Aid4Mail AI Provider and Model Configuration Guide

User Guide

Fookes Software Ltd
Charmey, Switzerland
www.aid4mail.com

Table of Contents

Aid4Mail AI Provider and Model Configuration Guide

This document describes the modular AI provider and model configuration system used by Aid4Mail. It covers the folder structure, file formats, placeholder mechanics, and customization procedures.

For guidance on choosing providers and models (accuracy, speed, cost, multilingual support), refer to the Aid4Mail AI Provider and Model Selection Guide. For feature documentation, setup instructions, and best practices, refer to the Aid4Mail AI Email Review Workflow Guide.

---

1. Design Principles

1. **Self-contained model files**: Each model file includes everything needed to function, with optional provider overrides.
2. **No implicit inheritance**: Avoids stale user overrides masking new features.
3. **Explicit override semantics**: Model configs can override provider defaults; user configs override system configs (file-level replacement).
4. **Retired model compatibility**: Older models continue working because they carry their complete configuration.

---

2. Directory Structure

All configuration files are located in the `AI Config` subfolder of the Aid4Mail program folder.

```
AI Config/
├── Archived_Models.zip       # Retired models
├── system_prompts.json       # Global task instructions (analyze, classify,
filter)
├── README.md                 # This file
│
├── Amazon Bedrock/           # AWS Bedrock (Claude via AWS)
│   ├── provider_config.json
│   ├── Claude Haiku 4.5 (Amazon Bedrock).json
│   ├── Claude Sonnet 4.5 (Amazon Bedrock).json
│   ├── Claude Sonnet 4.6 (Amazon Bedrock).json †
│   ├── Claude Opus 4.5 (Amazon Bedrock).json †
│   ├── Claude Opus 4.6 (Amazon Bedrock).json †
```

```
│   └── Claude Opus 4.7 (Amazon Bedrock).json †
│
├── Anthropic/                # Anthropic (Claude)
│   ├── provider_config.json
│   ├── Claude 4.5 Haiku.json
│   ├── Claude 4.5 Sonnet.json
│   ├── Claude 4.6 Sonnet.json †
│   ├── Claude 4.5 Opus.json †
│   ├── Claude 4.6 Opus.json †
│   └── Claude 4.7 Opus.json †
│
├── Gemini Enterprise Agent Platform/ # Google's enterprise platform (formerly
Google Vertex AI)
│   ├── provider_config.json
│   └── Gemini 3.1 Flash-Lite (Agent Platform).json †
│
├── Google/                   # Google AI (Gemini)
│   ├── provider_config.json
│   ├── Gemini 2.5 Flash.json
│   ├── Gemini 3.0 Flash (preview).json †
│   └── Gemini 3.1 Flash-Lite.json †
│
├── Google Vertex AI (Claude)/ # Enterprise-grade platform
│   ├── provider_config.json
│   ├── Claude Sonnet 4.5 (Google Vertex).json
│   ├── Claude Sonnet 4.6 (Google Vertex).json †
│   ├── Claude Opus 4.5 (Google Vertex).json †
│   ├── Claude Opus 4.6 (Google Vertex).json †
│   └── Claude Opus 4.7 (Google Vertex).json †
│
├── Google Vertex AI (Gemini)/ # Enterprise-grade platform
│   ├── provider_config.json
│   ├── Gemini 2.5 Flash (Google Vertex).json
│   └── Gemini 3.0 Flash (Google Vertex).json †
│
├── LM Studio/                # LM Studio (Local models)
│   ├── provider_config.json
│   └── Local Model (LM Studio).json
│
├── Meta AI/                  # Meta AI via LLMAPI (Llama, DeepSeek)
│   ├── provider_config.json
│   ├── Deepseek V3 (Llama).json
│   └── Llama 4 Maverick.json
│
├── Microsoft Foundry Models/ # Third-party models via Model Inference API
│   ├── provider_config.json
│   ├── GPT-5.2 (Foundry).json †
│   ├── GPT-5.4 (Foundry).json †
│   ├── Grok 4.1 Fast (Foundry)
│   ├── Grok 4.1 Fast+Reasoning (Foundry) †
│   ├── Grok 4.2 Non-Reasoning (Foundry).json
│   ├── Grok 4.2 Reasoning (Foundry).json †
│   └── Kimi K2 Thinking (Foundry).json †
│
├── Mistral AI/               # Mistral AI (Magistral, Mistral)
│   ├── provider_config.json
│   ├── Magistral Medium.json †
```

```
│   ├── Mistral Large 3.json
│   ├── Mistral Medium 3.json
│   └── Mistral Small 3.2.json
│
├── Ollama/                  # Ollama (Local models)
│   ├── provider_config.json
│   ├── Gemma 3 27B (Ollama).json
│   ├── Gemma 4 26B (Ollama).json †
│   ├── Gemma 4 31B (Ollama).json †
│   ├── GPT-OSS 120B (Ollama).json †
│   ├── GPT-OSS 20B (Ollama).json †
│   ├── Granite 4.1 30B (Ollama).json
│   ├── Llama 3.3 70B (Ollama).json
│   ├── Magistral 24B (Ollama).json †
│   ├── Ministral 3 14B (Ollama).json
│   ├── Mistral Small 3.2 24B (Ollama).json
│   ├── Nemotron 3 33b (Ollama).json †
│   ├── Qwen 3.5 27B (Ollama).json †
│   ├── Qwen 3.6 27B (Ollama).json †
│   └── Qwen 3.6 35B (Ollama).json †
│
├── OpenAI/                  # OpenAI (GPT)
│   ├── provider_config.json
│   ├── OpenAI GPT-5.2.json †
│   ├── OpenAI GPT-5.4.json †
│   └── OpenAI GPT-5.5.json †
│
└── xAI/                     # xAI (Grok)
    ├── provider_config.json
    ├── Grok 4.1 Fast.json
    ├── Grok 4.1 Fast+Reasoning.json †
    ├── Grok 4.2 Non-Reasoning.json
    ├── Grok 4.2 Reasoning.json †
    └── Grok 4.3.json †
```

