Remote and Cloud LLM Models

Running the Local AI model on a separate machine, a private/enterprise cloud, or a public cloud API — and how that call leaves the control network safely, across your security zones.

Version 10.1.5+

> **The model does not belong on the FrameworX host**
>
> FrameworX Local AI calls an OpenAI-compatible chat-completions endpoint over HTTP. For anything past a throwaway test, that endpoint runs on a **separate machine with real resources (ideally a GPU), or on a cloud service** — never on the same box as TServer. A small CPU-only model sharing the runtime host generates only **~2–4 tokens/second**, which is too slow to be useful even for a demonstration, and it steals CPU the runtime needs for scans, alarms, and clients. Pick a remote shape below and point the AI Engine endpoint at it.

Why the model runs off the FrameworX host

LLM inference is CPU-heavy: a single call on a 7B-class model uses every available core for seconds at a time. When the model shares the TServer host, every chat turn effectively pauses the runtime — Property Watch freezes, the Designer heartbeat times out, alarm reactions lag. The fix is topological: move inference off the runtime's critical path. The full tradeoff analysis and the four hosting topologies are on Local AI Deployment and Performance; this page covers the three *remote* shapes and the network/zone discipline they require.

Three remote shapes

Shape	Where the model runs	Typical use
Dedicated host on your network	A separate GPU-equipped machine on the plant / OT or IT network running Ollama (or any OpenAI-compatible server). Can be a refurbished workstation, a gaming PC, or a small GPU server. One host can serve several FrameworX solutions.	On-premises production, air-gapped sites, IP-sensitive prompts that must not leave the building. The default recommendation for real use.
Private / enterprise cloud	A model server in your own data center or private cloud (a VM with a GPU, or a managed inference service inside your tenancy), reached over the enterprise network.	Multi-site organizations that already centralize compute; consistent model governance across plants.
Public cloud API	A managed, OpenAI-compatible endpoint over HTTPS — for example Claude, GPT-class, or any provider exposing the chat-completions shape. Authenticated with an API key.	Top-tier reasoning / multi-modal quality, no local GPU to maintain. Requires outbound internet and a data-sensitivity review (see below).

From FrameworX's side all three are the same code path with a different URL — switching shape is a single field edit on the AI Engine tile. What changes between them is **which security zone the model lives in**, and therefore how the call must cross your network.

Crossing security zones

FrameworX follows the ISA-99 / IEC 62443 defense-in-depth model: the runtime that talks to PLCs and devices sits in the control zone (Purdue **Level 2**), while shared services and outbound connectivity belong to the enterprise zone (**Level 4**) and, beyond it, the public internet. A remote or cloud model lives in one of those outer zones, so every Local AI call crosses a zone boundary — and that crossing must be controlled, not a hole punched straight out of the control network.

The platform gives you the pieces to govern it:

- **Secure Multi-Port Gateway (TSecureGateway).** FrameworX's native cross-zone routing service — it relays only authorized traffic between designated points and keeps zones isolated from one another, the controlled conduit between Level 2 and Level 4. See SecureGateway Services Reference.
- **Segmented zones and conduits.** Place the model host (or the egress to the cloud provider) in an outer zone and open a single, narrow firewall conduit from the runtime to it — never expose the control zone directly. Background: Security and Performance § *Security Zones Architecture* and IEC 62443 Compliance How-to.
- **Credentials in the vault.** For any authenticated (cloud or proxied) endpoint, store the API key in SecuritySecrets and reference it from the endpoint config — it never sits in plain text in the solution. See SecuritySecrets Authentication for Local AI.

This is the same disciplined, controlled-egress pattern Tatsoft uses internally to connect its own tooling to cloud AI services such as Claude: the model is reached from the enterprise / outer zone through a governed boundary, with authentication and traffic restriction in front of it — not by reaching out directly from the process-control layer.

> **Data leaves the building with a public cloud model**
>
> The prompt a cloud model sees can include live tag values, alarm context, and batch IDs. Treat that as data egress: review what the solution sends, redact what must not leave, and get compliance sign-off before pointing FrameworX at a third-party endpoint. An on-network dedicated host avoids the question entirely.

Hand the chat to an egress server

From FrameworX 10.1.6, a solution sends its interactive AI chat to a designated egress FrameworX runtime instead of calling the LLM endpoint from the control-zone runtime. Set **Gateway IP** on the AI Engine (the `GatewayIP` field) to the egress server as `host` or `host:port`. With no port, the egress is reached on port 3101, the Production profile port. The egress server holds the endpoint configuration and the API key and makes the outbound LLM call, so the provider key for chat stays on the egress host. Leave the field blank to call the endpoint directly.

The egress is a FrameworX runtime server, not the Solution Center web service. On the egress, set its own Gateway IP to a network interface address of the egress machine itself, not a loopback address. Otherwise the egress refuses forwarded chat with *caller is not an authorized forwarding peer*.

Only interactive AI chat is routed. AI.Execute calls from server-side scripts still go directly from the runtime to the endpoint, with the solution's own endpoint settings and key, so plan the network path for those calls as well. Field reference: Local AI Configuration Reference.

Configure the endpoint

Point the AI Engine at the remote model in **Solution Capabilities AI Engine Edit Configuration** (or edit `SolutionCapabilities[LocalAI].Settings` directly). The full field reference is on Local AI Configuration Reference.

Dedicated host on your network

```
{
  "URL": "http://192.168.1.50:11434/v1/chat/completions",
  "Name": "qwen2.5:7b-instruct",
  "Authorization": "NoAuth",
  "Headers": "",
  "Info": "Ollama on a dedicated GPU host on the LAN.",
  "TimeoutSeconds": 60
}
```

Replace `192.168.1.50` with the IP address (or hostname) of the machine running your model. On that host, bind the server to the network (`OLLAMA_HOST=0.0.0.0:11434` for Ollama) and open inbound TCP 11434 only to the FrameworX server's address.

Public cloud API

```
{
  "URL": "https://api.your-provider.com/v1/chat/completions",
  "Name": "<provider-model-name>",
  "Authorization": "BearerToken /secret:LlmApiKey",
  "Headers": "",
  "Info": "OpenAI-compatible cloud endpoint; key in SecuritySecrets.",
  "TimeoutSeconds": 60
}
```

The `/secret:LlmApiKey` token pulls the key from the SecuritySecrets vault at call time, so it never appears in the solution file or an export. The setting is read fresh on every call — an edit takes effect on the next request with no restart.

Anthropic (Claude) endpoints are configured differently. The `BearerToken` shape above is the OpenAI-compatible convention. Anthropic authenticates with an `x-api-key` header plus a mandatory `anthropic-version` header, both set in the `Headers` field with `Authorization` left at `NoAuth`; and an API key that is not bound to a single workspace additionally requires an `anthropic-workspace-id` header on every request. See Local AI Configuration Reference § *Anthropic (Claude) endpoint* for the full JSON and the workspace ID lookup.

See also

- Local AI Deployment and Performance — the four hosting topologies, sizing, and the decision matrix by workload.
- Local AI Configuration Reference — every endpoint field, the ModelOptions bits, and topology JSON shapes.
- SecuritySecrets Authentication for Local AI — storing and referencing API keys safely.

- SecureGateway Services Reference — the Secure Multi-Port Gateway (TSecureGateway) cross-zone routing service.
- Security and Performance — the security-zones architecture and defense-in-depth model.

In this section...