TUTORIAL · JULY 13, 2026

Networking with Delphi

A five-part course on practical networking for Delphi developers — from IP addresses and ports to TCP sockets, HTTP, and REST web services, with runnable Delphi code in every part.

By Dr. Holger Flick



There's a moment that finds every Delphi desktop developer eventually: the requirement lands that *the app needs to talk to a REST API*, or *the service runs on another machine now*. You wire it up, press F9, and get **"connection refused"** — an error that assumes a mental model nobody ever gave us.

This tutorial is that missing sit-down, delivered as a five-part article series. No subnet arithmetic, no OSI-layer liturgy, no packet header diagrams — just the practical mental models that make networking errors instantly legible, each part building on the one before it, and real Delphi code in every part: the modern RTL's `TSocket`, `System.Net.HttpClient`, and WebBroker.

Read the parts in order — each one starts exactly where the previous one ended.

Part 1 — IP addresses, ports, and localhost

IP Addresses, Ports, and localhost: Networking for Delphi Devs

The three concepts everything else stands on. What an IP address actually is, what a port selects, and why `127.0.0.1` always means *this machine*. By the end you can look at `127.0.0.1:8080`, know exactly what each piece means, and diagnose the two most common connection errors before opening a search engine.

Part 2 — Names: DNS, the hosts file, and DHCP

DNS, the hosts File, and DHCP: Networking for Delphi Devs

You almost never type IP addresses — you type names. This part covers the pipeline that turns a name into an address (the resolver, the hosts file, DNS) and, in the other direction, who gave your machine its own address in the first place (DHCP). An entire class of *"it works on my machine"* bugs becomes diagnosable in minutes.

Part 3 — TCP connections and sockets

TCP Connections and Sockets in Delphi

What a connection actually *is*. Why one side listens while the other connects, why "refused", "reset", and "timed out" are three completely different failures, and why TCP hands you a stream of bytes rather than tidy messages — the number one beginner trap. The payoff: you open a raw connection from Delphi and speak to a real web server by hand, with nothing but `TSocket`.

Part 4 — How a web server actually works

How a Web Server Actually Works: HTTP for Delphi Developers

The exchange you typed by hand in Part 3, properly introduced: requests, responses, methods, status codes, and headers. Then the other side of the wire — a web server is a program, not a place — proven in Delphi with a WebBroker request handler and a client calling it via `System.Net.HttpClient`.

Part 5 — Web services are just endpoints

Web Services Are Just Endpoints: REST and JSON in Delphi

The moment your server returns data instead of pages, it stops being "a website" and becomes a web service. This finale demystifies *web service*, *REST API*, and *endpoint* word by word, shows the full round trip — a WebBroker action serving JSON, consumed by a desktop app with `THTTPClient` — and closes with how TMS XData turns all of it into a few attributes on an ordinary Delphi interface.

What you'll be able to do afterwards

- Read `192.168.1.20:8080` the way you read a file path, and know which side of a connection failed and why.
- Trace a hostname from the string in your FireDAC connection string to the IP address a socket actually dials.
- Open, use, and reason about TCP connections directly from the Delphi RTL — no components, no third-party libraries.
- Build an HTTP server with WebBroker, call it with `System.Net.HttpClient`, and read HTTP traffic like plain text — because it is.

- Say *endpoint*, *REST*, and *JSON* in meetings without crossing your fingers under the table.

INDEPENDENT · AD-FREE · READER-SUPPORTED

Free to read. Not free to make.

Every tutorial, article, and line of example code on Holger's Code is published without a paywall and without ads. This page is about what keeps it that way.

Professional work, published free

The content here is held to the same standard as professional client work: every tutorial is built as a real project, tested against current versions, documented step by step, and revisited when the frameworks move on. A single in-depth tutorial routinely takes days — and that's before the bills behind it: servers, storage, build infrastructure, licenses, and the hardware everything is verified on.

If an article or tutorial here has saved you an afternoon of debugging, taught you a technique, or helped you ship, it created real value for you. Supporting it isn't charity — it's paying for professional work you found useful, at whatever price you choose. And if you can't, keep reading anyway. That's what the free part means.

Ways to support

KO-FI

Buy me a coffee

A one-time tip — no account, no subscription, any amount.

BOOKS & COURSES

Buy the paid work

The paid tier of everything published free here — and you get something lasting in return.

SPREAD THE WORD

Share an article

Send a tutorial to a colleague, cite an article, subscribe to the RSS feed.

Nothing here goes behind a paywall, whether you chip in or not. But if something on this site earned its keep today, this is where to say so.

Support on Ko-fi —