† Non-deterministic—may produce different results between runs of the same prompt. See Section 3.3 for details.

Each provider has its own folder containing a `provider_config.json` (shared defaults) and one or more model JSON files. The model file's filename (without the `.json` extension) becomes its display name in the Aid4Mail GUI.

3. File Specifications

3.1. provider_config.json

Provider-level defaults that apply to all models in the folder unless overridden by an individual model config.

Field	Type	Required	Description
`api_endpoint`	string	Yes	API URL (may include placeholders)
`request_headers`	string	Yes	HTTP headers (CRLF-separated)
`requires_api_key`	boolean	No	If `false`, GUI hides the API key editor (default: `true`)
`requires_endpoint`	boolean	No	If `true`, GUI shows the endpoint URL editor
`requires_region`	boolean	No	If `true`, GUI shows the region selector
`requires_project_id`	boolean	No	If `true`, GUI shows the project ID field
`requires_resource_name`	boolean	No	If `true`, GUI shows the resource name field
`requires_deployment_name`	boolean	No	If `true`, GUI shows the deployment name field
`requires_service_account`	boolean	No	If `true`, GUI shows the service account key file field
`default_endpoint`	string	No	Default endpoint URL when `requires_endpoint` is `true`
`default_endpoint_region`	string	No	Default region when `requires_region` is `true`
`endpoint_regions`	array	No	List of valid region codes for the region selector dropdown
`tokens`	object	Yes	Response parsing configuration
`tokens.path_to_input_count`	string	Yes	JSON path to input token count
`tokens.path_to_output_count`	string	No	JSON path to output token count
`tokens.path_to_total_count`	string	No	JSON path to total token count
`tokens.regex_payload_error`	string	Yes	Regex to detect payload errors
`tokens.regex_excess_tokens`	string	No	Regex to extract excess token count

Notes:

- When `requires_region` is `true`, the optional `endpoint_regions` array populates the GUI dropdown. If omitted, the user enters the region manually.
- Set `requires_api_key` to `false` for local (offline) providers like Ollama and LM Studio.
- When `default_endpoint` or `default_endpoint_region` is specified, the provider works out-of-the-box without user configuration. Users can still override these defaults in the GUI.
- When `requires_project_id` is `true`, the GUI prompts for a project identifier (used by Gemini Enterprise Agent Platform / Vertex AI API).
- When `requires_resource_name` is `true`, the GUI prompts for an Azure resource name (used by Microsoft Foundry Models).

