Local AI - First Install Walkthrough

FrameworX 10.1.5 ships configured to talk to an Ollama endpoint on your own network — always a separate machine from the FrameworX Server (installing the model on the FrameworX host is against company recommendation, for production and testing alike) — with `qwen2.5:7b-instruct` as the recommended default model (~4.7 GB), the best balance of reasoning and reliable JSON tool-call output. It expects a machine with a GPU. This walkthrough covers a first Local AI setup end to end — install Ollama, pull the model, point FrameworX at the endpoint, and verify — plus what to expect on disk, time, and latency. FrameworX ships no local installer; you install Ollama yourself, and the per-OS commands (Windows, macOS, Linux) are on Local AI - Installing Models (Windows, macOS, Linux). On limited hardware with no GPU, switch the solution's `SolutionCapabilities[LocalAI].Settings.Name` field to the `qwen2.5:3b-instruct` fallback — see the **Switching the model** section below. Everything runs on your own network with no internet connection.

AI Integration Local AI First Install Walkthrough

Quick start

Three steps, on the machine that will serve the model — always a separate machine from the FrameworX Server, even for a demo (see Remote and Cloud LLM Models). The per-OS install detail (Windows, macOS, Linux) is on Local AI - Installing Models (Windows, macOS, Linux).

```
# 1. Install Ollama (Windows shown; see Installing Models for macOS / Linux)
winget install Ollama.Ollama

# 2. Pull the recommended default model (~4.7 GB)
ollama pull qwen2.5:7b-instruct

# 3. Confirm Ollama is serving and the model is present
ollama list
curl http://localhost:11434/v1/models
```

Then point your solution at it in Designer — see **Next: configure your model in Designer** below. On no-GPU hardware, pull `qwen2.5:3b-instruct` instead.

The recommended model: qwen2.5:7b-instruct

Item	Value
Name	`qwen2.5:7b-instruct`
License	Apache 2.0 — commercial use permitted, no royalty, no per-seat fee.
Size on disk	~4.7 GB (quantized Q4_K_M; pulled into `%USERPROFILE%\.ollama\models\`)
Why this model	FrameworX recommends `qwen2.5:7b-instruct` because the Local AI surface depends on the model returning reliable JSON tool-call output — for the **ChatRequest** action's tool catalog (tag reads, alarm queries, historian queries, solution-authored MCP Tools) and for the structured reply envelope every script consumer parses. Among permissively licensed local models in the 4–5 GB footprint range, the 7B-instruct tier is where reasoning quality and tool-call reliability become consistent enough for production use; below that (the 3B tier) tool-call malforms are common, and the larger `qwen2.5:32b-instruct` tier raises disk, RAM, and GPU cost steeply — reserve it for dedicated GPU hosts where maximum reasoning is worth it.

Hardware expectations

The 7B model runs on standard server or workstation hardware — always a separate machine from the FrameworX Server; a GPU is expected for a responsive interactive-chat experience (Ollama auto-detects and uses it). It still runs on CPU, but slower — CPU-only model hosts should prefer the 3B fallback for atomic tasks.

Resource	Recommended	Notes
Disk	~6.5 GB free	1.8 GB Ollama runtime + 4.7 GB model. Plan for ~10 GB if you expect to keep both the 3B (fallback) and 7B (default) variants installed.
RAM	16 GB or more	The model occupies ~5 GB while loaded — on the dedicated model host, never the FrameworX Server. Because the model runs on its own machine, the SCADA stack (TServer, Designer, plant Displays) never competes with it for memory. Smaller model hosts (8 GB) work for evaluation.
CPU	Modern x64	Any current Intel / AMD desktop or server CPU runs the 7B model. First call after load takes ~10–15 seconds; subsequent calls land in ~500 ms on a typical workstation.

GPU	Recommended	Expected for a responsive 7B interactive-chat experience. Ollama auto-detects an NVIDIA or AMD GPU and offloads layers transparently — no configuration needed; per-call latency drops by roughly 3–10×. The model still runs on CPU if no GPU is present, but slower.

What a first install involves

Step	Action	Already done if
1	Check port 11434 is free of any conflicting service	port is already held by Ollama
2	Install Ollama (`winget install Ollama.Ollama`, or the installer from `ollama.com/download`)	`ollama.exe` already on disk
3	Make sure Ollama is serving — it runs as a background service; `ollama serve` if not	endpoint at `http://localhost:11434` already responds (checked on the Ollama host, not the FrameworX server)
4	Pull `qwen2.5:7b-instruct` (~4.7 GB)	model already in `~/.ollama/models/`
5	Verify inference end to end (the Designer status indicator, or a `curl` to `/v1/models`)	—

None of these steps is destructive; re-doing them on an already-set-up machine is safe and changes nothing.

What to expect on first install

Item	Value
Total disk usage	**~6.5 GB** (1.8 GB Ollama runtime in `%LOCALAPPDATA%\Programs\Ollama\` + 4.36 GB model in `%USERPROFILE%\.ollama\`)
First-run time	~5 minutes on a 50 MB/s connection (1.8 GB Ollama installer + 4.7 GB model pull). Slower connections scale linearly.
Permissions required	None. Ollama installs per-user — no UAC / admin elevation needed.
First chat latency	**~15 seconds.** The model loads from disk into RAM on first call after startup or after the keep-alive window expires.
Subsequent chat latency	**~500 milliseconds** on a typical CPU; faster with a GPU.
Keep-alive	Default 5 minutes. Idle longer than that and the next call pays cold-load again. Set `OLLAMA_KEEP_ALIVE=24h` in the environment to keep the model resident.

