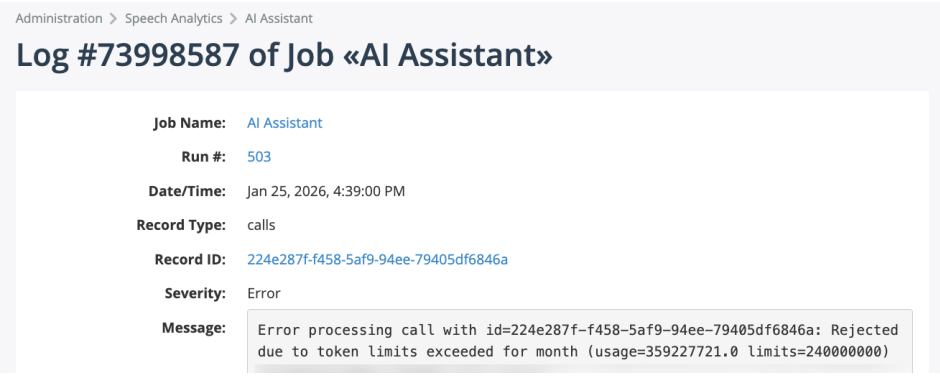

Figure: Detailed log entry showing task execution details.

Processing records

View per-conversation processing status from the **Processing records** tab on the job view. This shows which conversations were processed and their execution status.

Job controls

Operators can control AI Assistant jobs through the job configuration:

- **Enable/Disable** the job to start or stop processing
- **Schedule** settings control when the job runs
- **Filtering criteria** control which conversations are processed

5.2 Runbooks

This chapter contains deeper operational runbooks for common incidents. Each runbook follows a consistent structure:

- Symptom
- Scope assessment
- Immediate checks
- Root cause isolation
- Remediation steps
- Validation
- Escalation and what to collect
- Prevention

Runbook 1: Transcription backlog growing

Symptom

- Time from call ingestion → transcript available is increasing, backlog rising.

Scope assessment

- Affects: ☐ all tenants ☐ subset of tenants ☐ specific region/provider

Immediate checks

- transcription worker health and queue depth
- provider status/quota/rate limits
- recent deployments/config changes

Remediation

- scale transcription workers
- switch to alternate transcription engine/provider (if supported)
- throttle ingestion (temporary) if backlog threatens stability

Validation

- backlog decreases; latency returns to baseline
- new transcripts appear for test tenant

Escalation / collect

- provider error codes
- sample conversation IDs
- queue depth metrics
- worker logs

Runbook 2: AI Assistant job stopped / not processing

Symptom

- No new insights appear; job heartbeat missing; lag increasing.

Immediate checks

- job process health (service status)
- queue depth and worker availability
- credentials to AI engines still valid

Remediation

- restart job workers
- roll back recent changes
- temporarily disable expensive/unstable tasks

Validation

- job heartbeat restored
- test tenant conversation gets processed

Runbook 3: Spike in invalid JSON responses

Symptom

- Task failure rate spikes; schema validation errors increase.

Immediate checks

- identify which task(s) and engine(s)
- check recent prompt/schema updates
- examine 10 raw outputs

Remediation

- roll back prompt/schema
- switch engine/model for affected tasks
- tighten prompt to force JSON-only output
- reduce transcript length or enforce truncation

Validation

- failure rate returns to baseline
- outputs pass schema validation

Runbook 4: Suspected cross-tenant data exposure (SEV-1)

Symptom

- Any report of one tenant seeing another tenant's data.

Immediate response

- treat as SEV-1
- freeze changes and restrict access
- collect evidence and preserve logs (do not delete)

Investigation

- confirm tenant_id on affected records

- review access logs and query paths
- isolate the exposure vector (UI/API/search index)

Remediation

- hotfix enforcement at the boundary
- invalidate caches/indexes if needed
- communicate to affected parties per policy

Validation

- confirm isolation restored with synthetic tests
- run full regression suite on tenant access boundaries

Runbook 5: Cost anomaly / runaway spend

Symptom

- tokens or cost exceeds threshold; often after enabling tasks or editing prompts.