- When `requires_service_account` is `true`, the GUI prompts for a Google Service Account key file path (used by Gemini Enterprise Agent Platform / Vertex AI API for OAuth 2.0 authentication). See Section 7 for details.

3.2. Model Configuration Files

Each model is a single JSON file. The filename (minus the `.json` extension) becomes the model's display name in the GUI.

Field	Type	Required	Description
`provider`	object	No	Override provider_config.json settings
`provider.api_endpoint`	string	No	Override API endpoint
`provider.request_headers`	string	No	Override HTTP headers
`provider.tokens`	object	No	Override token parsing
`metadata`	object	Yes	Model identification and limits
`metadata.model_id`	string	Yes	API model identifier
`metadata.description`	string	Yes	Human-readable description
`metadata.input_token_limit`	integer	Yes	Maximum input context window
`metadata.output_token_limit`	integer	Yes	Maximum output tokens
`metadata.default_output_tokens`	integer	Yes	Default output token limit
`metadata.max_tokens_field_name`	string	No	Override the max tokens field name
`metadata.temperature_field`	boolean	No	If `false`, omit the temperature field from requests
`structured_output`	object	Yes	Config for filtering and enumerated classification
`unstructured_output`	object	Yes	Config for analysis and open-ended classification

3.3. Temperature Handling and Reproducibility

Some models do not accept a user-supplied temperature value. The API either rejects requests that include the field or only honors a fixed default. For these models, set `"temperature_field": false` in the `metadata` block and omit the `"temperature"` field from the `email_analysis` and `prompt_validation` payloads. Aid4Mail will then send each request without a temperature, allowing the model to use its internal default.

Separately, models that perform internal reasoning before producing a final answer (often called reasoning or thinking models) may produce different results between runs of the same prompt against the same email, even when temperature is set to 0. This variability is a property of the reasoning architecture and cannot be controlled through configuration. Users who require strictly reproducible classifications should select a non-reasoning model.

For the current list of models affected by either behavior, see the directory tree in Section 2—entries marked with † are non-deterministic.

---

4. Configuration Resolution Order

When Aid4Mail loads a model configuration, it follows this sequence:

1. Load `provider_config.json` from the provider folder.
2. Load the model's JSON file.
3. If the model has a `provider` block, merge it over `provider_config.json` (field-level merge for the `provider` block only).
4. Substitute placeholders in order:
    - `<model_id>` ← `metadata.model_id`
    - `<provider_endpoint_region>` ← User's GUI setting (e.g., `us-east-1`)
    - `<provider_project_id>` ← User's GUI setting (e.g., `aid4mail-ai`)
    - `<provider_resource_name>` ← User's GUI setting (e.g., `aid4mail-openai`)
    - `<provider_deployment_name>` ← User's GUI setting (e.g., `gpt-5-2-production`)
    - `<provider_endpoint>` ← User's custom endpoint URL
    - `<api_key>` ← User's stored API key
    - `<temperature>`, `<max_tokens>`, etc. ← Runtime values

---

5. Placeholder Reference

5.1. Standard Placeholders

Placeholder	Source	Description
`<model_id>`	`metadata.model_id`	API model identifier
`<api_key>`	App Settings	User's provider API key
`<service_account_token>`	Runtime	OAuth 2.0 bearer token from service account (auto-refreshed)
`<provider_endpoint_region>`	App Settings	Regional endpoint (e.g., `us-east-1`, `eu-west-1`)
`<provider_project_id>`	App Settings	Project identifier (e.g., Google Cloud project ID)
`<provider_resource_name>`	App Settings	Azure resource name (e.g., `aid4mail-openai`)
`<provider_deployment_name>`	App Settings	Azure deployment name (e.g., `gpt-5-2-production`)

