LocalAI Package

Privacy-First On-Device AI for Automation Anywhere

Agentic App Store Package Documentation

Version 2.12.78

Micah Smith — Automation Anywhere

July 2026

1. INTRODUCTION

1.1 Overview

LocalAI is an Automation Anywhere package that brings real AI capabilities directly to your bot agents — with zero cloud dependency. Using llama.cpp, it runs quantized Small Language Models (SLMs) entirely on the local device: no internet connection required, no API keys, no data leaving your organization.

You configure a task in plain terms — classify this email, extract these fields from this invoice, summarize this support ticket — and the model runs locally, returns a structured Dictionary result, and stays loaded in memory for fast subsequent calls.

The package ships with seven curated models, all under 3.5GB, selected for the best quality-to-size ratio. Download happens once, automatically, the first time a model is used.

1.2 Key Capabilities

Capability	Description
On-Device Inference	All AI processing runs on the Bot Agent machine. No external API calls.
Zero Configuration	No API keys, no credentials, no cloud accounts required
Seven Curated Models	Pre-configured models from 1.1GB to 3.1GB, auto-downloaded on first use
Structured Output	Every action returns a Dictionary — use {category}, {summary}, {json} directly downstream
Privacy by Design	Data never leaves the device. Suitable for GDPR, HIPAA, and high-security environments
Cross-Platform	Runs on Windows (x64) and macOS (Intel + Apple Silicon). CPU-only — no GPU required
Memory-Efficient	Models stay loaded between calls in the same session for faster subsequent inference

1.3 Use Cases

- Invoice & document processing — extract vendor, amount, due date, and PO number without OCR templates
- Email triage — classify incoming messages as urgent/normal/spam before routing

- Support ticket summarization — condense long ticket threads before escalation
- PII redaction — scrub names, emails, SSNs before sending text to external systems
- Data normalization — standardize inconsistent phone numbers, dates, addresses from legacy systems
- JSON repair — fix malformed API payloads before sending to downstream services
- Custom prompting — any free-form text generation task not covered by the specialized actions

2. REQUIREMENTS & PREREQUISITES

2.1 Platform Support

Platform	Supported	Notes
Windows 10/11 (x64)	Yes	Bot Agent 20.x or later
macOS (Apple Silicon ARM64)	Yes	Native ARM binary — no Rosetta needed
Linux	No	Not supported in this release

2.2 System Requirements

Requirement	Minimum	Recommended
RAM	8GB	16GB
Free Disk Space	4GB+	10GB (for multiple models)
CPU	Any modern x64 or ARM64	8+ cores for faster inference
Automation Anywhere	Control Room v31+	v32+
Bot Agent	v20.11+	Latest

Note: No GPU is required. All inference runs on CPU. A modern processor with 8+ cores will achieve inference speeds of roughly 5–20 tokens/second.

2.3 Available Models

Model ID	Display Name	Size	Context	Best For
qwen3-4b	Qwen3 4B (Q4_K_M)	~2.5GB	32K	Structured output, classification, extraction
llama3.2-3b	Llama 3.2 3B (Q4_K_M)	~2.0GB	8K	Fast baseline, general tasks
phi4-mini	Phi-4 Mini (Q4_K_M)	~2.5GB	128K	Instructions, reasoning, long documents

gemma3-4b	Gemma 3 4B (Q4_K_M)	~2.5GB	128K	Balanced all-rounder
gemma4-e2b	Gemma 4 E2B (Q4_K_M)	~3.1GB	128K	Highest quality output
deepseek-r1-1.5b	DeepSeek R1 1.5B (Q4_K_M)	~1.1GB	128K	Fastest inference, chain-of-thought
qwen2.5-coder-3b	Qwen2.5-Coder 3B (Q4_K_M)	~1.9GB	32K	Code generation & scripting tasks

Models are stored in:

- Windows: C:\Users\{username}\AppData\Local\AutomationAnywhere\LocalAI\
- macOS: /Users/{username}/localAI/