Immediate checks

- which tenant(s) and which task(s)
- retry rate
- transcript size distribution
- any unintended enablement across many tenants

Remediation

- disable the task (globally or for impacted tenants)
- tighten filters
- throttle backfill
- roll back prompt changes
- if provider rate limiting causes retries, implement fail-fast temporarily

Validation

- usage returns to baseline within expected time window

Implementation notes

Job logs are available via the **Logs** tab on the AI Assistant Job detail page. Each log entry can be expanded to view full details including error messages and processing context.

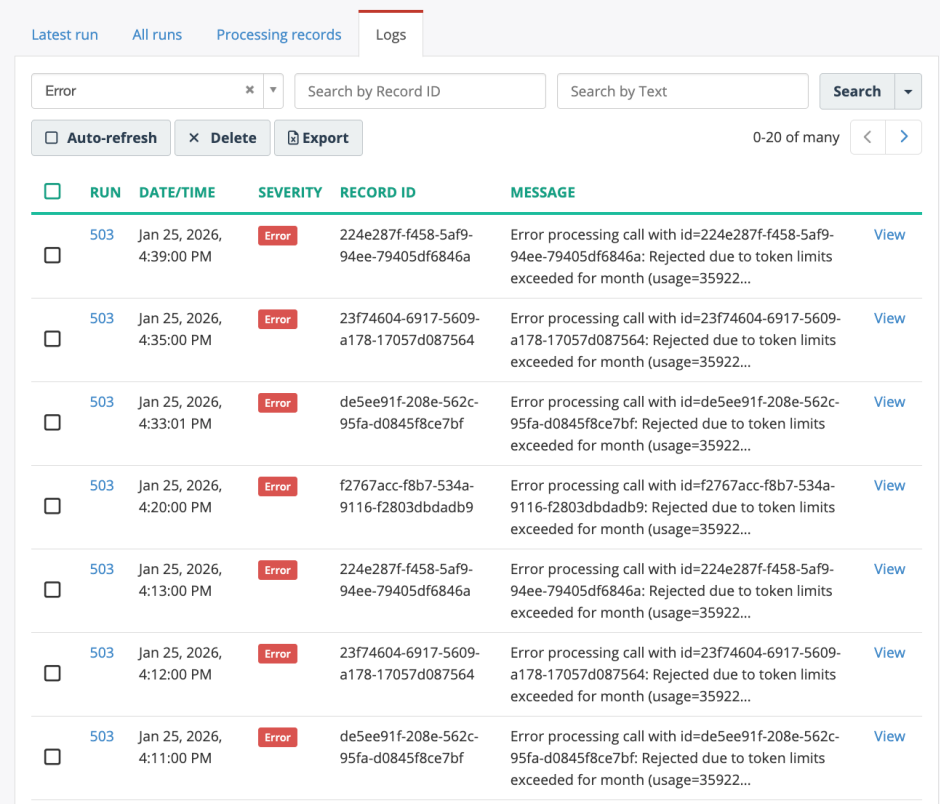

Figure: AI Assistant Job logs showing processing history.

When collecting information for MiaRec support, include:

- Conversation IDs (affected records)
- Job name and tenant
- Timestamp range of the issue
- Screenshots of error messages or log entries
- Any recent configuration changes

6. Reference

6.1 Naming Standards (Fields and Tasks)

Consistent naming is one of the highest-leverage things you can do as a platform operator. It improves: - dashboard readability - search discoverability - task reuse across tenants - upgrade safety and supportability

This chapter defines recommended naming standards for **Custom Fields**, **AI Tasks**, and **output attributes**.

Custom Fields

Display Name (human-friendly)

Use a consistent prefix by category:

- `CX – CSAT`
- `CX – NPS`
- `CX – Sentiment`
- `Conversation – Summary`
- `Conversation – Topic`
- `Sales – Lead Score`
- `Sales – Competitors Mentioned`

> If your UI already groups fields (e.g., Display Group = “CX Metrics”), you can keep display names shorter (e.g., “CSAT”).

