Enabling the MessageFoundry Console on a Remote PC

A customer setup guide. This guide is for the IT administrator who wants staff to run the MessageFoundry **admin console** from their own PCs instead of only on the server that hosts the engine. It walks through the three things you need to do — prepare a certificate, open the engine to the network securely, and point each console at it.

No code changes are required; everything here is configuration.

What you're setting up

The MessageFoundry **engine** runs on a server (usually as a Windows service). The **console** is a separate desktop application that monitors and operates the engine. By default the engine only accepts connections from the **same machine** (`127.0.0.1`), so the console must run on the server.

To use the console from another PC, you:

1. Give the engine a **TLS certificate** (so the connection is encrypted), and
2. Tell the engine to **listen on the network**, then
3. Point each remote console at the engine's `https://` address.

The connection is encrypted end-to-end and every user signs in — the same authentication the local console already uses. Nothing is exposed until you make these changes deliberately.

> **Before you begin — checklist** - The engine server's hostname or IP address (e.g. `mefor-srv01.hospital.local`). - A TLS certificate for that hostname (see Step 1). - The MessageFoundry console installed on each remote PC. - A network path from the PCs to the engine on its API port (default **8765**) — open the firewall if needed. - A MessageFoundry user account for each person who will sign in.

Step 1 — Prepare a TLS certificate for the engine

The engine needs a certificate so traffic (including login credentials) is encrypted. Choose **one**:

Option	When to use it	Notes
Your organization's internal CA (e.g. Active Directory Certificate Services)	**Recommended** for most sites	Issue a server certificate for the engine's hostname. Domain-joined PCs already trust your CA, so the console needs no extra setup.
A public CA (e.g. a commercial cert)	The engine has a public DNS name	Trusted everywhere automatically.
A self-signed certificate	Small/internal setups, pilots	Works fine — each console points at the certificate with `--cacert` (Step 3).

Whichever you choose, the certificate's **Subject Alternative Name (SAN)** must include the hostname or IP address the console will connect to. You'll end up with two files (or one combined file):

- a **certificate** PEM (e.g. `engine-cert.pem`), and
- a **private key** PEM (e.g. `engine-key.pem`).

Place them somewhere the engine's service account can read, e.g. `C:\MessageFoundry\tls\`.

Step 2 — Configure the engine server

Edit the engine's configuration file, **`messagefoundry.toml`**, on the server. Add or update the `[api]` section:

```
[api]
host = "0.0.0.0"                                  # listen on the network (or a specific NIC IP)
port = 8765                                       # the API port the console connects to
tls_cert_file = "C:/MessageFoundry/tls/engine-cert.pem"
tls_key_file  = "C:/MessageFoundry/tls/engine-key.pem"
```

Notes:

- `host = "0.0.0.0"` listens on all network interfaces; you can instead use a specific address (e.g. `"10.0.0.12"`) to limit it to one network.
- If your private key is **password-protected**, supply the passphrase via the environment variable `MEFOR_API_TLS_KEY_PASSWORD` — never put it in the file.
- The engine **will refuse to start** if you open it to the network **without** a certificate (this protects you from accidentally sending credentials in clear text).

Open the firewall on the chosen port (default **8765/TCP**) so remote PCs can reach the server, then **restart the MessageFoundry service** for the change to take effect.

> **Using a reverse proxy or load balancer?** If TLS is terminated by a proxy in front of the engine instead, set `tls_terminated_upstream = true` and list the proxy's address in `trusted_proxies = ["..."]` under

> `[api]`, and have the proxy forward to the engine. See the technical reference (`REMOTE-CONSOLE.md`) for details.

Step 3 — Connect the console from a remote PC

On each PC, launch the console pointed at the engine's `https://` address:

```
messagefoundry-console --url https://mefor-srv01.hospital.local:8765
```

(or, from a command prompt, `python -m messagefoundry.console --url https://mefor-srv01.hospital.local:8765`)

Certificate trust — usually nothing to configure:

- If the engine's certificate was issued by **your organization's CA** and the PC is **domain-joined**, it's already trusted — it just works.
- If you used a **public CA**, it's also already trusted.
- If you used a **self-signed certificate** (or an internal CA not yet installed on the PC), add `--cacert` pointing at the engine's certificate file:

`messagefoundry-console --url https://mefor-srv01.hospital.local:8765 --cacert C:\MessageFoundry\tls\engine-cert.pem`

(Alternatively, install your internal CA into the PC's Windows certificate store once, and you can drop the flag.)

Finally, **sign in** with your MessageFoundry account. If multi-factor authentication is enabled, you'll be prompted for your second factor — the same as on the local console.

Step 4 — Verify it's working

- The console connects and shows the engine **Status** page with live connection counts.
- The status indicator shows the engine is reachable and refreshes on its own.

If the console can't connect, see Troubleshooting below.

Optional — require a client certificate (mutual TLS)

For higher assurance you can require each console to present its **own** certificate, so only PCs holding an approved certificate can connect — in addition to the user signing in.

- On the engine, set `tls_client_ca_file` under `[api]` to the CA that issued the console certificates.

- On each console, add `--client-cert` (and `--client-key` if the key is separate):

```
messagefoundry-console --url https://mefor-srv01.hospital.local:8765 ^ --client-cert
C:\MessageFoundry\tls\console.pem --client-key C:\MessageFoundry\tls\console-key.pem
```

This is optional and off by default.

Troubleshooting

What you see	What it means / what to do
`certificate verify failed` / "not trusted by the trust provider"	The PC doesn't trust the engine's certificate. Add `--cacert <file>`, or install the issuing CA into the PC's Windows certificate store.
`refusing to use plaintext http to non-loopback host …`	You used an `http://` address to a remote engine. Use the `https://` address (configure the certificate in Step 2).
The **engine** won't start after editing the config	You set `host` to a network address without a certificate. Add `tls_cert_file` (+ `tls_key_file`) under `[api]`, or revert `host` to `127.0.0.1`.
"hostname mismatch" when connecting	The certificate's name (SAN) doesn't match the address in `--url`. Reissue the certificate for the correct hostname/IP.
Console can't reach the server at all	Check the firewall on the engine server (default port **8765/TCP**) and that the service is running.

Security notes

- **Stays on your network.** The engine runs on-premises; remote access is to *your* server over *your* network — no data leaves your environment.
- **Encrypted in transit.** All console–engine traffic, including login, is protected by TLS.
- **Authenticated and audited.** Every user signs in, and access to patient data is recorded with the acting user. We recommend enabling **multi-factor authentication** for any network-exposed engine.
- `--insecure` only permits unencrypted `http` on a trusted test network; it does **not** weaken certificate checking for `https`. Don't use it for production.

More information

- **Technical reference** (full `[api]` settings, in-process vs. upstream TLS): `docs/REMOTE-CONSOLE.md`
- **Security overview** (authentication, TLS, auditing): `docs/SECURITY.md`
- **Running the engine as a service**: `docs/SERVICE.md`

If you'd like help planning this rollout, contact your MessageFoundry support contact.