Local AI - First Install Walkthrough

FrameworX 10.1.5 ships configured to talk to a local Ollama with `qwen2.5:7b-instruct` by default. This page documents **Install-LocalAI.ps1** — the idempotent setup script that gets a fresh machine into that state.

Quick start

Run **Install-LocalAI.ps1** from the FrameworX `AISetup` folder. The script is idempotent: re-running on an already-set-up machine is safe and finishes in a few seconds.

```
powershell -ExecutionPolicy Bypass -File "<FX-install>\AISetup\Install-LocalAI.ps1"
```

Replace `<FX-install>` with your FrameworX install location, typically `C:\Program Files\Tatsoft\FrameworX\fx-10`.

What the script does

Step	Action	Skipped if
1	Pre-checks port 11434 for any conflicting service	port is already held by Ollama
2	Installs Ollama via `winget` if missing	`ollama.exe` already on disk
3	Starts `ollama serve` if not already responding	endpoint at `http://localhost:11434` already responds
4	Pulls `qwen2.5:7b-instruct` (~4.7 GB) if missing	model already in `~/.ollama/models/`
5	Smoke-tests inference via `/v1/chat/completions`	always runs (proves end-to-end working state)

When all five steps already pass, the script does nothing destructive and returns in seconds.

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.

Sample run output

A green run on an already-set-up machine looks like this:

```
FrameworX Local AI - Install and Verify
Default model: qwen2.5:7b-instruct
Endpoint:      http://localhost:11434/v1/chat/completions

==> Checking port 11434
  ok  Ollama already serving on 11434 (PID 6224)
==> Checking Ollama install
  ok  Ollama present at C:\Users\<user>\AppData\Local\Programs\Ollama\ollama.exe
==> Checking Ollama server is responding
  ok  Ollama server responding on http://localhost:11434
==> Checking model 'qwen2.5:7b-instruct'
  ok  Model present (4.36 GB on disk)
==> Smoke-testing inference (this also primes the model into RAM)
  ok  Inference returned in 10.3s: 'pong'

All green. Total time: 12.9s.
FrameworX is ready to talk to Local AI at http://localhost:11434/v1/chat/completions
```

Each `ok` line is a state-check that found things already correct and skipped the work. On a fresh machine, those `ok` lines are replaced with progress messages from `winget install` and `ollama pull`.

Next: configure your model in Designer

After the script reports **All green**, open Designer and confirm FrameworX can reach the model:

1. Open your solution in Designer.
2. Navigate to **Unified Namespace Data Servers**.
3. Find the **Local AI** tile. The status should resolve to **Reachable** within a few seconds.
4. Click the tile to edit 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 tile shows **Unreachable**, re-run `Install-LocalAI.ps1` — the script's smoke test will surface whatever changed (Ollama not started, model not pulled, port held by another process). If it shows **Auth required**, see SecuritySecrets Authentication for Local AI. Full configuration reference: Local AI Configuration.

If the script reports a port conflict

If port 11434 is already held by **a different process** (LM Studio, llama.cpp server, oobabooga, an old test server, etc.), the script aborts with a clear message naming the offending process. To resolve:

1. Stop the conflicting service, or move it to a different port.
2. Re-run `Install-LocalAI.ps1`.

The script intentionally does NOT kill foreign processes on its own — port 11434 is heavily used by the LLM ecosystem and a silent process kill is the wrong default.

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 `SolutionSettings.ModelSettings` to point `URL` at `http://<ollama-host-ip>:11434/v1/chat/completions`.

A future revision of this script will accept a `-RemoteHost` flag to automate steps 1 and 2.

Choosing a different model

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` for low-RAM gateways, `qwen2.5:14b-instruct` for capable servers, `llama3.1`, `mistral`, etc.).
2. In FrameworX, edit `SolutionSettings.ModelSettings` 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 — the main reference page (parent of this one).
- AI Integration — the broader AI surface in FrameworX.

In this section...