Computer name (stable identifier)

Recommended style: - lowercase `snake_case` - no spaces - stable forever once published

Examples: - `csat` - `csat_explanation` (if using separate explanation field) - `sentiment_label` - `summary_text` - `topic_primary` - `call_reason`

Avoid

- changing computer names after tasks are mapped
- using tenant-specific jargon for global fields
- ambiguous names like `score` or `rating`

AI Tasks

Task Name

Recommended pattern: `<Category> – <Insight> (<Channel>)`

Examples: - `CX – CSAT (Call)` - `Conversation – Summarization (Call)` - `Conversation – Topic (Call)` - `Sales – Competitors Mentioned (Call)`

When omni-channel arrives, include channel only if the task is channel-specific.

Task Description

Include: - what it extracts - which fields it populates - any key filters (e.g., “Inbound only; duration > 15s”)

Versioning

For significant logic changes: - clone and suffix with `v2`, `v3` - do not change the original task in place unless change is minor

Examples: - `CX – CSAT (Call) v2`

Output attributes (task ATTRIBUTE column)

Attributes should: - be stable identifiers - be short and specific - match the JSON key names

Examples: - `csat` - `sentiment` - `topic` - `reason` - `outcome` - `explanation` (only if you map it separately)

> Recommendation: Align attribute keys with the destination field computer names where practical.

Standard categories (recommended)

- **CX**
- **Conversation**
- **Sales**
- **QA**
- **Compliance**
- **Operations**

Use these consistently across fields and tasks.

Implementation notes

- Custom Fields have both a display name (shown in UI) and an internal identifier
- Display groups (e.g., "CX Metrics", "Sales Metrics") help organize fields in the UI
- Provide a published "Global Field Catalog" and "Global Task Catalog" as reference documents
- Lock or strongly discourage edits to field identifiers after initial publication

6.2 Prompt and Schema Standards

This chapter defines recommended standards for writing prompts and output schemas for AI Tasks. These standards help ensure:

- stable, parseable outputs (especially for dashboards/search),
- predictable costs,
- easier troubleshooting,
- consistency across tenants.

Core principles

1. **Use JSON for structured outputs**
2. Scores, categories, entities, dates → JSON.
3. **Specify a strict output contract**
4. Exact keys, allowed values, and types.
5. **Return value + explanation**
6. Structured value for dashboards/search, plus a short human explanation for trust and review.
7. **Handle “unknown / not mentioned” explicitly**
8. Avoid forcing the model to guess.
9. **Keep prompts deterministic**
10. Avoid open-ended phrasing when you need a stable metric.

Prompt structure (recommended)

If the UI supports separate fields:

- **Task instructions**: role + high-level rules (short)
- **Task inputs**: transcript/thread + detailed scoring rules + response format

If the UI has one prompt field, combine them in the same order.

Variables and inputs

Common input variable (voice): - `${transcript}`

For omni-channel (text): - `${thread}` or `${messages}` (confirm actual variable names)

> Always instruct the model to use **only** the provided text; do not assume external facts.

Standard output patterns

Pattern A: Numeric score + explanation (CSAT, NPS-like)

JSON

```
{
  "csat": {
    "value": 1,
    "explanation": "Short justification grounded in transcript evidence."
  }
}
```

JSON schema (example)

```
{
  "type": "object",
  "required": ["csat"],
  "properties": {
    "csat": {
      "type": "object",
      "required": ["value", "explanation"],
      "properties": {
        "value": { "type": "integer", "minimum": 1, "maximum": 5 },
        "explanation": { "type": "string", "minLength": 1 }
      },
      "additionalProperties": false
    }
  },
  "additionalProperties": false
}
```

Pattern B: Single-select classification + explanation (Sentiment, Reason)

JSON

```
{
  "sentiment": {
    "value": "negative",
    "explanation": "Customer expresses frustration about unresolved billing issue."
  }
}
```

Prompt rule snippet - Allowed values: `positive | neutral | negative | unknown` - Use `unknown` if insufficient evidence.

Pattern C: Multi-output related fields

Use when outputs are tightly related (e.g., reason + outcome + explanation):

```
{
  "reason": { "value": "Billing", "explanation": "..." },
  "outcome": { "value": "Resolved", "explanation": "..." }
}
```

> Recommendation: Keep keys short, stable, and mapped to Custom Fields.