3. GETTING STARTED

3.1 Quick Start

1. Download the LocalAI package from the Agentic App Store
2. Navigate to Control Room → Manage → Packages → Add Package and upload the JAR file
3. The package will appear as "LocalAI" in your package list
4. In a new TaskBot, add a "Validate Device" action and select your preferred model — this downloads it (~1–15 minutes on first run)
5. Add any of the AI actions (Classify Text, Extract Data, Summarize Text, etc.) and map the Dictionary output to your bot variables
6. Run the bot

3.2 Installation Steps

Step 1: Upload the Package

7. Navigate to Control Room → Manage → Packages
8. Click Add Package and upload LocalAI-2.12.78.jar
9. The package will appear as LocalAI in your package list with version 2.12.78

Step 2: Pre-Download a Model (Recommended)

Run the Validate Device action once in an initialization bot before using any AI actions. This ensures the model is downloaded before your production bot needs it. A 2.5GB model takes 5–15 minutes on a typical connection; subsequent runs load from disk in under 30 seconds.

Step 3: Use AI Actions in Your Bot

Drag any LocalAI action into your TaskBot, configure parameters, assign the Dictionary output to a variable, and reference keys like {myResult.category} or {myResult.summary} in downstream steps.

4. ACTIONS REFERENCE

LocalAI provides ten actions grouped into three categories:

- AI Inference Actions (use a model): Classify Text, Extract Data, Summarize Text, Redact PII, Normalize and Standardize, Transform to JSON, Sanitize JSON, Prompt Model
- Device Management Actions (no model inference): Validate Device, Delete Model

All inference actions return a Dictionary with at minimum: status ("success"/"error"), message (timing and model info), and model (model ID used), plus action-specific keys documented below.

4.1 Validate Device

Downloads or validates that a selected Small Language Model is available on the local device. Run this action once — in an initialization bot or at the start of a workflow — before using any inference actions. If the model already exists locally and passes a file-size integrity check, it returns immediately. If not, it downloads the model from HuggingFace automatically.

Parameters

Parameter	Type	Required	Default	Description
Model	Select	Yes	qwen3-4b	The model to download/validate. All 7 models available.

Return Values

Key	Type	Description
supported	String	"true" if the device supports the model
model_name	String	The model ID (e.g., "qwen3-4b")
model_path	String	Full path to the model file on disk
model_size_mb	String	Actual file size in MB after download
status	String	"success" or "error"
message	String	Timing info and model details

Example Use Case

Place Validate Device at the start of a daily invoice processing bot. Set Model to "phi4-mini". On the first run, the ~2.5GB model downloads automatically (~10 minutes). On every subsequent run,

validation completes in under 2 seconds. Downstream Extract Data actions are then guaranteed to have the model available without any delay.

4.2 Delete Model

Unloads a model from memory (if currently loaded) and deletes its files from local storage. Use this to reclaim disk space when a model is no longer needed. Returns a Boolean true on success.

Note: Delete Model returns a Boolean, not a Dictionary. It is the only action in the package with this return type.

Parameters

Parameter	Type	Required	Default	Description
Model	Select	Yes	qwen3-4b	The model to remove from disk

Return Value

Boolean — true if the model was deleted or did not exist; throws an error if deletion fails.

Example Use Case

In a quarterly cleanup bot, loop through models no longer in use and call Delete Model for each. Each deletion frees 1.1–3.1GB of disk space. The action is safe to call even if the model is not downloaded — it returns true immediately.

4.3 Classify Text

Categorizes input text into one of a set of user-defined categories using an on-device SLM. The categories are defined at design time as a comma-separated list. Optionally returns a confidence score (0.0–1.0) and a brief explanation of the classification reasoning.