`<provider_endpoint>`	App Settings	User-defined endpoint URL (e.g., Hugging Face endpoints)
`<user_prompt>`	Project Settings	User's classification/analysis prompt
`<email_data>`	Runtime	Processed email content
`<fixed_instructions>`	`system_prompts.json`	Task-specific system prompt
`<validate_instructions>`	`system_prompts.json`	Validation system prompt
`<temperature>`	Runtime	Temperature setting
`<max_tokens>`	Project Settings	User-specified max output tokens
`<default_output_tokens>`	`metadata`	Default output token limit
`<answer_json_enum>`	Runtime	JSON array of valid classification values
`<answer_csv_list>`	Runtime	Comma-separated list of valid values
`<max_tokens_field_name>`	`metadata`	Field name for max tokens

5.2. Typed Placeholders for JSON Validity

Model configuration files must be valid JSON to allow validation in editors like VS Code. However, some placeholders represent non-string JSON types (arrays, numbers). To maintain JSON validity while supporting these types, use **typed placeholder syntax**.

Array Placeholders: <array:token>

Use for placeholders that expand to JSON arrays.

Syntax	Expands To	Final Result
`"<array:answer_json_enum>"`	`[<answer_json_enum>]`	`["Yes", "No", "Maybe"]`

Example:

```
{
  "enum": "<array:answer_json_enum>"
}
```

At runtime, this becomes:

```
{
  "enum": ["Yes", "No", "Maybe"]
}
```

Numeric Placeholders: <number:token>

Use for placeholders that expand to numeric values (integers or floats).

Syntax	Expands To	Final Result
`"<number:max_tokens>"`	`<max_tokens>`	`1024`
`"<number:temperature>"`	`<temperature>`	`0.7`

`"<number:default_output_tokens>"`	`<default_output_tokens>`	`512`

Example:

```
{
  "max_tokens": "<number:max_tokens>",
  "temperature": "<number:temperature>"
}
```

At runtime, this becomes:

```
{
  "max_tokens": 1024,
  "temperature": 0.7
}
```

Summary: When to Use Each Style

JSON Value Type	Placeholder Style	Example
String	`"<token>"`	`"model": "<model_id>"`
Number	`"<number:token>"`	`"temperature": "<number:temperature>"`
Array	`"<array:token>"`	`"enum": "<array:answer_json_enum>"`

String placeholders like `"<model_id>"` don't need special syntax because they're already valid JSON strings.

---

6. Provider-Specific Endpoint Templates

Each provider uses a different URL pattern and authentication mechanism. The sections below show the key `provider_config.json` fields for each provider type.

6.1. Direct API Providers (Anthropic, Google, Mistral AI, OpenAI, xAI)

Most direct API providers use a static endpoint with an API key header. No special placeholders beyond `<model_id>` and `<api_key>` are required. Example:

```
{
  "api_endpoint": "https://api.example.com/v1/chat/completions",
  "request_headers": "Content-Type: application/json\r\nAuthorization: Bearer
<api_key>",
  "requires_api_key": true
}
```

6.2. Amazon Bedrock

Uses `<provider_endpoint_region>` for regional deployment. Authentication is handled via AWS SigV4 signing.

```
{
  "api_endpoint": "https://bedrock-
runtime.<provider_endpoint_region>.amazonaws.com/model/<model_id>/converse",
  "requires_region": true
}
```

6.3. Gemini Enterprise Agent Platform

Uses `<provider_endpoint_region>`, `<provider_project_id>`, and service account authentication.

```
{
  "api_endpoint":
"https://aiplatform.<provider_endpoint_region>.rep.googleapis.com/v1/projects/<prov
ider_project_id>/locations/<provider_endpoint_region>/publishers/google/models/<mod
el_id>:generateContent,
  "request_headers": "Authorization: Bearer <service_account_token>",
  "requires_api_key": false,
  "requires_region": true,
  "requires_project_id": true,
  "requires_service_account": true
}
```

