EEAI Software Guide

The on-device assistant layer inside EEOS

EEAI is the local assistant runtime inside the EEOS stack. It is built for EESystem centers that need a trained interface sitting on top of the operating layer, the Vibe scoring path, and the center's local knowledge.

It is not a cloud chatbot bolted onto a center later. In the `center-ai` deployment, EEAI ships as part of the stack, loads a local model into RAM, serves a local endpoint, opens a browser-facing assistant, and stays inside the room with the rest of the system.

What EEAI does

EEAI gives the operator a fast way to work with the stack while the room is live.

- Explains Vibe session output in center-safe wellness language
- Interprets score bands, quality flags, and before-during-after shifts
- Helps the operator understand what changed in a session
- Guides runtime support by checking evidence before suggesting action
- Carries center memory through local docs and the Vibe knowledgebase

Without EEAI, the stack can measure. With EEAI, the stack can explain.

Where EEAI sits in the stack

The EEOS stack is split into four layers:

- **EEOS**: the operating layer, clean boot, RAM-first runtime, stable center environment
- **Vibe**: the biometric layer, session scoring, HRV interpretation, longitudinal session record
- **ContextOS**: the local memory layer, center documents, retrieval, tuned center context
- **EEAI**: the trained assistant layer, live operator support, session interpretation, local reasoning

EEAI is the front end of that stack from the operator's point of view. It is where the score, the memory, and the runtime state get turned into something usable in the moment.

Runtime architecture

The shipped runtime is local by design.

CORE PATHS AND SETTINGS

Setting	Value
Model path	`/opt/eeos-ai/models/default.gguf`
Inference engine	`/opt/eeos-ai/bin/llama-server`
Local host	`127.0.0.1`
Local port	`7333`
Local API	`http://127.0.0.1:7333/v1`
Context window	`4096`
Threads	`0` (auto-detect)
GPU layers	`0` (CPU-only by default)
System prompt path	`/opt/eeos-ai/system-prompt.txt`
Config file	`/etc/eeos/ai.conf`

The quantized model is expected to ship in GGUF format. The README specifies a 4-bit quantized model sized to fit on the system and load into RAM cleanly.

WHY THE LOCAL RUNTIME MATTERS

- The operator is not waiting on a remote API to answer
- Center data does not have to leave the room for normal operation
- The assistant can work against the same machine state the operator is looking at
- The runtime stays aligned with the broader EEOS discipline of clean, predictable behavior

The `center-ai` deployment profile

EEAI is already accounted for in the profile system. The `center-ai` profile is the reference deployment for an AI-enabled center.

PROFILE BEHAVIOR

Profile field	Value
Profile name	`EEOS AI Center`
Desktop	`full`
Wallpaper	`eeos_default`

Profile field	Value
HHFE autostart	`yes`
HHFE mode	`desktop`
EEAI assistant	`yes`
Boot entries	`reduced`
Default boot	`EEOS 1.0 – Copy SFS to RAM`
Diagnostics	`yes`
Remote agent	`optional`
Kiosk	`available`
Vitals	`optional`
AI assistant feature	`yes`

The important design choice is the default boot mode. The `center-ai` profile defaults to copying the runtime into RAM. That keeps the assistant and the operating layer living at memory speed and avoids unnecessary disk churn while the room is active.

What EEAI knows

EEAI is not meant to guess. It is grounded in the local prompt, the support rules, and the Vibe knowledgebase.

BUILT-IN DOMAINS

- Vibe session interpretation
- Wellness-safe operator language
- Vibe score band explanation
- Quality flag handling
- Session phase explanation
- Support and diagnostics workflow
- Center-specific documents loaded into local context

VIBE LANGUAGE IT ALREADY UNDERSTANDS

The knowledgebase maps internal metrics into operator-facing or client-facing language:

Internal metric	Display language
SDNN	Heart Rhythm Variability
RMSSD	Relaxation Depth
LF Power	Active Balance
HF Power	Rest Response
Composite score	VIBE Score

EEAI also knows the session structure:

- **Pre**: seated baseline before HHFE starts
- **During**: rolling session measurement window while HHFE is active
- **Post**: immediate carryover after HHFE stops

It can explain score bands, basic shifts, and quality flags while staying inside wellness framing.

Support behavior

The local Go runtime bakes in hard rules for operator support. The assistant is supposed to verify first, then answer.

NON-NEGOTIABLE SUPPORT RULES

- Never suggest a fix before verifying the problem exists
- Never delete, remove, or overwrite files
- If diagnostics say the system is fine, present the evidence instead of making up a repair story
- If restart advice keeps failing, stop and escalate to a support report
- Never touch SFS files, session saves, `INVOKE.TXT`, or boot configuration
- Never give medical advice
- Use `VIBE Score` as the client-facing score name

FIRST RESPONSE PATH FOR RUNTIME ISSUES

When an operator reports a problem, EEAI is supposed to point them to the desktop support workflow first:

> Click the EEOS Support icon on your desktop. If you have a USB drive, plug it in first. The report saves automatically.

That matters because it gets the diagnostic evidence before the assistant starts guessing about repair steps.

Wellness boundaries

EEAI is a wellness technology assistant, not a clinical system.

WHAT EEAI CAN DO

- Explain session wellness data
- Describe autonomic trends in non-clinical language
- Help the operator understand the session record
- Guide support and diagnostics on the machine itself

WHAT EEAI CANNOT DO

- Diagnose conditions
- Prescribe or recommend treatment
- Make disease claims
- Suggest medication changes
- Present the platform as a medical device

The system prompt is explicit here: use wellness language such as relaxation, recovery, vitality, and autonomic balance. If someone asks a medical question, EEAI should send them to a healthcare provider.

Operator workflows

SESSION INTERPRETATION

The operator can use EEAI to:

- explain what a Vibe score means in plain language
- compare before, during, and after phase changes
- explain why a score was moderate, strong, flat, or mixed
- describe quality flags without creating fear

RUNTIME SUPPORT

The operator can use EEAI to:

- identify the next read-only check
- explain what a local service is doing
- surface the right support report path
- walk through known operator tasks with visible commands

CENTER-SPECIFIC BEHAVIOR

Because the assistant runs with local context, it can be tuned around:

- center policies
- center documents
- operator language
- local troubleshooting notes
- Vibe interpretation guidance specific to that site

Why this layer matters

If the intelligence layer lives outside the stack, every answer arrives stripped of local context. EEAI fixes that by staying on the machine with the operating system, the biometric layer, and the center's own memory.

That gives the center:

- faster answers
- less operator friction
- less context loss
- fewer blind support loops
- a direct way to explain what happened in the room

Source files behind this guide

This guide is grounded in the shipped source and docs:

- `v2/ai-assistant/README.md`
- `v2/ai-assistant/eeos-ai.conf`
- `v2/ai-assistant/system-prompt.txt`
- `v2/ai-assistant/eeai/main.go`
- `profiles/center-ai.conf`
- `docs/PROFILES.md`
- `v2/vitals/EEAI_VITALS_KNOWLEDGEBASE.md`

Closing note

EEAI is already part of the stack. The model path is real. The local endpoint is real. The profile is real. The support rules are real. The Vibe interpretation layer is real.

It has its own page because it is its own product layer, not because it is a future idea.