Parameters

Parameter	Type	Required	Default	Description
Input Text	Text Area	Yes	—	The text to classify
Categories	Text	Yes	—	Comma-separated category list (e.g., "urgent, normal, spam")
Model	Select	Yes	qwen3-4b	SLM to use for classification
Include Confidence Score	Checkbox	No	false	Adds a "confidence" key to the output (0.0–1.0)
Include Explanation	Checkbox	No	false	Adds an "explanation" key with a brief reasoning string

Timeout (seconds)	Number	Yes	30	Maximum wait time before throwing a timeout error

Return Values

Key	Type	Description
category	String	The matched category name (exactly as provided in Categories)
confidence	String	Confidence score 0.0–1.0. Only present if Include Confidence Score is enabled.
explanation	String	Brief reasoning for the classification. Only present if Include Explanation is enabled.
status	String	"success" or "error"
message	String	Timing info and model used
model	String	Model ID used for this call

Example Use Case

An HR bot receives incoming employee emails. Classify Text is configured with categories "payroll inquiry, benefits question, IT request, general feedback". The returned {result.category} value routes the email to the correct queue via a Switch action. With Include Confidence Score enabled, emails below 0.7 confidence are flagged for human review rather than auto-routed.

4.4 Extract Data

Pulls specific named fields from unstructured text without requiring regex, templates, or model training. You define what to extract using a simple "fieldName: description" format — one field per line. The action returns a Dictionary where each key is a field name you defined, making extracted values immediately usable in downstream bot steps. If a field is not found in the source text, its value is set to "NOT_FOUND".

Parameters

Parameter	Type	Required	Default	Description
Input Text	Text Area	Yes	—	The document, email, invoice, or any unstructured text to extract from
Fields To Extract	Text Area	Yes	—	One field per line: "fieldName: description of what to find". Example: "invoice_number: the invoice ID or reference number"
Model	Select	Yes	qwen3-4b	SLM to use for extraction
Timeout (seconds)	Number	Yes	60	Maximum wait time. Use a higher value for longer documents.

Return Values

Key	Type	Description
{fieldName}	String	One key per field defined in Fields To Extract. Value is the extracted text, or "NOT_FOUND" if absent.
status	String	"success" or "error"
message	String	Timing info and model used
model	String	Model ID used for this call

Example Use Case

A finance bot processes incoming PDF invoices (already converted to text). Extract Data is configured with the following fields:

invoice_number: the invoice ID or reference number

vendor_name: the name of the company issuing the invoice

total_amount: the total amount due including taxes

due_date: the payment due date

po_number: the purchase order number if present

The bot uses {result.invoice_number}, {result.vendor_name}, {result.total_amount} to populate fields in the ERP system. Fields where the value is "NOT_FOUND" trigger an exception workflow for manual review.

4.5 Summarize Text

Condenses long documents, email threads, or support tickets into a concise summary. Three summary lengths are available: a single sentence, 2–3 sentences, or a full paragraph. An optional Focus Area parameter narrows the summary to a specific aspect (e.g., "action items", "financial impact", "technical root cause").

Parameters

Parameter	Type	Required	Default	Description
Input Text	Text Area	Yes	—	The text to summarize
Summary Length	Select	Yes	short	One sentence / Short (2–3 sentences) / Detailed (full paragraph)
Focus Area	Text	No	—	Optional: focus the summary on a specific aspect of the text
Model	Select	Yes	qwen3-4b	SLM to use for summarization
Timeout (seconds)	Number	Yes	45	Maximum wait time

Return Values

Key	Type	Description
summary	String	The generated summary text
word_count	String	Word count of the summary
status	String	"success" or "error"
message	String	Timing info and model used

model	String	Model ID used for this call

Example Use Case

A customer support bot retrieves open ticket threads before escalating to Tier 2. Summarize Text is set to "Short (2–3 sentences)" with Focus Area = "technical issues and steps already attempted". The {result.summary} is prepended to the escalation email, saving the Tier 2 agent from reading the full thread. For executive dashboards, the same action is reused with Summary Length = "One sentence" to populate a status column.

4.6 Redact PII

Replaces personally identifiable information (PII) in text with a configurable token before the text is logged, archived, or sent to any external system. Uses on-device AI for context-aware redaction that goes beyond simple regex — correctly handling names in varied formats, partial addresses, and phone numbers written in prose. Supports targeted redaction (names only, emails only, etc.) or full all-PII redaction.