6.4. Google Vertex AI (Gemini)

Uses `<provider_endpoint_region>`, `<provider_project_id>`, and service account authentication.

```
{
  "api_endpoint": "https://<provider_endpoint_region>-
aiplatform.googleapis.com/v1/projects/<provider_project_id>/locations/<provider_end
point_region>/publishers/google/models/<model_id>:generateContent",
  "request_headers": "Authorization: Bearer <service_account_token>",
  "requires_api_key": false,
  "requires_region": true,
  "requires_project_id": true,
  "requires_service_account": true
}
```

6.5. Google Vertex AI (Claude)

Similar to the Gemini variant, but uses the Anthropic publisher path and `rawPredict` endpoint.

```
{
  "api_endpoint": "https://<provider_endpoint_region>-
aiplatform.googleapis.com/v1/projects/<provider_project_id>/locations/<provider_end
point_region>/publishers/anthropic/models/<model_id>:rawPredict",
  "request_headers": "Authorization: Bearer <service_account_token>\r\nContent-
Type: application/json",
  "requires_api_key": false,
  "requires_region": true,
  "requires_project_id": true,
  "requires_service_account": true
}
```

6.6. Microsoft Foundry Models

Uses `<provider_resource_name>`. The Model Inference API specifies the model in the request body rather than the URL, so no deployment name placeholder is required in the endpoint.

```
{
  "api_endpoint":
"https://<provider_resource_name>.services.ai.azure.com/models/chat/completions?api
-version=2024-05-01-preview",
  "requires_resource_name": true
}
```

6.7. Local Providers (Ollama, LM Studio)

Uses `<provider_endpoint>` with a default endpoint for zero-configuration setup.

```
{
  "api_endpoint": "<provider_endpoint>/v1/chat/completions",
  "request_headers": "Content-Type: application/json",
  "requires_api_key": false,
  "requires_endpoint": true,
  "default_endpoint": "http://127.0.0.1:11434"
}
```

With `default_endpoint` specified, the provider works immediately without user configuration. Users can override the default in the GUI if needed (e.g., for a remote Ollama server).

---

7. Service Account Authentication (Gemini Enterprise Agent Platform / Vertex AI API)

Aid4Mail's Gemini Enterprise Agent Platform / Vertex AI configurations use OAuth 2.0 authentication with a Google Cloud Service Account instead of static API keys. Google has renamed the platform surface, but the API service, IAM role, project, location, endpoint, and service-account authentication concepts used here remain Vertex AI API (`aiplatform.googleapis.com`) concepts. This authentication method is suitable for long-running email processing workflows where jobs may run for hours or days.

7.1. How It Works

Aid4Mail implements the **Service Account OAuth 2.0 JWT Bearer flow**:

1. **Load credentials**: Aid4Mail reads your Service Account JSON key file containing the private key and client email.
2. **Create JWT**: A JSON Web Token is created with required claims (issuer, scope, audience, expiration).
3. **Sign JWT**: The JWT is signed using RSA-SHA256 with the service account's private key.
4. **Exchange for token**: The signed JWT is exchanged for an access token via Google's OAuth 2.0 token endpoint.

5. **Auto-refresh**: Tokens are cached and automatically refreshed ~5 minutes before expiration.

This flow runs transparently—you simply configure the service account key file path, and Aid4Mail handles all token management automatically.

7.2. Setup Instructions

Step 1: Create a Service Account in the Google Cloud Console

1. Go to Google Cloud Console.
2. Select your project (or create a new one).
3. Navigate to **IAM & Admin > Service Accounts**.
4. Click **Create Service Account**.
5. Enter a name (e.g., `aid4mail-ai`) and description.
6. Click **Create and Continue**.
7. Grant the Service Account the role **Agent Platform User** (`roles/aiplatform.user`).
8. Click **Done**.

Step 2: Create and Download the Key File

1. Click on the newly created service account.
2. Go to the **Keys** tab.
3. Click **Add Key > Create new key**.
4. Select **JSON** format.
5. Click **Create**—the key file downloads automatically.
6. Store this file securely (e.g., `C:\Aid4Mail\Credentials\aid4mail-ai-171c58d6927f.json`).

Security Note: The JSON key file contains a private key that grants access to your Google Cloud resources. Store it securely and never commit it to version control.