Verify it works

Two quick checks confirm the runtime is ready before you wire up the solution. Run these on the **Ollama host** (the model machine), not the FrameworX server. First, that Ollama is serving and the model is present:

```
# Models on disk
> ollama list
NAME                    ID              SIZE     MODIFIED
qwen2.5:7b-instruct     a1b2c3d4e5f6    4.7 GB   2 minutes ago

# Endpoint reachable (lists the served models as JSON)
> curl http://localhost:11434/v1/models
{"object":"list","data":[{"id":"qwen2.5:7b-instruct","object":"model"}]}
```

Second, the end-to-end path FrameworX actually uses: the **Status** indicator on the AI Engine tile in **Solution Capabilities** turns **Reachable** (green), and a **ChatRequest** from any Display panel returns a reply envelope with `status = "ok"` and a populated `text` field. Any failure there points at the same short list of causes: Ollama not started, model not pulled, or the port held by another process.

Next: configure your model in Designer

With Ollama serving and the model pulled, open Designer and connect the solution:

1. Open your solution in Designer.
2. Navigate to **Solution Capabilities AI Engine**.

3. Tick **Enable Local AI** to flip `SolutionCapabilities[LocalAI].Enabled` to `true`. The status indicator should resolve to **Reachable** within a few seconds.
4. Click **Edit Configuration** to set the endpoint URL, model name, response timeout, or authorization — or to point at a remote / cloud LLM instead of the default local Ollama.

If the indicator stays on **Unreachable**, confirm Ollama is up on the endpoint host (`ollama list` and `curl http://localhost:11434/v1/models` on that machine) — the usual causes are Ollama not started, the model not pulled, or the port held by another process. If it shows **Auth required**, see SecuritySecrets Authentication for Local AI. Full configuration reference: Local AI Configuration Reference.

If port 11434 is already in use

If port 11434 is held by **a different process** (LM Studio, llama.cpp server, oobabooga, an old test server, etc.) rather than Ollama, Ollama cannot serve on it. Identify the holder first:

```
# Windows: which PID holds 11434
netstat -ano | findstr 11434
```

1. Stop the conflicting service, or move it to a different port.
2. Start Ollama — it claims 11434 once the port is free.

Do not blindly kill whatever holds the port — 11434 is heavily used by the LLM ecosystem, and a silent process kill is the wrong default. Confirm what it is first.

Running the model on a different host

By default Ollama binds **localhost only**. To run Ollama on a separate machine (typically a GPU server) and have FrameworX talk to it over the network:

1. On the Ollama host: set `OLLAMA_HOST=0.0.0.0:11434` in the system environment, then restart Ollama.
2. Open inbound TCP 11434 in the Ollama host's firewall.
3. In FrameworX, edit `SolutionCapabilities[LocalAI].Settings` to point `URL` at `http://<ollama-host-ip>:11434/v1/chat/completions`.

Authentication note. Ollama has no built-in authentication. For deployments beyond a trusted LAN, restrict the firewall rule to the FX server's IP, or front port 11434 with a reverse proxy that adds an API key — then set the FX `Authorization` field on the LocalAI capability to `BearerToken` with that key. Do not expose port 11434 directly to an untrusted network.

Switching the model

> **Recommended default and limited-hardware fallback**
>
> FrameworX recommends `qwen2.5:7b-instruct` as the default (best reasoning and tool-call reliability); it expects a GPU. On a machine with no GPU, use the `qwen2.5:3b-instruct` fallback (~2 GB) — lower speed and quality, suited to atomic tasks rather than interactive chat. For maximum reasoning on a strong GPU, `qwen2.5:32b-instruct` is the performance tier.

Switching any solution between models is a two-step change. The common cases: moving to the `qwen2.5:3b-instruct` fallback on a no-GPU machine, or moving an older copy that was configured for 3B (some demo solutions used to ship configured for 3B for no-GPU evaluation — that path is no longer recommended: CPU-only inference produces only ~2–4 tokens/sec and is too slow even to evaluate; run the model on a separate GPU machine instead) up to the recommended `qwen2.5:7b-instruct`.

1. **Pull the target model** with `ollama pull <name>` — for example `ollama pull qwen2.5:7b-instruct`. Models already on disk are not removed; multiple tiers coexist.
2. **Update the solution's Local AI configuration.** In Designer, open **Solution Capabilities AI Engine**, click **Edit Configuration**, and set the `Name` field to the target model. Save. The next chat or `AI.Execute` call uses the new model — no restart required.

To confirm the switch took effect: ask the chat panel any question and inspect the reply. The `status` field on the reply JSON envelope reports `ok` and the `latencyMs` field will be slightly higher than with 3B (the 7B model takes longer per call — usually ~500 ms vs. ~200 ms on the same CPU). Reply quality — specifically multi-step reasoning, tool selection, and JSON tool-call formatting — improves visibly.

Choosing a different model entirely

The default `qwen2.5:7b-instruct` is a balance of quality and footprint for typical SCADA hardware. To use a different model:

1. Pull it with Ollama: `ollama pull <model-name>` (for example, `qwen2.5:3b-instruct` for low-RAM / no-GPU machines, `qwen2.5:32b-instruct` for strong-GPU servers, or another OpenAI-compatible model your hardware supports).
2. In FrameworX, edit `SolutionCapabilities[LocalAI].Settings` and set the `Name` field to the new model name.

Any OpenAI-compatible endpoint works — including cloud LLMs (OpenAI, Azure OpenAI, Anthropic via OpenAI-compat proxy). Set `URL` and `Authorization` accordingly.

See also

- Local AI - Installing Models (Windows, macOS, Linux) — per-OS Ollama install and model-pull commands.
- Local AI — the main reference page (parent of this one).
- AI Integration — the broader AI surface in FrameworX.

In this section...