Parameters

Parameter	Type	Required	Default	Description
Input Text	Text Area	Yes	—	The text to redact PII from
PII Types to Redact	Select	Yes	all	All PII / Names only / Email addresses only / Phone numbers only / SSNs only / Credit card numbers only / Physical addresses only / Dates of birth only
Replacement Token	Text	No	[REDACTED]	Text to substitute for each redacted item (e.g., ***, <PII>, [NAME])
Model	Select	Yes	qwen3-4b	SLM to use for redaction
Timeout (seconds)	Number	Yes	45	Maximum wait time

Return Values

Key	Type	Description
redacted_text	String	The original text with all matching PII replaced by the replacement token
items_redacted	String	Count of replacement tokens inserted (estimate of items redacted)
status	String	"success" or "error"
message	String	Timing info and model used
model	String	Model ID used for this call

Example Use Case

A compliance bot processes customer complaint emails before writing them to an audit log. Redact PII is configured with PII Types = "All PII" and Replacement Token = "[REDACTED]". The {result.redacted_text} is written to the log; the {result.items_redacted} count is stored as a metric. This satisfies GDPR Article 25 (data minimization) without requiring any changes to the logging infrastructure. For HIPAA workflows, a second pass uses PII Types = "Dates of birth" with Token = "[DOB]" to label redactions by type.

4.7 Normalize and Standardize

Converts inconsistently formatted data — phone numbers, dates, addresses, names, emails — into a target standard format. The model understands context and handles edge cases that regex-based approaches miss (e.g., international phone numbers, abbreviated month names, hyphenated surnames). If normalization fails and Preserve Original On Failure is enabled, the original input is returned unchanged rather than throwing an error.

Parameters

Parameter	Type	Required	Default	Description
Input Text	Text Area	Yes	—	The text to normalize
Data Type	Select	Yes	auto-detect	Phone Number / Date / Address / Name / Email / Auto-Detect / Custom
Output Format	Text	No	—	Target format description (e.g., "YYYY-MM-DD", "E.164", "digits only", or a custom instruction)
Model	Select	Yes	qwen3-4b	SLM to use
Preserve Original On Failure	Checkbox	No	true	Return original input instead of throwing an error if normalization fails
Timeout (seconds)	Number	Yes	30	Maximum wait time

Return Values

Key	Type	Description
result	String	The normalized/standardized text

original	String	The original input text (for comparison or fallback logic)
data_type	String	The data type used (as selected or detected)
status	String	"success" or "error"
message	String	Timing info and model used
model	String	Model ID used for this call

Example Use Case

A CRM integration bot reads contact records from a legacy system where phone numbers appear in inconsistent formats: "(555) 123-4567", "555.123.4567", "+1-555-123-4567". Normalize and Standardize is configured with Data Type = "Phone Number" and Output Format = "digits only". The {result.result} value is written directly to the CRM API field. Preserve Original On Failure is left enabled so records with unparseable values pass through rather than failing the bot.

4.8 Transform to JSON

Converts structured text in various formats — CSV, TSV, key-value pairs, tables, bullet lists — into valid JSON. Output can be a single object {} or an array of objects [{}], and can be compact or pretty-printed. The model identifies field names from the input and generates syntactically valid JSON validated by Gson before returning. If the model generates malformed JSON, an actionable error is thrown.

Parameters

Parameter	Type	Required	Default	Description
Input Text	Text Area	Yes	—	The text to convert to JSON (CSV, TSV, key-value, table, or list)
Input Format	Select	Yes	auto-detect	CSV / TSV (Tab-Separated) / Key-Value Pairs / Table / List/Bullet Points / Auto-Detect
Output Style	Select	Yes	compact	Compact (no whitespace) / Pretty (formatted)
Output Type	Select	Yes	object	Object (single {}) / Array (multiple [{}])
Model	Select	Yes	qwen3-4b	SLM to use
Timeout (seconds)	Number	Yes	30	Maximum wait time

Return Values

Key	Type	Description
json	String	A syntactically valid JSON string (object or array as selected)
status	String	"success" or "error"
message	String	Timing info and model used
model	String	Model ID used for this call

Example Use Case

A data pipeline bot receives daily export files in CSV format from a legacy application. Transform to JSON converts each row to a JSON object (Input Format = "CSV", Output Type = "Array") which is then posted to a REST API endpoint using an HTTP action. The {result.json} value is passed directly as the request body. For single-record key-value forms, Output Type = "Object" produces a clean JSON payload in one step.