Step 3: Enable the Vertex AI API (aiplatform.googleapis.com)

1. In Google Cloud Console, navigate to **APIs & Services > Library**.
2. Search for "Agent Platform API" (`aiplatform.googleapis.com`).
3. Click **Enable**.

Step 4: Configure Aid4Mail

In Aid4Mail's **App Settings > AI**, locate the Gemini Enterprise Agent Platform / Vertex AI provider and specify:

- **Project ID**: Your Google Cloud project ID (e.g., `my-project-123456`)
- **Region**: A supported Google Cloud region/location for the selected model (e.g., `us-central1`)
- **Service Account Key File**: Full path to your JSON key file

7.3. Service Account Key File Format

The downloaded JSON key file has this structure (do not modify it):

```
{
  "type": "service_account",
  "project_id": "your-project-id",
  "private_key_id": "key-id-here",
  "private_key": "-----BEGIN PRIVATE KEY-----\n...\n-----END PRIVATE KEY-----\n",
  "client_email": "aid4mail@your-project-id.iam.gserviceaccount.com",
  "client_id": "123456789",
  "auth_uri": "https://accounts.google.com/o/oauth2/auth",
  "token_uri": "https://oauth2.googleapis.com/token",
  "auth_provider_x509_cert_url": "https://www.googleapis.com/oauth2/v1/certs",
  "client_x509_cert_url": "https://www.googleapis.com/robot/v1/metadata/x509/..."
}
```

Aid4Mail uses the `private_key` and `client_email` fields for authentication. The `project_id` from the file is also extracted but can be overridden in the GUI settings.

7.4. Token Lifecycle

- **Token duration**: Access tokens are valid for 1 hour.
- **Auto-refresh**: Aid4Mail refreshes tokens 5 minutes before expiration.
- **Error recovery**: If a 401 (Unauthorized) error occurs, the token is automatically refreshed.
- **No user interaction**: Token refresh happens transparently during processing.

7.5. Troubleshooting

Error	Cause	Solution
"Service account key file not found"	Invalid file path	Verify the path in App Settings
"Service account JSON missing 'private_key' field"	Corrupted or invalid key file	Re-download the key from Google Cloud Console
"Token exchange failed: invalid_grant"	Clock skew or expired key	Sync system clock; regenerate key if expired
"Token exchange failed: unauthorized_client"	Service account lacks permissions	Grant "Vertex AI User" role in IAM
"HTTP 403 Forbidden"	API not enabled or quota exceeded	Enable Vertex AI API (`aiplatform.googleapis.com`); check quotas

8. User Customization

Users can override system configurations or add entirely new providers by creating a parallel folder structure in their user data directory:

```
%APPDATA%\Aid4Mail6\AI Config\
├── Anthropic/
│   └── Claude 4.6 Sonnet.json   # Completely replaces the system version
│
└── Custom Provider/             # User-added provider
    ├── provider_config.json
    └── My Custom Model.json
```

Merge behavior:

- System configs load first.
- User configs override at the **file level** (complete replacement, not field merge). If you place a `Claude 4.5 Sonnet.json` in the user directory, it entirely replaces the system version of that file.
- User-added providers (folders that don't exist in the system directory) appear alongside system providers in the GUI.

---

9. Retired Models and Archived Configurations

When a model is retired:

1. Its JSON file is removed from the installer's `AI Config/` distribution.
2. The model file is published on the Aid4Mail website as a downloadable archive.
3. Users who need the retired model download and place it in their user data directory (see Section 8).
4. The retired model continues working because it's self-contained—no schema changes affect it.

Restoring Archived Models

The `Archived_Models.zip` file in the `AI Config\` folder contains retired models that may still be functioning. To restore one or more:

1. Extract the archive to `%APPDATA%\Aid4Mail6\AI Config\`.
2. Ensure the folder structure from the archive is preserved during extraction.
3. Restart Aid4Mail—the restored models will become available in the GUI.

---

10. Adding New Providers and Models

10.1. Adding a New Provider

1. Create a folder with the desired display name inside `%APPDATA%\Aid4Mail6\AI Config\` (e.g., `My Provider/`).
2. Add a `provider_config.json` with the required fields (see Section 3.1).
3. Add one or more model JSON files (see Section 3.2).

