TUTORIAL · JULY 13, 2026

Netzwerktechnik mit Delphi

By Dr. Holger Flick



Es gibt einen Moment, der früher oder später jeden Delphi-Desktop-Entwickler ereilt: Die Anforderung landet auf dem Tisch, dass *die App mit einer REST-API sprechen muss*, oder dass *der Dienst jetzt auf einer anderen Maschine läuft*. Sie verdrahten alles, drücken F9 — und bekommen **„connection refused"**: einen Fehler, der ein mentales Modell voraussetzt, das uns nie jemand vermittelt hat.

Dieses Tutorial ist genau dieses fehlende Gespräch, ausgeliefert als fünfteilige Artikelserie. Keine Subnetz-Arithmetik, keine OSI-Schichten-Liturgie, keine Diagramme von Paket-Headern — nur die praktischen Denkmodelle, die Netzwerkfehler auf einen Blick lesbar machen, jeder Teil auf dem vorherigen aufbauend, und echter Delphi-Code in jedem Teil: das moderne `TSocket` der RTL, `System.Net.HttpClient` und WebBroker.

Lesen Sie die Teile der Reihe nach — jeder setzt genau dort an, wo der vorherige aufgehört hat.

Teil 1 — IP-Adressen, Ports und localhost

IP-Adressen, Ports und localhost: Netzwerk-Grundlagen für Delphi-Entwickler

Die drei Konzepte, auf denen alles Weitere steht. Was eine IP-Adresse wirklich ist, was ein Port auswählt und warum `127.0.0.1` immer *diese Maschine* meint. Am Ende können Sie `127.0.0.1:8080` ansehen, wissen genau, was jeder Teil bedeutet, und diagnostizieren die zwei häufigsten Verbindungsfehler, bevor Sie überhaupt eine Suchmaschine öffnen.

Teil 2 — Namen: DNS, die hosts-Datei und DHCP

DNS, die hosts-Datei und DHCP: Netzwerk-Grundlagen für Delphi-Entwickler

Sie tippen fast nie IP-Adressen — Sie tippen Namen. Dieser Teil behandelt die Pipeline, die einen Namen in eine Adresse verwandelt (den Resolver, die hosts-Datei, DNS) und, in die andere Richtung, wer Ihrer Maschine überhaupt ihre eigene Adresse gegeben hat (DHCP). Eine ganze Klasse von *„bei mir läuft's doch"*-Fehlern wird damit in Minuten diagnostizierbar.

Teil 3 — TCP-Verbindungen und Sockets

TCP-Verbindungen und Sockets in Delphi

Was eine Verbindung wirklich *ist*. Warum eine Seite lauscht, während die andere verbindet, warum „refused", „reset" und „timed out" drei völlig verschiedene Fehler sind, und warum TCP Ihnen einen Strom von Bytes reicht statt ordentlicher Nachrichten — die häufigste Anfängerfalle. Der Lohn: Sie öffnen von Delphi aus eine rohe Verbindung und sprechen von Hand mit einem echten Web Server, mit nichts als `TSocket`.

Teil 4 — Wie ein Web Server wirklich funktioniert

Wie ein Web Server wirklich funktioniert: HTTP für Delphi-Entwickler

Der Austausch, den Sie in Teil 3 von Hand getippt haben, ordentlich vorgestellt: Requests, Responses, Methoden, Statuscodes und Header. Dann die andere Seite der Leitung — ein Web Server ist ein Programm, kein Ort — in Delphi bewiesen mit einem WebBroker-Request-Handler und einem Client, der ihn über `System.Net.HttpClient` aufruft.

Teil 5 — Web Services sind nur Endpoints

Web Services sind nur Endpoints: REST und JSON in Delphi

In dem Moment, in dem Ihr Server Daten statt Seiten zurückgibt, hört er auf, „eine Website" zu sein, und wird zu einem Web Service. Dieses Finale entmystifiziert *Web Service*, *REST-API* und *Endpoint* Wort für Wort, zeigt den vollständigen Rundlauf — eine WebBroker-Action, die JSON ausliefert, konsumiert von einer Desktop-App mit `THTTPClient` — und schließt damit, wie TMS XData all das in ein paar Attribute auf einem ganz gewöhnlichen Delphi-Interface verwandelt.

Was Sie danach können werden

- `192.168.1.20:8080` so lesen, wie Sie einen Dateipfad lesen, und wissen, welche Seite einer Verbindung gescheitert ist und warum.
- Einen Hostnamen von der Zeichenkette in Ihrem FireDAC-Connection-String bis zu der IP-Adresse verfolgen, die ein Socket tatsächlich anwählt.
- TCP-Verbindungen direkt aus der Delphi-RTL öffnen, nutzen und durchdenken — keine Komponenten, keine Fremdbibliotheken.
- Einen HTTP-Server mit WebBroker bauen, ihn mit `System.Net.HttpClient` aufrufen und HTTP-Verkehr wie reinen Text lesen — denn genau das ist er.

- *Endpoint*, *REST* und *JSON* im Meeting sagen, ohne unter dem Tisch die Finger zu kreuzen.

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 —