Pattern D: Entity extraction (Competitor, Next Action)

Return normalized values and include explanation:

```
{
  "competitor_mentioned": {
    "value": true,
    "explanation": "Customer compares pricing to AcmeCo at 04:12."
  },
  "competitor_name": {
    "value": "AcmeCo",
    "explanation": "Mentioned explicitly by the customer."
  }
}
```

“JSON only” enforcement (recommended wording)

Include a strict instruction such as:

> Output MUST be valid JSON matching the schema exactly.
> Do not include markdown, code fences, or any text outside the JSON.

Also avoid including example JSON inside code fences if your model tends to copy fences.

Handling long transcripts

If transcripts are long, consider: - truncating input (last N minutes) for certain tasks - multi-stage workflows (summarize → score), only if the platform supports task chaining - adding an instruction: “If transcript is too long, focus on the final resolution segment.”

Testing standards

For every global task: - test on at least 10 transcripts: - 2 positive - 2 negative - 2 neutral/ambiguous - 2 short calls - 2 edge cases (transfers, escalations, silence) - track: - JSON validity rate - token usage per execution - distribution stability

---

Implementation notes

MiaRec AI Tasks support the following variables in prompts:

- `${transcript}` – the call transcript
- `${direction}` – call direction (inbound/outbound)
- `${duration}` – call duration
- `${caller-name}` – caller name
- `${called-name}` – called party name
- `${options}` – dropdown options (for classification tasks)

Response type can be set to: **Auto detect**, **Text**, or **JSON**.

> Even if schema validation is not strictly enforced, define expected schemas and test JSON validity rate. Treat invalid JSON as a defect. Standardize allowed labels for classifications globally to keep dashboards meaningful.

6.3 Default Filters Library

Filters help control **cost, relevance, and quality** by limiting which conversations an AI Task runs on.

This chapter provides a library of recommended default filters for common AI Tasks. Treat these as starting points—tenants may override filters for their needs.

General guidance

- Start with **conservative filters** for expensive tasks.
- Validate filter impact by measuring:
- eligible conversations per day
- skipped conversations due to filters
- Avoid filters that silently exclude too much (e.g., duration > 2 minutes might exclude many relevant calls).

Voice call filters (speech)

Filter: Minimum duration

Use for: CSAT, sentiment, topics, reason/outcome
Recommended: duration > **15 seconds** (or 30 seconds if many short/empty calls)

Why: Very short calls often lack enough evidence for scoring.

Filter: Inbound calls only

Use for: support-oriented CX metrics (CSAT, churn risk)
Why: Outbound calls may represent sales follow-ups or proactive outreach.

Filter: Exclude internal/test calls

Use for: all analytics
How: exclude calls tagged as test, or from internal queues/numbers (depends on available metadata)

Text channel filters (chat/email/tickets)

Filter: Minimum content length

Use for: summarization, sentiment, topics
Recommended: minimum total characters or minimum number of messages (if supported)

Filter: Ticket status

Use for: resolution/outcome metrics
Recommended: analyze only resolved/closed tickets if you want final outcomes.

Task-specific filter recommendations

Summarization

- include most conversations (low filtering)
- consider excluding extremely short conversations

CSAT / NPS-like scores

- minimum duration/content length
- optional: only when a resolution is present (if metadata supports)

Topic analysis

- minimum duration/content length
- optional: exclude voicemail/hold segments if identifiable

Competitors mentioned

- focus on sales queues/campaigns if metadata exists

Measuring filter effectiveness

Recommended metrics: - eligible % of conversations - output coverage % (how many get a result) - error rate on eligible conversations - cost per eligible conversation

Available filter attributes

Common filterable attributes in MiaRec include:

- **Duration** – filter by call length
- **Direction** – inbound / outbound
- **Channel** – call / chat / email (as supported)
- **Queue/team** – filter to specific routing groups

> Provide a standard set of filters for every global task and document them. Track filter skip rates to detect misconfiguration early.