10.2. Adding a New Model to an Existing Provider

1. Create a JSON file in the appropriate provider folder (either the system folder for distribution, or the user folder for personal use): e.g., `My Model.json`.
2. Include all required fields in `metadata`, `structured_output`, and `unstructured_output`.
3. Optionally add a `provider` block to override provider defaults for this specific model.

The easiest approach is to copy an existing model file from the same provider and modify the `metadata.model_id`, `metadata.description`, token limits, and any provider overrides as needed.

10.3. Configuring a Local (Offline) Provider

For local AI deployments (Ollama, LM Studio, or other local inference servers):

1. Set `"requires_api_key": false` and `"requires_endpoint": true` in the provider's `provider_config.json`.
2. Use the `<provider_endpoint>` placeholder in the `api_endpoint` field.
3. Optionally set `"default_endpoint"` for zero-configuration setup.
4. The GUI will show an endpoint URL field instead of an API key field.

See Section 6.6 for a complete example.

11. Local AI Configuration

This section covers the configuration settings specific to running offline models with Ollama or LM Studio. For installation instructions, model downloads, and server setup, refer to the *Aid4Mail AI Provider and Model Selection Guide* (Sections 4.7 and 4.8). For model selection guidance (accuracy, speed, hardware requirements), refer to the same guide (Section 4).

11.1. Supported Offline Models

The following models are pre-configured in Aid4Mail and have been tested for email classification. The table shows the minimum VRAM required to run the listed models at optimal throughput.

Model	Context @ Min. VRAM	Min. VRAM (Q4_K_M, MXFP4)
Gemma 3 27B	32K	32 GB
Gemma 4 26B	32K	32 GB
Gemma 4 31B	32K	32 GB
GPT-OSS 20B	32K	24 GB
GPT-OSS 120B	32K	96 GB
Granite 4.1 30B	32K	32 GB
Llama 3.3 70B	64K	80 GB

Magistral 24B	32K	24 GB
Ministral 3 14B	32K	16 GB
Mistral Small 3.2 24B	32K	24 GB
Nemotron 3 33B	32K	32 GB
Qwen 3.5 27B	32K	32 GB
Qwen 3.6 27B	32K	32 GB
Qwen 3.6 35B	32K	32 GB

All offline models should be deployed with **Q4_K_M or MXFP4 quantization**. Higher-precision variants (Q8_0, FP16) significantly reduce processing speed without meaningful improvements in classification accuracy for email tasks.

11.2. Context Length Configuration

Context length determines how much text the model can process in a single request and directly affects GPU memory usage. Setting it too high can push the model out of VRAM and cause severe slowdowns; setting it too low risks truncating emails.

Recommended Context Length by VRAM (Q4_K_M)

Installed VRAM	14B (up to 256K*)	20B–27B (up to 128K)	35B MoE (up to 128K)	70B (up to 128K)
16 GB	32K	Not recommended	Not recommended	Not recommended
24 GB	64K	32K	Not recommended	Not recommended
32 GB	128K	64K	32K	Not recommended
40 GB	256K*	64K	64K	Not recommended
48 GB	256K*	128K	64K	Not recommended
80 GB	256K*	128K	128K	64K
96 GB	256K*	128K	128K	128K

Notes:

- The 256K* values apply **only** to models that architecturally support a 256K context window (e.g., Ministral 3 14B). Other 14B models are capped at their native limit.
- Setting a high context length reserves KV cache memory upfront, even for short prompts. This means oversized context lengths waste VRAM without benefit on shorter emails.
- Performance drops sharply at very high context sizes due to KV cache growth. For batch-style email classification workloads in Aid4Mail, **32K–64K is typically the best balance of speed and VRAM efficiency**, unless your emails regularly include large attachments that require a larger context window.
- If you experience unexpectedly slow processing, try reducing the context length—this is often the single most effective tuning adjustment.

- "Not recommended" indicates that the model cannot be kept fully GPU-resident at any viable context length on that VRAM tier, meaning inference will partially offload to CPU and throughput will be severely degraded. The model may still run, but not at production-viable speeds.