4.9 Sanitize JSON

Repairs malformed JSON using an on-device SLM. Fixes the most common causes of JSON parse failures: unescaped double quotes inside string values, bare backslashes, literal newlines and tabs inside strings, and trailing commas. The output is validated with Gson before being returned — if the model cannot produce valid JSON, a clear error is thrown. Use this action when receiving JSON from legacy systems, user input, or other sources where the payload may not be reliably escaped.

Parameters

Parameter	Type	Required	Default	Description
Input JSON	Text Area	Yes	—	The malformed JSON text to repair
Output Style	Select	Yes	compact	Compact (no whitespace) / Pretty (formatted)
Model	Select	Yes	qwen3-4b	SLM to use (Qwen3 4B recommended for JSON tasks)
Timeout (seconds)	Number	Yes	30	Maximum wait time

Return Values

Key	Type	Description
sanitized_json	String	The repaired, syntactically valid JSON string
status	String	"success" or "error"
message	String	Timing info and model used
model	String	Model ID used for this call

Example Use Case

An integration bot receives webhook payloads from a third-party ticketing system. Occasionally the payloads contain ticket notes with unescaped quotes and Windows file paths with bare backslashes, causing the downstream JSON parse step to fail. Sanitize JSON is placed immediately after the HTTP receive step. The {result.sanitized_json} is passed to the JSON parse action, eliminating the parse failures without any changes to the upstream system.

4.10 Prompt Model

The free-form action. Sends any prompt directly to a selected SLM and returns the generated text. Use this for tasks not covered by the specialized actions — custom question answering, creative text generation, code generation, translation, or any other text task. Temperature controls output randomness: low values (0.0–0.3) produce focused, deterministic responses; high values (0.7–1.0) produce more varied, creative output.

Parameters

Parameter	Type	Required	Default	Description
Prompt	Text Area	Yes	—	The full prompt or instruction to send to the model
Model	Select	Yes	qwen3-4b	SLM to use for generation
Timeout (seconds)	Number	Yes	30	Maximum wait time
Temperature	Number	Yes	0.3	Randomness control: 0.0–1.0. Low = focused/deterministic. High = creative/varied.

Return Values

Key	Type	Description
response	String	The model’s generated text response
status	String	"success" or "error"
message	String	Timing info and model used
model	String	Model ID used for this call

Example Use Case

A vendor onboarding bot needs to generate a personalized welcome email for each new supplier. Prompt Model is configured with a template prompt like: "Write a professional 3-sentence welcome email to {vendor_name} at {vendor_email}. They supply {category} products. Mention our 30-day payment terms." Temperature is set to 0.6 for natural variation. The {result.response} is used as the email body in a Send Email action. Using Qwen2.5-Coder 3B, the same action can generate SQL queries or script fragments as part of a developer automation workflow.

5. MODEL SELECTION GUIDE

5.1 Model Comparison

Model	Size	Context	Speed	Quality	Best For
Qwen3 4B	~2.5GB	32K	Medium	High	Default choice. Structured output, classification, extraction, JSON
Llama 3.2 3B	~2.0GB	8K	Fast	Good	Fast baseline. Short texts, simple tasks, high-volume processing
Phi-4 Mini	~2.5GB	128K	Medium	High	Long documents, complex instructions, reasoning tasks
Gemma 3 4B	~2.5GB	128K	Medium	High	Balanced general-purpose model. Good across all task types
Gemma 4 E2B	~3.1GB	128K	Slower	Highest	Maximum quality output. Use when accuracy matters most
DeepSeek R1 1.5B	~1.1GB	128K	Fastest	Basic	Fastest inference. Simple tasks, very short outputs
Qwen2.5-Coder 3B	~1.9GB	32K	Fast	High (code)	Code generation, SQL, scripts, JSON/XML manipulation