Configuring Context Length in Aid4Mail

Aid4Mail needs to know the context length you've assigned in Ollama or LM Studio so that it doesn't send a payload that exceeds the model's configured limit. After setting an appropriate context length in your local inference tool, enter the same value in Aid4Mail:

1. Go to **App Settings > AI**.
2. Under **Provider configurations**, locate the corresponding provider (Ollama or LM Studio).
3. Enter the context length as a full token count (raw numbers only, no separators).

When converting from the shorthand 'k' values used by Ollama, multiply by 1,024:

Ollama Value	Aid4Mail Value
8k	8192
16k	16384
32k	32768
64k	65536
128k	131072
256k	262144

Valid values range from 8192 to 262144. We don't recommend values below 32768—smaller context windows significantly limit the amount of email content (especially attachments) that can be analyzed in a single request.

11.3. Configuring Ollama in Aid4Mail

With Ollama installed and its server running (see the *Provider and Model Selection Guide*, Section 4.7, for installation and setup):

1. In Aid4Mail's **App Settings > AI** tab, locate the **Ollama** provider and click **Configure…**.
2. Tick the **Available** checkbox.
3. Verify the endpoint URL is `http://127.0.0.1:11434` (the default). Update it if you've configured Ollama to use a different port or are connecting to a remote server.
4. Enter the **Context Length** to match your Ollama `num_ctx` setting (see Section 11.2).
5. In **Project Settings > AI**, select your preferred Ollama model (e.g., **Mistral Small 3.2 24B (Ollama)**) for the task you want to run (Filter, Classify, or Analyze).

Important: Ollama's server must be running before you start processing emails. Run `ollama serve` in a terminal if it isn't already running. VRAM requirements depend on the model—if VRAM is insufficient, Ollama falls back to CPU inference, which is significantly slower.

11.4. Configuring LM Studio in Aid4Mail

With LM Studio installed, a model loaded, and its server started (see the *Provider and Model Selection Guide*, Section 4.8, for installation and setup):

1. In Aid4Mail's **App Settings > AI** tab, locate the **LM Studio** provider and click **Configure…**.
2. Tick the **Available** checkbox.
3. Verify the endpoint URL is `http://127.0.0.1:1234` (the default). Update it if you've configured LM Studio to use a different port.
4. Enter the **Context Length** to match the context length configured in LM Studio (see Section 11.2).
5. In **Project Settings > AI**, select **Local Model (LM Studio)** for your task.

Important: LM Studio serves whichever model is currently loaded. If you change the model in LM Studio, Aid4Mail automatically uses the new model on subsequent requests—no Aid4Mail configuration change is needed. Remember to update the context length in both LM Studio and Aid4Mail when switching models, as different models may support different maximum context sizes.

---

12. system_prompts.json Reference

The `system_prompts.json` file in the root of the `AI Config` folder contains the system-level instructions that Aid4Mail prepends to every AI request. These instructions define the behavior for each task type and are referenced by the `<fixed_instructions>` and `<validate_instructions>` placeholders in model configurations.

The file defines four task types, each with two instruction sets:

Task Type	Key	Description
Analyze	`analyze.prompt_instructions`	System prompt for email analysis tasks (summarization, translation, extraction)
	`analyze.validate_instructions`	System prompt for validating analysis prompts
Classify (Enumerated)	`classify_enum.prompt_instructions`	System prompt for classification with a predefined list of categories
	`classify_enum.validate_instructions`	System prompt for validating enumerated classification prompts
Classify (Open-Ended)	`classify_open_ended.prompt_instructions`	System prompt for classification where the model determines the label

	`classify_open_ended.validate_instructions`	System prompt for validating open-ended classification prompts
Filter	`filter.prompt_instructions`	System prompt for Boolean email filtering (True/False)
	`filter.validate_instructions`	System prompt for validating filter prompts

These instructions are not intended for user modification under normal circumstances. They define Aid4Mail's core AI behavior—constraining responses to structured formats, requiring plain-text output without Markdown, and ensuring the model analyzes only the provided email content.

Date of publication: May 15, 2026.