5.2 Recommended Models by Action

Action	Recommended Model	Reason
Classify Text	Qwen3 4B	Best category matching on short to medium texts
Extract Data	Qwen3 4B	Structured field-value output
Summarize Text	Phi-4 Mini or Gemma 3 4B	Handles long documents; Phi-4 for technical content

Redact PII	Qwen3 4B	Consistent instruction following for redaction
Normalize and Standardize	Qwen3 4B or Llama 3.2 3B	Short output; Llama is faster for high volume
Transform to JSON	Qwen3 4B	Reliable JSON structure generation
Sanitize JSON	Qwen3 4B	Strong at understanding and repairing JSON
Prompt Model	Varies	Use Gemma 4 E2B for quality; DeepSeek R1 for speed; Coder 3B for code

6. PRIVACY & SECURITY

6.1 Privacy by Design

- All inference runs entirely on the local Bot Agent machine
- No text, prompt, or model output is transmitted over the network
- No API keys, user accounts, or cloud services are required
- Models are downloaded once from HuggingFace and stored locally; all subsequent inference is offline
- Suitable for air-gapped environments after initial model download

6.2 Data Handling

- Input text passed to actions is processed in memory and never written to disk
- Models are stored in the user's disk in GGUF format (Windows: %LOCALAPPDATA%\AutomationAnywhere\LocalAI\, macOS: ~/localAI/)
- No logging of prompt content or model responses by the package — standard Automation Anywhere audit logging applies
- The Delete Model action permanently removes all model files from disk

6.3 Compliance Considerations

Note: The following are informational points, not legal advice. Consult your compliance team for guidance specific to your organization.

- GDPR / Data Minimization: Redact PII before writing to logs or external systems
- HIPAA: Use Redact PII with targeted PII types to anonymize before archival
- High-Security Environments: Models can be pre-downloaded in a network-connected staging environment, then the model files copied to air-gapped Bot Agents manually

7. FAQS & TROUBLESHOOTING

7.1 Frequently Asked Questions

Q: Does this package require an internet connection?

A: Only for the initial model download. Once a model is downloaded to the Bot Agent, all inference runs offline. Use Validate Device to pre-download before deploying to production.

Q: How long does model loading take?

A: The first time a model is used in a session, loading takes 5–30 seconds depending on the model size and disk speed. Subsequent calls within the same session reuse the loaded model and are much faster. The model stays in memory until the bot session ends.

Q: How much RAM does inference use?

A: Approximately 1.5–2× the model size. A 2.5GB model typically uses 4–5GB of RAM during inference. 8GB system RAM is the minimum; 16GB is recommended for smooth operation.

Q: Can I run multiple models at the same time?

A: Yes. Models are cached in a shared ModelManager singleton. Each unique model you call is loaded once and stays in memory. Only one model is loaded at a time. Switching to a different model stops the current server and starts a new one (5–30 seconds). Plan for model-switch time in workflows that use more than one model.

Q: What happens if the model returns an unexpected response?

A: Structured actions (Classify Text, Extract Data, Transform to JSON, Sanitize JSON) validate the model output and throw a descriptive BotCommandException if it cannot be parsed. Free-form actions (Summarize Text, Redact PII, Prompt Model) return the cleaned text directly. Normalize and Standardize can optionally return the original input on failure (Preserve Original On Failure checkbox).

Q: Can I use this on both Windows and macOS?

A: Yes. The package bundles the official llama-server binary for Windows x64, macOS Intel (x86-64), and macOS Apple Silicon (ARM64). The correct binary is extracted automatically on first use. The same JAR file works on all three platforms.

Q: The bot times out during inference. What should I try?

A: Increase the Timeout parameter. A 2.5GB model on an 8-core CPU processes roughly 5–15 tokens/second. For longer outputs, increase the timeout proportionally. Alternatively, use a smaller/faster model (Llama 3.2 3B or DeepSeek R1 1.5B).

7.2 Troubleshooting

Issue	Solution
Model download fails or is interrupted	Re-run Validate Device. The downloader resumes from scratch (no partial resume). Ensure the Bot Agent has internet access and ~4GB free disk space.

"Unknown model type" error	Check that the modelName parameter exactly matches one of the 7 supported model IDs (e.g., "qwen3-4b", not "Qwen3"). Model IDs are case-insensitive but must match the listed values.
"Model returned empty response"	Increase Timeout. The model may have timed out mid-generation. For Classify Text, try a more capable model (Phi-4 Mini or Gemma 3 4B).
Classification returns wrong category	Rephrase category names to be more distinct. Avoid categories that are semantically similar. Enable Include Explanation to understand the model's reasoning.
"Generated invalid JSON" (Transform to JSON)	The model generated a structure that didn't match the Output Type. Switch between Object and Array, or try a different model.
Inference is very slow	Ensure no other memory-intensive applications are running. Use a smaller model. For Windows, check that the process is not swapping to disk (Task Manager → Memory).
OutOfMemoryError during model load	The system does not have enough available RAM. Close other applications or switch to a smaller model (DeepSeek R1 1.5B at ~1.1GB is the smallest option).

8. APPENDIX

8.1 Supported Model IDs

qwen3-4b, llama3.2-3b, phi4-mini, gemma3-4b, gemma4-e2b, deepseek-r1-1.5b, qwen2.5-coder-3b

8.2 Model Storage Structure

~/localAI/

qwen3-4b-q4/ Qwen3-4B-Q4_K_M.gguf

llama3.2-3b-q4/ Llama-3.2-3B-Instruct-Q4_K_M.gguf

phi4-mini-q4/ microsoft_Phi-4-mini-instruct-Q4_K_M.gguf

gemma3-4b-q4/ gemma-3-4b-it-Q4_K_M.gguf

gemma4-e2b-q4/ gemma-4-E2B-it-Q4_K_M.gguf

deepseek-r1-1.5b-q4/ DeepSeek-R1-Distill-Qwen-1.5B-Q4_K_M.gguf

qwen2.5-coder-3b-q4/ Qwen2.5-Coder-3B-Instruct-Q4_K_M.gguf

8.3 Dependencies

Library	Version	Purpose
llama-server (llama.cpp)	b9481	Bundled llama-server binary for local model inference (Windows x64, macOS ARM64/x86-64). Supports all model architectures; replaces the JNI approach.
OkHttp (com.squareup.okhttp3:okhttp)	4.12.0	Model download with progress tracking
Gson (com.google.code.gson:gson)	2.10.1	JSON validation and formatting

8.4 Record of Changes

Version	Date	Changes
2.12.78	June 2026	Grammar-constrained JSON infrastructure (JsonGrammar/GBNF); empty output guard for TransformToJSON; 8 new unit tests for parsing layer
2.12.74	June 2026	Security hardening: per-session API key auth, race condition fix in complete(), zip slip protection, temp file cleanup, Gson-based JSON parsing

2.12.67	June 2026	Replaced JNI inference (de.kherud:llama) with llama-server subprocess + HTTP API; bundled llama-server binaries in JAR for Windows x64, macOS ARM64/x86-64; supports all architectures including qwen3, gemma4
2.12.28	May 2026	SanitizeJSON migrated to model-based inference; added Qwen2.5-Coder 3B as 7th model; model labels standardized; all actions return DICTIONARY with return_sub_type STRING
2.12.17	May 2026	Added ExtractData, SummarizeText, RedactPII actions; DictionaryHelper standardized return format; all actions migrated to Dictionary return
2.12.1	May 2026	Initial public release. ClassifyText, NormalizeAndStandardize, TransformToJSON, SanitizeJSON, Prompt, ValidateDevice, DeleteModel

8.5 License

LocalAI is released under the Apache License 2.0. You are free to use, modify, and distribute this package.

GGUF model weights are subject to their respective model licenses on HuggingFace. The llama.cpp project (llama-server binary) is licensed under the MIT License.